

Universidade Federal do Piauí
Centro de Ciências da Natureza
Programa de Pós-Graduação em Ciência da Computação

Avaliação de Desempenho de Redes Veiculares Ad Hoc com RSUs e Filas de Prioridade

Jorge Rafael Loiola de Macêdo

Teresina-PI, Dezembro de 2024

Jorge Rafael Loiola de Macêdo

Avaliação de Desempenho de Redes Veiculares Ad Hoc com RSUs e Filas de Prioridade

Dissertação de Mestrado apresentada ao Programa de Pós-Graduação em Ciência da Computação da UFPI (área de concentração: Sistemas de Computação), como parte dos requisitos necessários para a obtenção do Título de Mestre em Ciência da Computação.

Universidade Federal do Piauí – UFPI

Centro de Ciências da Natureza

Programa de Pós-Graduação em Ciência da Computação

Orientador: Prof. Dr. Francisco Airton Pereira da Silva

Teresina-PI

Dezembro de 2024

FICHA CATALOGRÁFICA
Universidade Federal do Piauí
Sistema de Bibliotecas UFPI - SIBi/UFPI
Biblioteca Setorial do CCN

M141a Macêdo, Jorge Rafael Loiola de.
Avaliação de desempenho de redes veiculares ad hoc com RSUs e filas de prioridade / Jorge Rafael Loiola de Macêdo. -- 2025.
89 f.

Dissertação (Mestrado) - Universidade Federal do Piauí. Centro de Ciências da Natureza. Programa de Pós-Graduação em Ciências da Computação, Teresina, 2025.
"Orientador: Prof. Dr. Francisco Airton Pereira da Silva".

1. Sistemas operativos específicos. 2. Sistemas de Transporte Intelignetes (ITS). 3. Roadside Units (RSUs). 4. Redes de Petri Estocásticas (SPN). 5. Escalonamento. I. Silva, Francisco Airton Pereira da. II. Titulo.

CDD 005.268

Bibliotecária: Caryne Maria da Silva Gomes - CRB3/1461

Jorge Rafael Loiola de Macêdo

Avaliação de Desempenho de Redes Veiculares Ad Hoc com RSUs e Filas de Prioridade

Dissertação de Mestrado apresentada ao Programa de Pós-Graduação em Ciência da Computação da UFPI (área de concentração: Sistemas de Computação), como parte dos requisitos necessários para a obtenção do Título de Mestre em Ciência da Computação.

gov.br Documento assinado digitalmente
FRANCISCO AIRTON PEREIRA DA SILVA
Data: 18/02/2025 10:17:42-0300
Verifique em https://validar.iti.gov.br

Prof. Dr. Francisco Airton Pereira da Silva
Orientador

gov.br Documento assinado digitalmente
ANDSON MARREIROS BALIEIRO
Data: 18/02/2025 17:22:51-0300
Verifique em https://validar.iti.gov.br

Prof. Dr. Andson Marreiros Balieiro
Membro da Banca

gov.br Documento assinado digitalmente
JULIANA OLIVEIRA DE CARVALHO
Data: 17/02/2025 17:40:26-0300
Verifique em https://validar.iti.gov.br

Profa. Dra. Juliana Oliveira de Carvalho
Membro da Banca

gov.br Documento assinado digitalmente
GERALDO PEREIRA ROCHA FILHO
Data: 17/02/2025 17:15:06-0300
Verifique em https://validar.iti.gov.br

Prof. Dr. Geraldo Pereira Rocha Filho
Membro da Banca

Teresina-PI
Dezembro de 2024

À minha amada esposa Priscilla pelos sonhos que sonhamos e vivemos juntos.

"Tudo que aprendemos nos engrandece de alguma forma.
Um dia partiremos e quando esse momento chegar,
o conhecimento de uma vida irá conosco, ou
o deixaremos como nosso legado"
(Jorge Rafael Loiola de Macêdo)

Resumo

O aumento constante da frota de veículos tem tornado o trânsito mais complexo, tanto para veículos convencionais quanto para os autônomos. Para lidar com essa complexidade, surgem os Sistemas de Transporte Inteligentes (ITS) e as redes veiculares ad hoc (VANETs), que permitem a troca de informações entre veículos e infraestrutura, melhorando a segurança e o controle do tráfego. No entanto, a implementação dessas redes enfrenta desafios significativos, como o elevado custo de infraestrutura e a complexidade crescente à medida que a rede se expande, o que pode resultar em falhas de comunicação e perda de pacotes. Este trabalho propõe um modelo de Redes de Petri Estocásticas (SPN) para avaliar e otimizar o desempenho de redes veiculares ad hoc (VANETs), focando em cenários que envolvem a utilização de Roadside Units (RSUs) com múltiplas filas de prioridade e escalonamento circular. O estudo modela o comportamento de RSUs e servidores Fog sob diferentes cargas de trabalho, avaliando métricas como tempo de resposta, utilização de recursos e probabilidade de descarte. O modelo visa permitir que projetistas e engenheiros possam avaliar de forma detalhada e antecipada a capacidade das RSUs em diferentes configurações de rede, ajustando parâmetros como transições e capacidades de fila ou processamento antes da implementação física. A proposta também aborda a complexidade inerente a esses sistemas, proporcionando uma solução que avalia as interações entre veículos e infraestrutura de forma eficiente.O modelo proposto permite simular e verificar o impacto de diferentes configurações na performance da rede, possibilitando a identificação de gargalos e áreas de melhoria. Essa abordagem facilita o planejamento de futuras implementações e garante uma maior eficiência na alocação de recursos, reduzindo custos e riscos associados à implantação de redes veiculares. Ao oferecer um ambiente de simulação flexível, o modelo também contribui para o desenvolvimento de soluções mais robustas e escaláveis, capazes de lidar com o aumento da complexidade e da demanda no trânsito urbano moderno. Os experimentos foram validados por meio de uma ferramenta customizada que replica os cenários modelados, confirmando a precisão e a aplicabilidade do modelo. Os resultados demonstram que o uso de escalonamento circular entre classes de prioridade permite melhor distribuição dos recursos, garantindo atendimento adequado às requisições críticas sem prejudicar as de menor prioridade. A abordagem proposta melhora significativamente o desempenho em situações críticas, otimizando a alocação de recursos computacionais e priorizando mensagens de maior relevância. Este trabalho contribui para o planejamento e gestão de sistemas complexos, oferecendo um suporte analítico robusto para a tomada de decisão em redes veiculares, auxiliando no dimensionamento de recursos, eficiência e confiabilidade.

Palavras-chaves: ITS, VANETs, RSUs, Desempenho, Prioridade, Filas, Escalonamento, SPN

Abstract

The constant increase in the vehicle fleet has made traffic more complex, both for occasional and independent vehicles. To deal with this complexity, Intelligent Transportation Systems (ITS) and vehicular ad hoc networks (VANETs) have emerged, which allow the exchange of information between vehicles and infrastructure, improving safety and traffic control. However, the implementation of these networks faces significant challenges, such as the high cost of infrastructure and the increasing complexity as the network expands, which can result in communication failures and packet loss. This work proposes a Stochastic Petri Nets (SPN) model to evaluate and optimize the performance of vehicular ad hoc networks (VANETs), focusing on scenarios involving the use of Roadside Units (RSUs) with multiple priority queues and circular scheduling. The study models the behavior of RSUs and Fog servers under different workloads, evaluating statistics such as response time, resource utilization, and drop probability. The model aims to enable designers and engineers to assess in detail and in advance the capacity of RSUs in different network configurations, adjusting parameters such as transitions and queue or processing capacities before physical implementation. The proposal also addresses the inherent complexity of these systems, providing a solution that evaluates the interactions between vehicles and infrastructure in an efficient and scalable way. The proposed model allows simulating and verifying the impact of different configurations on network performance, enabling the identification of bottlenecks and areas for improvement. This approach facilitates the planning of future implementations and ensures greater efficiency in the allocation of resources, reducing costs and risks associated with the deployment of vehicular networks. By offering a flexible simulation environment, the model also contributes to the development of more robust and scalable solutions, capable of dealing with the increased complexity and demand in modern urban traffic. The experiments were validated using a customized tool that replicates the modeled scenarios, confirming the accuracy and applicability of the model. The results demonstrate that the use of circular scheduling between priority classes allows for better distribution of resources, ensuring adequate service to critical requests without deficiencies such as lower priority requests. The proposed approach significantly improves performance in critical situations, optimizing the allocation of computational resources and prioritizing messages of greater relevance. This work contributes to the planning and management of complex systems, offering robust analytical support for decision-making in vehicular networks, assisting in resource sizing, efficiency, and reliability.

Keywords: ITS, VANETs, RSUs, Performance, Priority, Queues, Scheduling, SPN

Lista de ilustrações

Figura 1 – Componentes principais de uma SPN. 13
Figura 2 – Exemplo simples de SPN. 13
Figura 3 – Exemplo de SPN com fila de processamento. 14
Figura 4 – Exemplo de SPN com fila de chegada e fila de processamento. 14
Figura 5 – Metodologia de desenvolvimento do modelo analítico. 29
Figura 6 – Arquitetura proposta 32
Figura 7 – Modelo com duas máquinas, contendo uma única fila, que pode conter requisições prioritárias e não prioritárias 34
Figura 8 – Resultados da análise do modelo considerando diferentes números de núcleos para RSU no Cenário 1 40
Figura 9 – Resultados da análise do modelo considerando diferentes números de núcleos para *Fog*, no Cenário 2 42
Figura 10 – Modelo contendo três filas de prioridades, usando bloqueio entre as filas 44
Figura 11 – Resultados da análise Caso 2, para RSU com 8 núcleos 46
Figura 12 – Resultados da análise Caso 2, para RSU com 20 núcleos 47
Figura 13 – Modelo contendo três filas de prioridades, usando escalonamento entre as filas 48
Figura 14 – Resultados da análise Caso 3, para RSU com 8 núcleos 51
Figura 15 – Resultados da análise Caso 3, para RSU com 20 núcleos 52
Figura 16 – Fluxo de Validação 55
Figura 17 – Modelo SPN da Validação 56
Figura 18 – Ambiente na AWS 57
Figura 19 – Resultado da Validação 60

Lista de tabelas

Tabela 1 – Trabalhos relacionados. 21
Tabela 2 – Definições utilizadas no modelo SPN da Figura 7 34
Tabela 3 – Principais transições temporizadas utilizadas no modelo SPN da Figura 7 34
Tabela 4 – Relação entre `AR1`, `ADC1` e Interpretação dos Valores 35
Tabela 5 – Expressões das métricas utilizadas no modelo SPN Base 36
Tabela 6 – Parâmetros para RSU - Cenário 01 37
Tabela 7 – Parâmetros para *Fog* - Cenário 02 37
Tabela 8 – Componentes adicionados no modelo sem bloqueio entre classes 44
Tabela 9 – Métricas utilizadas no modelo SPN com três filas (Figura 10) 45
Tabela 10 – Expressões de Guarda 50
Tabela 11 – Principais transições temporizadas utilizadas no modelo SPN da Figura 17 56
Tabela 12 – Configuração do experimento no ambiente da AWS 57
Tabela 13 – Parâmetros para validação 59
Tabela 14 – Resultado Test-T 60

Lista de abreviaturas e siglas

AD Atraso de Chegada.

Amazon EC2 *Amazon Elastic Compute Cloud.*

Amazon VPC *Amazon Virtual Private Cloud.*

AR Taxa de Chegada.

AWS *Amazon Web Services.*

DP Probabilidade de Descarte.

DSRC *Dedicated Short Range Communication.*

EMR-ICN Roteamento de Mensagens de Emergência em Redes Intermitentes.

FCFS *First Come, First Served.*

FRIIS *Free Space Propagation Model.*

GSPN *Rede de Petri Estocástica Generalizada.*

IIA-ORD *Intersection Influence Analysis System for Optimal RSU Deployment.*

IoT *Internet of Things.*

IP Internet Protocol.

ITS *Intelligent Transportation Systems.*

LCFS *Last Come, First Served.*

LD *Log-Distance Propagation Model.*

LSTM Memória de Longo e Curto Prazo.

MRT Tempo Médio de Resposta.

NK *Nakagami Fading Model.*

NTP *Network Time Protocol.*

OBU *On-board Unit.*

PDR Taxa de Entrega de Pacotes.

PN *Rede de Petri*.

QoS Qualidade de Serviço.

RSU *Roadside Unit*.

RTA *Regional Trusted Authority*.

RTEAM *Risk-Based Trust Evaluation Advanced*.

SD Sistema Distribuído.

SDN Redes Definidas por Software.

SIRO *Service In Random Order*.

SMT Teorias de Módulos de Satisfatibilidade.

SPN Rede de Petri Estocástica.

TA *Trusted Authority*.

TCP Transmission Control Protocol.

TIE Tempo Intermediário Externo.

TII Tempo Intermediário Interno.

TOPSIS Técnica para a Ordem de Preferência por Similaridade à Solução Ideal.

TR *Two-Ray Ground Propagation Model*.

UAVs Veículos Aéreos Não Tripulados.

UFS Utilização do Servidor de *Fog*.

UQFS Utilização da Fila no Servidor de *Fog*.

UQRSU Utilização da Fila na RSU.

URSU Utilização da RSU.

V2I *Vehicle to Infrastructure*.

V2V *Vehicle to Vehicle*.

VANET *Vehicular ad hoc Network*.

WAVE *Wireless Access for Vehicular Environment*.

Sumário

1 INTRODUÇÃO . . . 1
1.1 Escopo e motivação . . . 2
1.2 Definição do Problema . . . 3
1.3 Objetivos . . . 4
1.4 Publicações . . . 4
1.5 Estrutura do Trabalho . . . 5

2 REFERENCIAL TEÓRICO . . . 7
2.1 Internet of Things . . . 7
2.1.1 Computação em Edge . . . 8
2.1.2 Computação em Fog . . . 9
2.2 Vehicular ad hoc Networks . . . 10
2.2.1 Tipos de Comunicação . . . 10
2.2.2 Principais Componentes de uma VANET . . . 11
2.3 Redes de Petri Estocásticas . . . 12
2.4 Técnicas de Validação de Modelos . . . 15
2.4.1 Medições de Sistemas Reais . . . 15
2.4.2 Resultados Teóricos . . . 16
2.4.3 Técnicas Estatísticas para Validação . . . 16
2.4.3.1 Intervalo de Confiança para a Média . . . 16
2.4.3.2 Comparação de Duas Alternativas . . . 17
2.4.3.3 Determinação do Tamanho da Amostra . . . 18

3 TRABALHOS RELACIONADOS . . . 21
3.1 Tipo de Modelo . . . 23
3.2 Métricas . . . 24
3.3 Uso de RSUs . . . 25
3.4 Múltiplas Filas de Prioridade . . . 25
3.5 Usa Escalonamento entre filas de prioridade . . . 26
3.6 Foco . . . 26

4 METODOLOGIA . . . 29

5 ARQUITETURA E MODELOS SPN . . . 31
5.1 Visão Geral . . . 31
5.1.1 Suposições e discussões . . . 32

5.2 Modelo SPN Base, com 1 Classe de Prioridade Genérica 33
5.2.1 Métricas 35
5.2.2 Estudo de Caso 1 36
5.2.2.1 Cenário 1 37
5.2.2.2 Cenário 2 39
5.3 Modelos SPN estendidos, com 3 Classes de Prioridades 41
5.3.1 Modelo com Bloqueio entre Classes 41
5.3.2 Estudo de Caso 2 43
5.3.3 Modelo com Escalonamento entre Classes 45
5.3.4 Estudo de Caso 3 49
5.4 Conclusões Capítulo 49

6 VALIDAÇÃO 53
6.1 Processo de Validação 53
6.2 Modelo, Ambiente do Experimento e Ferramenta de Validação . . . 56
6.2.1 Modelo 56
6.2.2 Ambiente do Experimento 57
6.2.3 Ferramenta de Validação 58
6.2.4 Cenário de Validação 58
6.3 Resultados 59

7 CONCLUSÕES 61
7.1 Trabalhos Futuros 62

REFERÊNCIAS 63

1 Introdução

Em 2023 foram vendidos 1.720.834 carros novos no Brasil, representando aumento de 9,13% em relação ao ano de 2022 (FENABRAVE, 2024). Estimativas apontam que em 2030 haverá cerca de 20,8 milhões de veículos autônomos só nos Estados Unidos (WIRE, 2018). O Brasil conta com uma frota total de 119,1 milhões de veículos [1] e uma frota ativa [2] de 82,1 milhões de veículos, onde no ano de 2023 foram registrados 650.359 acidentes que ocasionaram 14.986 óbitos (RENASET, 2024). O aumento anual da frota intensifica a complexidade do trânsito e requer melhores tomadas de decisão, seja em veículos conduzidos por pessoas, seja em veículos de condução autônoma. Informações sobre as condições da via melhoram a segurança e eficiência dos sistemas de transporte (SINHA; MISHRA, 2014), bem como diminuem as taxas de acidentes (ISMAIL et al., 2022).

O trânsito de veículos é dinâmico e exige respostas rápidas que otimizem o seu fluxo. *Intelligent Transportation Systems* (ITS) permitem a troca de informações entre os veículos, ao tempo que fornecem dados para o controle de tráfego, aumentando a segurança e conforto no trânsito (ISMAIL et al., 2022; SEEMA et al., 2022; SINHA; MISHRA, 2014; HOTA et al., 2022; BOUKHALFA et al., 2021). As *Vehicular ad hoc Networks* (VANETs) fazem parte dos ITS e permitem comunicação eficiente e segura, seja *Vehicle to Vehicle* (V2V), seja *Vehicle to Infrastructure* (V2I) (HOTA et al., 2022). As *Roadside Units* (RSUs) são equipamentos instalados ao longo das vias e interseções (HOTA et al., 2022), que processam informações enviadas pelos veículos e por outras unidades. As informações processadas são encaminhadas para os veículos próximos, outras RSUs e demais componentes do ITS. Esse modo de comunicação se refere a V2I, que geralmente abrange curta distância entre um veículo e um nó da infraestrutura (LIM et al., 2021). Os veículos são equipados com *On-board Units* (OBUs) para se comunicarem com as RSUs (HOTA et al., 2022).

A implementação de VANETs pode envolver elevado custo por conta da infraestrutura a ser instalada, como dispositivos de comunicação e processamento, fornecimento de energia para os equipamentos e acesso às redes de dados para funcionamento de RSUs. Determinar a quantidade e localização dos equipamentos para maximizar a cobertura é um desafio que afeta o custo (GHOSH; MISRA; CHAKRABORTY, 2023; ULLAH et al., 2023), além disso a topologia das vias e obstáculos como construções e árvores afeta a propagação do sinal (HOTA et al., 2022; TAN; CHUNG, 2021). A adaptação de veículos que não possuam recursos de comunicação V2V ou V2I, também é um fator que limita a

[1] Carros, motos, caminhões, dentre outras categorias
[2] Veículos com pelo menos um licenciamento anual ou infrações no últimos 10 anos (RANAINF)

adoção de VANETs.

À medida que a rede cresce, também aumenta sua complexidade, pois a infraestrutura passa a incluir equipamentos com características e configurações diversas. Assim, o volume de dados trafegando na rede aumenta, bem como configurações heterogêneas de equipamentos favorecem a propensão a erros, como perda de pacotes, atrasos e exposição a ações maliciosas (TAN; CHUNG, 2021; JABALLAH; CONTI; LAL, 2019). Redes veiculares são sistemas críticos devido à necessidade de alta confiabilidade, baixa latência, disponibilidade e segurança para suportar cenários que envolvem riscos à vida e à infraestrutura. A comunicação confiável e eficiente entre veículos e RSUs é uma das principais preocupações do ITS (HOTA et al., 2022).

Desta forma, avaliar grandes Sistemas Distribuídos (SDs) no mundo real nem sempre é possível, além de implicar em testes caros, demorados e de difícil reprodução (HIROFUCHI; LEBRE; POUILLOUX, 2015). Há diversas pesquisas relacionadas à conectividade em diferentes cenários, entretanto poucas delas oferecem resultados experimentais reais para corroborar suas alegações, mesmo que em pequena escala (RITO et al., 2023). Dado o custo de implementação de SDs, as ferramentas de simulação surgem como alternativa atraente (CASANOVA et al., 2013), devido à sua eficiência e escalabilidade (DOBRE; POP; CRISTEA, 2008) para avaliar configurações apropriadas e compreender o comportamento do sistema propondo otimizações. A análise prévia permite dimensionar corretamente o sistema e minimizar o custo envolvido, seja na ampliação, seja na redução da quantidade de dispositivos no SD. Embora ferramentas de simulação permitam criar cenários realistas (DOBRE; POP; CRISTEA, 2008), elas enfrentam desafios em representar interações complexas e dinâmicas em redes de alta densidade. Este trabalho propõe um modelo que aborda essas lacunas ao integrar técnicas de Redes de Petri Estocásticas (SPNs) para simulações mais robustas e detalhadas.

1.1 Escopo e motivação

A conectividade entre veículos e infraestrutura tem se mostrado uma aliada na segurança, condução e otimização do fluxo de trânsito, principalmente em ambientes urbanos de alta complexidade. O advento de sistemas como as VANETs, integrados ao ITS, oferecem uma abordagem eficiente para enfrentar os desafios associados ao aumento da frota de veículos, bem como às demandas crescentes por segurança e fluidez no trânsito. O escopo deste trabalho está centrado na avaliação de desempenho de VANETs, com ênfase no impacto impacto das filas de prioridade em RSUs e na camada de *Fog*. Ele aborda a priorização de diferentes tipos de requisições em cenários críticos que geram riscos à vida, como desastres naturais, acidentes ou emergências de saúde.

Motivações para este estudo incluem a necessidade de identificar gargalos e melhorar

o gerenciamento de filas em cenários de tráfego denso, bem como garantir que veículos de emergência recebam a devida prioridade em situações que exigem respostas rápidas. A capacidade de reorganizar o tráfego em tempo real e garantir a identificação de rotas alternativas tem impacto direto na redução de acidentes, eficiência das operações de resgate e atendimento a situações de risco à segurança pública, seja individual, seja coletiva. É importante ressaltar que a priorização do tráfego de urgência e a redução de acidentes resultam na diminuição de custos com alocação de pessoal, veículos, consumo de combustível e materiais necessários aos resgates, além de liberar recursos hospitalares para outros tipos de atendimento. O escopo deste trabalho **não inclui**:

- **Segurança da Comunicação:** As facilidades proferidas pelas VANETs trazem desafios relacionados à segurança robusta. Dentre as questões envolvidas há ênfase em incluir mecanismos de autenticação, autorização e controle de acesso (LIM et al., 2021). Dentre os problemas mais sérios se destaca o ataque de negação de serviço (DoS), pois se a rede estiver inacessível poderá causar acidente ou qualquer evento indesejado (SINHA; MISHRA, 2014)
- **Qualidade de Serviço (QoS):** Permite que aplicações de alta prioridade e tráfego sejam executados de forma confiável, mesmo quando a capacidade da rede é limitada (MAZHAR et al., 2023). Para alcançar QoS, cada aplicação necessita de prioridades diferenciadas e possui critérios específicos de atraso e confiabilidade (ISMAIL et al., 2022). O desafio está em garantir a qualidade apropriada para as diversas classes de serviço (STRZĘCIWILK, 2023).

Embora segurança da comunicação e QoS sejam aspectos fundamentais em sistemas críticos, a exclusão destes aspectos ocorreu para possibilitar uma análise detalhada do desempenho computacional da rede, permitindo estudar de forma isolada as interações entre as filas de prioridade, os tempos de resposta e a utilização de recursos. Características como segurança e QoS são reconhecidas como essenciais e podem ser exploradas em trabalhos futuros, complementando a análise de desempenho proposta.

1.2 Definição do Problema

O crescimento da frota de veículos e a inerente complexidade do trânsito moderno impõem desafios significativos para a organização, fluidez e planejamento do tráfego. Esses desafios são ainda mais críticos em cenários imprevisíveis, que demandam respostas rápidas e coordenadas, tais como emergências médicas e acidentes graves. A principal dificuldade reside em prever o comportamento do sistema em situações dinâmicas.

Ferramentas eficazes para prever o desempenho de sistemas são essenciais no planejamento do trânsito. Além disso, o gerenciamento eficiente de múltiplas requisições

com diferentes níveis de prioridade torna-se essencial para minimizar o risco à vida, exigindo soluções que priorizem mensagens críticas. Este estudo aborda essas lacunas ao propor um modelo com múltiplas filas de prioridade, que facilite o planejamento e a análise do desempenho em sistemas complexos, permitindo simulações que auxiliem no suporte à tomada de decisão. Para guiar o desenvolvimento deste trabalho, foram formuladas as seguintes perguntas de pesquisa:

Pergunta Principal:

- Quais são os impactos de diferentes configurações de filas de prioridade sobre o desempenho das *Roadside Units* (RSUs) em cenários com alta densidade de tráfego?

Pergunta Secundária:

- Quais são as limitações da capacidade de processamento das RSUs em cenários de sobrecarga de requisições?

1.3 Objetivos

Com base nas perguntas de pesquisa apresentadas, este trabalho tem como objetivo principal desenvolver um modelo em Rede de Petri Estocástica (SPN) para avaliar o desempenho de redes veiculares com RSUs, focando na priorização de requisições em cenários críticos.

Os objetivos específicos incluem:

- Desenvolver um modelo SPN para simular o comportamento de filas prioritárias em redes veiculares.
- Analisar como diferentes configurações de filas de prioridade afetam o desempenho de RSUs na camada *Edge*, bem como servidores na camada *Fog*, para identificar gargalos e otimizar recursos em situações de alta demanda.
- Avaliar o impacto de grandes volumes de requisições sobre o tempo de resposta, probabilidade de descarte, utilização das camadas *Edge* e *Fog*.

1.4 Publicações

Estes são os trabalhos publicados durante o mestrado:

- Relacionado ao tema da dissertação

  - MACÊDO, J., Carvalho, V. Andrade, E. and Silva, F. A. Modelagem e Análise da Comunicação em VANETs com uso de RSUs. In: **Anais do XXI Workshop em Desempenho de Sistemas Computacionais e de Comunicação**. SBC, 2022. p. 96-107.

- Relacionado a outros temas

  - Silva, F. A., Carvalho, A., Miqueias, J., Macedo, J., Carvalho, J., Callou, G. . Availability assessment and sensitivity analysis of an MBaaS platform. In: **International Journal of Computational Science and Engineering**, 2024, 27(3), 326-340.

1.5 Estrutura do Trabalho

O restante deste trabalho está organizado da seguinte forma. O Capítulo 2 apresenta referencial teórico, com os principais conceitos necessários para a compreensão deste trabalho. O Capítulo 3 apresenta os trabalhos relacionados, comparando-os com nossa proposta. O Capítulo 4 apresenta a metodologia utilizada nesta pesquisa. O Capítulo 5 apresenta a arquitetura e os modelos SPN. O Capítulo 6 apresenta o processo de validação do modelo SPN. O Capítulo 7 apresenta as principais conclusões obtidas acerca dos estudos realizados.

2 Referencial Teórico

Em sistemas de filas é essencial compreender como um estágio do sistema impacta nos estágios seguintes. Cada estágio do sistema pode ser decomposto em filas de prioridades, entretanto o correto balanceamento do sistema evita perda de informação essencial. Assim, a elaboração de modelos ajuda a mitigar problemas de desempenho e utilização de recursos. Este capítulo apresenta conceitos fundamentais para a compreensão do conteúdo proposto nesta dissertação. Inicialmente, será apresentada uma introdução sobre *Internet of Things* (IoT), *Vehicular ad hoc Networks*, Redes de Petri e, posteriormente, conceitos sobre Técnicas de Validação.

2.1 Internet of Things

O conceito de *Internet of Things* surgiu da evolução de diversas áreas como microeletrônica, sensoriamento, comunicação e sistemas distribuídos. Porém, em poucas palavras, IoT nada mais é do que uma extensão da Internet atual, que permite controlar e monitorar a rotina diária em objetos inteligentes, mas com habilidades computacionais e de comunicação, para se conectar à Internet (LEA, 2018). A IoT é a interligação de redes de dispositivos físicos incorporados com eletrônicos, software, sensores e troca de dados (também chamados de "dispositivos conectados" e "dispositivos inteligentes") (BRIEF, 2019), permitindo a criação de sistemas como redes veiculares, que integram veículos, infraestrutura e dispositivos inteligentes para otimizar o trânsito urbano. Sendo assim, a IoT transforma objetos físicos (sensores, dispositivos de comunicação com/sem fio) em dispositivos inteligentes para coletar, comunicar, monitorar e interpretar informações de seus arredores em tempo real, e os interconectam logicamente com objetos virtuais (aplicativos da web, máquinas virtuais) sobre a infraestrutura de Internet existente (PAUL; JEYARAJ, 2019). Os sistemas IoT podem ser inseridos em várias áreas e em cada uma tem um impacto diferente. A indústria e a manufatura são os maiores segmentos no espaço geral da IoT, considerando o número de dispositivos conectados ao valor que esses serviços agregam à indústria (LEA, 2018).

De acordo com (PAUL; JEYARAJ, 2019), a arquitetura IoT consiste em cinco fases: coleta de dados, transmissão de dados, protocolo usado para transportar dados, plataforma para armazenar e processar e, finalmente, a tomada de decisão. A fase de coleta de dados pode ser realizada, por exemplo, com o uso de sensores, atuadores, transceptores e GPS. Na fase de transmissão de dados, diversos tipos de rede podem ser utilizados: redes de longa distância (*Wide Area Networks* - WAN), redes locais (*Local Area Networks* - LAN), e redes de área pessoal (*Personal Area Networks* - PAN). Na fase de transportar dados,

o protocolo usado pode ser IP ou *Leach*. Na fase de armazenamento e processamento, algumas plataformas que podem ser utilizadas são: computação em *cloud*, computação em *fog* e computação em *edge*. Por fim, a fase de tomada de decisão se refere ao contexto de aplicação, que pode ser referente à cuidados de saúde, segurança, conforto climático, entre outros.

O surgimento de vários novos aplicativos de computação e dispositivos inteligentes, tornou-se possível devido à disponibilidade de um grande pool de recursos. O ambiente IoT requer uma infraestrutura para armazenar e processar os dados para a tomada de decisões, no entanto os aplicativos sensíveis a atraso apresentam alto nível de exigência para as infraestruturas computacionais convencionais. Dessa forma, o uso de serviços de computação em *cloud*, *fog* e *edge* tornaram-se partes importantes para aplicativos IoT. A computação em *cloud* tem como principal objetivo fazer melhor uso dos recursos distribuídos e resolver problemas de computação em grande escala. Por exemplo, a computação em *cloud* pode concentrar o poder de milhares de computadores em um problema, permitindo que os pesquisadores façam seu trabalho mais rápido do que nunca (SADIKU; MUSA; MOMOH, 2014). No entanto o paradigma da computação em *cloud* não consegue atender aos requisitos de baixa latência e reconhecimento de local. Nesse contexto, pode-se introduzir a computação em *edge* e *fog*.

2.1.1 Computação em Edge

Com a crescente quantidade de dados gerados na *edge* da rede, a velocidade do transporte de dados está se tornando o gargalo para o paradigma da computação baseada em *cloud* (SHI et al., 2016). Nesse caso, os dados precisam ser processados na *edge* para um tempo de resposta mais curto, processamento mais eficiente e menor uso da infraestrutura de rede. A computação em *edge* se refere às tecnologias que permitem que a computação seja realizada na borda da rede, para que ela aconteça perto das fontes de dados. Dispositivos como RSUs podem atuar como nós de borda, processando dados localmente e reduzindo a dependência da *cloud* para tomadas de decisão em tempo real. Dessa forma, os provedores de conteúdo e desenvolvedores de aplicativos podem usar os sistemas de computação em *edge*, oferecendo aos usuários serviços mais próximos a eles. Os casos de uso da computação em *edge* são serviços de localização, realidade aumentada, análise de vídeo e armazenamento em cache de dados (KHAN et al., 2019). Em (SHI et al., 2016) os autores citam algumas formas de melhorar a qualidade de serviços interativos usando a *edge*, de acordo com o tipo de aplicação:

- Os aplicativos de navegação podem mover os serviços de navegação ou pesquisa para a *edge* de uma área local, nos caso em que apenas alguns blocos de mapa estão envolvidos.

- A filtragem/agregação de conteúdo pode ser feita nos nós de extremidade para reduzir o volume de dados a ser transferido.
- Aplicativos em tempo real, como jogos de entretenimento de auxílio à visão, realidade aumentada e saúde conectada, podem dar respostas rápidas usando nós de extremidade.

Na computação em *edge*, seus dispositivos podem ser qualquer recurso de computação ou rede que resida entre fontes de dados e data centers baseados em *cloud*. Por exemplo, um dispositivo de *edge* pode ser um *smartphone* que se encontra entre os sensores e a *cloud*. Na computação em *edge*, o dispositivo final não apenas consome dados, mas também produz dados. E na extremidade da rede, os dispositivos não apenas solicitam serviços e informações da *cloud*, mas também lidam com tarefas de computação - incluindo processamento, armazenamento, armazenamento em cache e balanceamento de carga - em dados enviados de e para a *cloud*. A *edge* deve ser projetada bem o suficiente para lidar com essas tarefas de forma eficiente, confiável, segura e com privacidade em mente. Portanto, deve oferecer suporte a requisitos como diferenciação, extensibilidade, isolamento e confiabilidade. Segundo (KHAN et al., 2019), os requisitos para utilizar os sistemas de computação em *edge* podem ser classificados em mecanismo de faturamento inteligente, suporte a aplicativos em tempo real, modelo de negócios conjunto para gerenciamento e implantação, gerenciamento de recursos, arquitetura escalonável, recursos de redundância e segurança.

2.1.2 Computação em Fog

Computação em *fog* refere-se a um paradigma de computação distribuída que move o armazenamento e a computação geralmente perto dos nós finais da rede com o objetivo de reduzir a sobrecarga da rede e computar as informações coletadas o mais rápido possível (BELLAVISTA et al., 2019). Semelhante a *cloud*, a *fog* fornece serviços de dados, computação, armazenamento e aplicativos para usuários finais. Em contraste com a *cloud*, a *fog* fornece serviços com resposta mais rápida e maior qualidade. Na computação em *fog*, os serviços podem ser hospedados em dispositivos finais, como decodificadores ou pontos de acesso. A infraestrutura desta computação distribuída permite que os aplicativos sejam executados o mais próximo possível de dados vindos de pessoas, processos e coisas.

A computação em *fog* oferece vantagens importantes para os aplicativos de hoje, como processamento em tempo real, escalonamento rápido e acessível, conteúdo local e *pool* de recursos. Assim, a computação em *fog* atraiu rapidamente muita atenção da indústria e da academia. Naturalmente, ele conecta a IoT com a infraestrutura de computação da internet. Em (CHEN; ZHANG; SHI, 2017) os autores citam alguns aplicativos que exigem computação com *fog*, são eles: veículos conectados, veículos de piloto automático, redes

inteligentes, redes de sensores e atuadores sem fio, casas inteligentes, cidades inteligentes, manufatura conectada, sistemas de óleo e gás conectados e sistemas móveis de saúde.

Os ambientes de computação em *fog* e *cloud* devem cumprir alguns requisitos para operar corretamente e alcançar a satisfação do usuário. Em (BELLAVISTA et al., 2019) tais requisitos são definidos como escalabilidade, interoperabilidade, capacidade de resposta em tempo real, qualidade de dados, segurança, percepção de localização, mobilidade e confiabilidade.

2.2 Vehicular ad hoc Networks

A rápida urbanização expandiu a frota de veículos e impulsionou a pesquisa em *Vehicular ad hoc Networks* (VANETs) (HOTA et al., 2022), tanto no meio acadêmico, quanto na indústria (BOUKHALFA et al., 2021), principalmente porque possuem potencial para aumentar a segurança no trânsito e melhorar a eficiência do tráfego, bem como proporcionar conveniência e conforto aos passageiros e motoristas (KASSIR et al., 2020; BOUKHALFA et al., 2021; TAHER et al., 2024). As VANETs são redes sem fio descentralizadas e auto-organizadas, onde os veículos trocam informações relevantes entre si e com a infraestrutura ao longo da via (MAHMOOD; HORVÁTH, 2020; BALZANO; STRANIERI, 2019; YU et al., 2021; SHAHIN et al., 2023). Em um cenário de tráfego seguro, a informação sobre um acidente de carro é difundida através da VANET e veículos distantes podem planejar rotas alternativas (YU et al., 2021).

2.2.1 Tipos de Comunicação

A comunicação *Vehicle to Vehicle* (V2V), *Vehicle to Infrastructure* (V2I) e hibrida são as principais formas de transmissão de dados em VANETs (CUNHA et al., 2021). Cada modo de comunicação possui características adequadas para diferentes tipos de aplicações. A comunicação V2V permite a troca de dados diretamente entre os veículos (CUNHA et al., 2021), onde os veículos atuam como nós de uma rede (AHMED; MOHAMED; SADEK, 2017a). Os dados podem ser orientados a eventos inesperados, tais como colisões e paradas de emergência, além de mensagens de controle relacionadas ao veículo, como localização, velocidade e distância entre veículos (ISMAIL et al., 2022). A comunicação V2V é eficaz em situações onde muitos veículos estão próximos, mas menos confiável em cenários de baixa densidade de veículos (KASSIR et al., 2020). A comunicação V2I consiste na troca de dados entre os veículos e unidades de infraestrutura fixa (CUNHA et al., 2021). A modalidade V2I atua como intermediária quando a comunicação V2V está indisponível (GAO et al., 2020), devido à distância entre veículos ou outros fatores.

2.2.2 Principais Componentes de uma VANET

Uma arquitetura VANET é composta por três componentes principais: *On-board Unit* (OBU), *Roadside Unit* (RSU) e *Trusted Authority* (TA) (AHMED; MOHAMED; SADEK, 2017b; KHAN et al., 2021). Os veículos participantes de uma VANET são equipados com *On-board Units* (OBUs), que é responsável por realizar a comunicação *Vehicle to Vehicle* (V2V), bem como comunicação *Vehicle to Infrastructure* (V2I) (HOTA et al., 2022; KASSIR et al., 2020; MOUSA; HUSZÁK; SALMAN, 2021). As OBUs são equipamentos de comunicação sem fio, que usam tecnologias de comunicação como *Dedicated Short Range Communication* (DSRC), definido na IEEE 802.11p e IEEE 1609 (HOTA et al., 2022; MOUSA; HUSZÁK; SALMAN, 2021; SHAHIN et al., 2023), bem como *Wireless Access for Vehicular Environment* (WAVE), que permitem comunicação eficiente e proveem alta taxa de dados, links confiáveis e menos latência (HOTA et al., 2022). A OBU utiliza informações coletadas pelos sensores do veículo, tais como posição e velocidade, que são trocadas na comunicação V2V e V2I (MOUSA; HUSZÁK; SALMAN, 2021). As informações trocadas pelos veículos auxiliam na tomada de decisão, seja de motoristas, seja de veículos de condução autônoma.

As *Roadside Units* (RSUs) são dispositivos instalados ao longo das ruas e estradas, que permitem a comunicação entre veículos e servem como meio de acesso à infraestrutura de rede mais ampla (HOTA et al., 2022; AHMED; MOHAMED; SADEK, 2017b). As RSUs se comunicam entre si por meio de conexões com fio rápidas e estáveis (GAO et al., 2020), mas também podem ter um link dedicado com *datacenters* ou Internet (MOUSA; HUSZÁK; SALMAN, 2021). RSUs realizam o processamento em tempo real dos dados detectados antes de enviá-los para a nuvem (SHAHIN et al., 2023). Assim, as RSUs agregam informações recebidas dos veículos e enviam para um servidor que processará e analisará dados offline (SHAHIN et al., 2023). O servidor retransmite uma mensagem para as RSUs instaladas na mesma área geográfica, que por sua vez transmitirão a mensagem para sua área de cobertura (JABALLAH; CONTI; LAL, 2019). Os veículos podem usar tecnologia sem fio *multi-hop* para transmitir dados para veículos adjacentes, mas ao entrarem no alcance das RSUs eles também podem se comunicar no modo sem fio (GAO et al., 2020). RSUs podem ser instaladas em locais estratégicos, como cruzamentos ou perto de estacionamentos (AHMED; MOHAMED; SADEK, 2017b). A distribuição das RSUs deve maximizar a área de cobertura e a utilização (SHAHIN et al., 2023).

O terceiro componente é a *Trusted Authority* (TA), que atua como uma entidade de terceiros confiável (AHMED; MOHAMED; SADEK, 2017b), que gerencia a confiança e segurança das VANETs, sendo a base de todo o ITS (KHAN et al., 2021). A TA registra as RSUs e OBUs, emitindo certificados para as entidades na VANET, além de fornecer parâmetros de criptografia, como pares de chave pública e privada (AHMED; MOHAMED; SADEK, 2017b). A TA também verifica os elementos da rede através da RSU e identifica

OBUs que enviam pacotes maliciosos (KHAN et al., 2021). As VANETs podem ter várias *Regional Trusted Authority* (RTA), onde cada uma é responsável por uma região específica (AHMED; MOHAMED; SADEK, 2017b). A TA fica no centro da cidade e é gerenciado por governos (KHAN et al., 2021)

2.3 Redes de Petri Estocásticas

Rede de Petri (PN) é uma ferramenta de modelagem gráfica e matemática, que permite descrever e estudar sistemas de processamento de informações caracterizados como sendo concorrentes, assíncronos, distribuídos e paralelos (PETERSON, 1977; LIMA; CALLOU; ANDRADE, 2021). Como ferramenta gráfica, PNs podem ser usadas como auxílio à comunicação visual, semelhante aos gráficos de fluxo, diagramas de blocos e diagramas de redes. Modelos diversas variações de PNs foram propostos visando desenvolver uma ferramenta que permitisse integrar descrição formal, prova de correção e avaliação de desempenho. Dentre as variações de PNs, destacam-se as Redes de Petri Estocásticas Generalizadas (GSPNs), que incorporam elementos estocásticos para representar sistemas de forma ainda mais precisa (MOLLOY, 1982; GERMAN, 2000a; MARSAN et al., 1994; MACIEL, 2023b; MACIEL, 2023a). Entretanto, por simplicidade, chamaremos essas redes apenas de SPNs. Como ferramenta matemática, é possível configurar equações de estado, equações algébricas e outros modelos matemáticos que regem o comportamento de sistemas. Desde o trabalho seminal de Petri, muitas representações e extensões foram propostas, permitindo descrições mais concisas e a representação de características de sistemas não observadas nos modelos iniciais (MURATA, 1989).

Uma SPN é representada graficamente por arcos, que podem ter pesos, transições temporizadas, transições imediatas, lugares e *tokens* (LIMA; CALLOU; ANDRADE, 2021). A Figura 1 exibe os componentes usados para modelar uma SPN. Os lugares são representados por círculos, que significam possíveis estados de um determinado sistema. Os lugares podem conter *tokens* (pontos pretos), que representam o estado da SPN (PETERSON, 1977). As transições são representadas como retângulos e correspondem à execução de uma ação. As transições temporizadas possuem um atraso associado, sendo a principal característica de uma SPN (MURATA, 1989). As transições determinísticas têm um tempo de disparo fixo, enquanto as transições imediatas não têm tempo associado. As transições imediatas possuem tempo de disparo equivalente a zero, assim têm precedência em relação às transições temporizadas e determinísticas. Arcos conectam lugares a transições e vice-versa. Um arco inibidor é representado por um pequeno círculo branco em uma borda, em vez de uma seta. Arcos inibidores são usados para desativar as transições se houver *tokens* presentes em um lugar.

A Figura 2 mostra uma SPN que representa um procedimento simplificado de

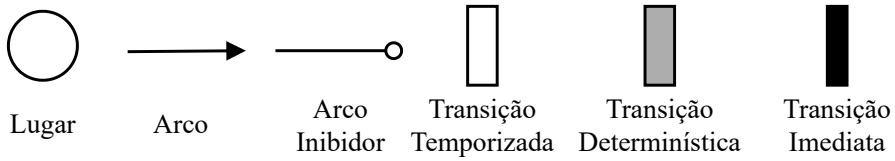


Figura 1 – Componentes principais de uma SPN.

embarque. O *token* no lugar *Fila de check-in* indica que há um passageiro pronto para iniciar o procedimento. Neste momento a transição imediata fica habilitada para disparar e remove o *token* para o lugar *Passageiro em check-in*. O tempo de atendimento varia de passageiro para passageiro, pois depende de questões como assentos disponíveis, quantidade de bagagem para despachar, documentação correta, dentre outros fatores. Quando decorrer o tempo necessário para atendimento, a transição *Finalizar check-in* é habilitada e dispara. Assim o *token* migra para o *Check-in realizado*. Como não existem mais *tokens* nos lugares anteriores, a SPN finaliza.

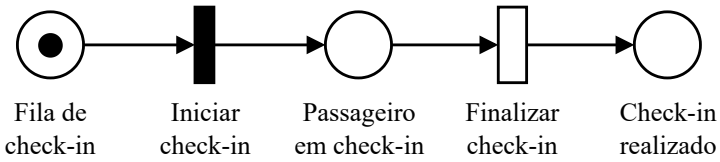


Figura 2 – Exemplo simples de SPN.

A Figura 3 mostra uma SPN com capacidade de processamento paralelo. Nesta versão do procedimento de check-in, a sala conta com quatro guichês de atendimento, onde o lugar *Guichês disponíveis* mostra quantos atendentes podem chamar um passageiro para iniciar o procedimento. Na figura temos três passageiros (*tokens*) em atendimento, três passageiros aguardando e um guichê com atendente disponível. Neste caso a transição imediata *Iniciar check-in* fica habilitada para disparar. A transição realiza uma junção, onde remove um token que representa um atendente, juntamente com o token que representa um passageiro, migrando para o lugar *Passageiro em check-in*. Quando *Finalizar check-in* dispara, ela remove um *token* e divide, onde o passageiro agora está com *Check-in realizado* e um atendente fica com o guichê disponível para novo atendimento. Considerando que neste terminal o embarque é realizado com intervalo de tempo fixo, a transição determinística *Embarcar* representa o momento de embarque. Todavia, o arco que chega nessa transição possui peso 10, ou seja, além do tempo fixo é necessário que existam 10 passageiros em *Check-in realizado*, para que a transição *Embarcar*.

A SPN da Figura 4 detalha o procedimento de *check-in*. Neste modelo, a sala de embarque possui vagas limitadas para o *check-in*. Os passageiros que se dirigem ao terminal chegam no tempo determinado pela transição exponencial *Tempo de chegada*. Quando essa transição dispara, um *token* é depositado em *Passageiros chegando* representando os passageiros que querem entrar na sala de embarque. Se houver *token* em *Passageiros chegando* e um *token* em *Vagas na sala de embarque*, então a transição *Acessar a fila*

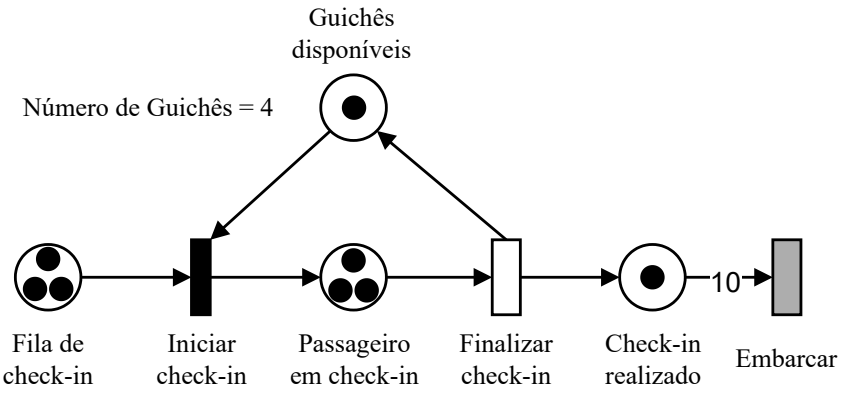


Figura 3 – Exemplo de SPN com fila de processamento.

é habilitada para disparo. Assim, a transição junta os *tokens* e os deposita na *Fila de check-in*. Quando houver atendente disponível, o passageiro inicia o *check-in* e uma vaga na fila é liberada.

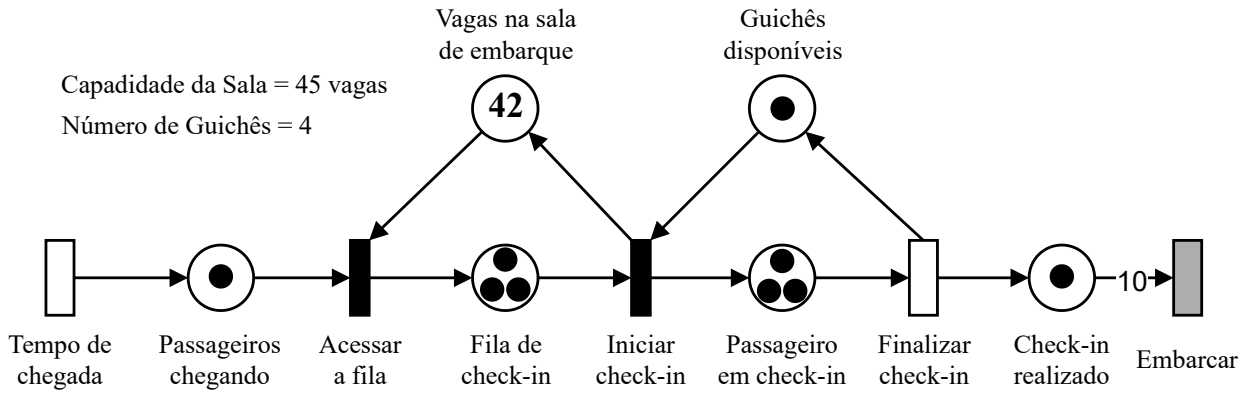


Figura 4 – Exemplo de SPN com fila de chegada e fila de processamento.

A Figura 2 apresenta um modelo SPN com estado absorvente, ou seja, o *token* segue o fluxo determinado e quando todos os *tokens* chegam ao fim do modelo, nenhuma transição pode ser habilitada novamente. A Figura 3 não possui estado absorvente, entretanto, ela finaliza porque os *tokens* disponíveis transitam até o último estado, mas os novos *tokens* não são gerados após a transição determinística disparar. A Figura 4 também não possui estado absorvente, entretanto, neste caso a movimentação dos *tokens* nunca termina, tende ao infinito, pois novos *tokens* sempre são gerados e a capacidade é sempre reposta.

O comportamento da SPNs é definido em termos de um fluxo de *tokens*. Os *tokens* são criados e destruídos conforme os disparos de transição (GERMAN, 2000b). As transições imediatas representam atividades instantâneas e têm maior prioridade de disparo do que transições temporizadas. Tais transições também podem conter uma condição de guarda e o usuário pode especificar uma prioridade de disparo diferente entre outras transições imediatas. Há também funções de guarda em SPNs. As funções de guarda são expressões booleanas que controlam o disparo de uma transição, declarando alguma condição em relação à marcação da rede. Se a função de guarda de uma transição produz

um valor verdadeiro, ela é habilitada; caso contrário, a transição é desabilitada (MARSAN et al., 1998).

2.4 Técnicas de Validação de Modelos

A validação de modelos é essencial para garantir que as suposições usadas no desenvolvimento sejam razoáveis e que o modelo produza resultados próximos aos observados em sistemas reais. As técnicas de validação variam conforme as suposições e os sistemas modelados. Ao contrário das técnicas de verificação, que são geralmente aplicáveis, as técnicas de validação podem não se aplicar a outras simulações (SANTNER, 2003; JAIN, 1990).

A validação do modelo envolve a verificação de três aspectos principais: premissas, valores e distribuições dos parâmetros de entrada, bem como os valores e conclusões de saída. Esses aspectos podem ser testados comparando-os com a intuição de especialistas, medições de sistemas reais e resultados teóricos (JAIN, 1990).

2.4.1 Medições de Sistemas Reais

A comparação com sistemas reais é a forma mais confiável e preferida para validar um modelo de simulação. Na prática, no entanto, isso muitas vezes não é viável devido à inexistência de sistemas reais ou ao alto custo das medições necessárias. Mesmo assim, uma ou duas medições podem aumentar significativamente a validade de uma simulação. É fundamental que as suposições, valores de entrada e saída, cargas de trabalho, configurações e comportamento do sistema sejam comparados com aqueles observados no mundo real. Técnicas estatísticas, como intervalo de confiança para a média, comparação de duas alternativas e determinação do tamanho da amostra, podem ser empregadas para verificar se dois sistemas são idênticos, possibilitando a comparação entre o modelo de simulação e os dados reais coletados. Por exemplo, as distribuições de entrada podem ser determinadas por técnicas de ajuste de distribuição estatística ou usando testes estatísticos, como o teste qui-quadrado (*chi-square test*) (JAIN, 1990).

Em simulações baseadas em rastreamento, é essencial utilizar múltiplos rastreamentos obtidos em diferentes ambientes para validar o modelo. Esse processo garante que o modelo não apenas reflita condições específicas, mas também seja robusto o suficiente para generalizar a partir de diferentes cenários. Utilizar essas técnicas avançadas permite uma validação mais rigorosa e confiável, proporcionando maior confiança nos resultados da simulação e na sua aplicabilidade em situações reais (JAIN, 1990; SANTNER, 2003).

2.4.2 Resultados Teóricos

Em alguns casos, é possível modelar analiticamente o sistema sob suposições simplificadoras. Também pode ser possível determinar analiticamente quais deveriam ser as distribuições de entrada. Nesses casos, a similaridade dos resultados teóricos e de simulação é usada para validar o modelo de simulação. Da mesma forma, ao desenvolver modelos analíticos, os pesquisadores verificam que o comportamento do sistema real é difícil de modelar com exatidão (JAIN, 1990). Nesses casos, eles desenvolvem uma técnica aproximada e a validam comparando os resultados com aqueles obtidos de uma simulação.

A validação por comparação de resultados teóricos e de simulação deve ser usada com cuidado, pois ambos podem ser inválidos no sentido de que ambos podem não representar o comportamento de um sistema real. No entanto, é uma técnica útil, desde que alguns pontos (de teoria ou simulação) tenham sido validados usando a intuição de especialistas ou medições de sistemas reais (JAIN, 1990). A modelagem analítica permite validação para casos mais complexos. Por exemplo, um modelo que simula multiprocessadores, pode ser validado usando alguns sistemas reais com um ou dois processadores e, em seguida, a teoria pode ser usada para validar um número maior de processadores (SANTNER, 2003).

A validação de modelos é essencial para garantir a confiabilidade e precisão das simulações. Ela envolve verificar se as suposições do modelo são realistas, bem como se os resultados se aproximam dos observados em sistemas reais. Existem três técnicas principais de validação: análise intuitiva de especialistas, medições em sistemas reais e comparação com resultados teóricos. A escolha da técnica depende das características do sistema modelado e dos recursos disponíveis para validação (JAIN, 1990).

2.4.3 Técnicas Estatísticas para Validação

As técnicas estatísticas para a análise de dados de simulação e validação de modelos serão apresentadas nesta subseção. Inicialmente, exploramos os intervalos de confiança para a média, que estimam a média populacional com base em amostras e fornecem uma faixa de valores esperada com certo nível de confiança. Essa ferramenta é essencial para quantificar a precisão das estimativas (JAIN, 1990). Além disso, detalhamos métodos de comparação entre duas alternativas, considerando observações pareadas ou não pareadas. Essas técnicas permitem avaliar diferenças significativas entre sistemas. Também discutimos a importância do tamanho adequado da amostra para estimativas confiáveis (JAIN, 1990).

2.4.3.1 Intervalo de Confiança para a Média

Uma das ferramentas fundamentais e, amplamente utilizadas em estatística, é a técnica do intervalo de confiança para a média. Essa técnica nos permite estimar a média de uma população com base em uma amostra e fornecer uma faixa de valores dentro da

qual esperamos que a média verdadeira da população se situe com um determinado nível de confiança, geralmente 95% ou 99%. Para calcular um intervalo de confiança, precisamos primeiro determinar a média amostral ($\bar{x}$), que é a soma dos valores observados na amostra dividida pelo número de observações n. A média amostral serve como um ponto central para o intervalo de confiança. Em seguida, calculamos o erro padrão da média (SE), que é o desvio padrão da amostra (σ) dividido pela raiz quadrada do número de observações ($\sqrt{n}$). O erro padrão fornece uma medida da variabilidade ou dispersão dos valores da amostra em torno da média (JAIN, 1990).

Uma vez que temos a média amostral e o erro padrão, podemos calcular o intervalo de confiança usando a fórmula $(\bar{x}) \pm z \times SE$, onde z é o valor crítico da distribuição normal para o nível de confiança desejado. Por exemplo, para um nível de confiança de 95%, z é aproximadamente 1,96. Isso significa que esperamos que 95% dos intervalos de confiança calculados a partir de amostras aleatórias dessa população contenham a média verdadeira. O intervalo de confiança é, portanto, uma faixa de valores simétrica em torno da média amostral. O intervalo de confiança se expande ou contrai dependendo da variabilidade dos dados (refletida pelo erro padrão), bem como do tamanho da amostra. Esta técnica é extremamente útil porque, além de fornecer uma estimativa pontual (a média amostral), também nos dá uma medida de incerteza associada a essa estimativa (JAIN, 1990).

Por exemplo, considere que estamos analisando o tempo de resposta de um sistema de TI e coletamos uma amostra de 30 tempos de resposta. A análise encontrou uma média amostral de 20 segundos e um desvio padrão de 5 segundos. O erro padrão seria calculado como $\frac{5}{\sqrt{30}} \approx 0.91$. Para um intervalo de confiança de 95%, multiplicamos o erro padrão por 1,96, resultando em aproximadamente 1,79. Portanto, o intervalo de confiança para a média de tempo de resposta seria de $20 \pm 1{,}79$ segundos. Este intervalo sugere que, com 95% de confiança, a média verdadeira do tempo de resposta do sistema está na faixa que vai de 18,21 segundos a 21,79 segundos. Utilizar intervalos de confiança é crucial em análises de desempenho e experimentos científicos, porque permite aos analistas e pesquisadores fazer inferências sobre a população com base em amostras. Assim, consideram a variabilidade e proporcionam uma visão mais robusta do que simplesmente reportar uma média pontual.

2.4.3.2 Comparação de Duas Alternativas

Comparar duas alternativas é uma tarefa comum e crítica na análise de desempenho e na tomada de decisões. As técnicas estatísticas para esta comparação podem ser aplicadas a observações pareadas ou não pareadas, dependendo da natureza dos dados. Observações pareadas são aquelas em que cada par de observações consiste em um valor de cada sistema sob a mesma condição ou tempo, permitindo um controle mais rigoroso das variáveis externas. A análise das diferenças entre cada par de observações permite avaliar se uma das alternativas tem um desempenho consistentemente melhor. A diferença entre cada

par é calculada e, a média dessas diferenças juntamente com seu intervalo de confiança, é utilizada para determinar se há uma diferença significativa entre os sistemas. Por exemplo, se tivermos os tempos de resposta de dois sistemas A e B, sob seis *workloads* diferentes, e calcularmos as diferenças para cada *workload*, poderemos construir um intervalo de confiança para a média dessas diferenças. Se o intervalo de confiança não incluir zero, isso indica que há uma diferença significativa entre os sistemas (JAIN, 1990).

Para observações não pareadas, onde as observações dos dois sistemas são independentes, a abordagem é um pouco diferente. Neste caso, calculamos a diferença entre as médias dos dois grupos e construímos um intervalo de confiança para essa diferença. Isso envolve considerar as variâncias de ambos os grupos, pois a dispersão dos dados pode influenciar a interpretação dos resultados. A fórmula para o intervalo de confiança da diferença entre duas médias independentes é mais complexa e envolve o cálculo do erro padrão combinado das duas amostras. Por exemplo, suponha que tenhamos dois sistemas de TI, A e B, e coletamos amostras independentes de tempos de resposta para cada sistema. Se a média e o desvio padrão dos tempos de resposta para o sistema A forem 20 segundos e 5 segundos, respectivamente, com 30 observações, e para o sistema B forem 22 segundos e 4 segundos, com 35 observações, podemos calcular o erro padrão combinado e o intervalo de confiança para a diferença entre as duas médias (JAIN, 1990).

A comparação de duas alternativas é essencial para validar melhorias ou mudanças em sistemas. Um exemplo prático disso seria avaliar duas versões de um algoritmo de compressão de dados. Suponha que as diferenças nos tempos de compressão entre a versão antiga e a nova para um conjunto de arquivos sejam medidas. Se a média das diferenças de tempo for significativamente diferente de zero, com um intervalo de confiança que não inclua zero, podemos concluir que uma versão do algoritmo tem um desempenho melhor que a outra. Essa análise pode ser complementada por gráficos de *boxplot* ou histogramas das diferenças, que visualmente reforçam a interpretação dos dados estatísticos. Assim, comparar duas alternativas com técnicas estatísticas robustas nos permite tomar decisões informadas sobre qual sistema ou método adotar, baseando-se em evidências quantitativas e não apenas em intuições ou suposições (JAIN, 1990).

2.4.3.3 Determinação do Tamanho da Amostra

A determinação do tamanho adequado da amostra é um aspecto crítico na condução de estudos estatísticos e experimentos, garantindo que os resultados sejam confiáveis e precisos. O tamanho da amostra necessário para obter uma estimativa precisa depende da variabilidade dos dados, do nível de confiança desejado e da precisão requerida. Para estimar uma média, o tamanho da amostra pode ser calculado para garantir que o intervalo de confiança tenha uma largura desejada. A fórmula para calcular o tamanho da amostra para a média $n = \left(\frac{z\sigma}{r}\right)^2$ é, onde z é o valor crítico da distribuição normal para o nível de

confiança desejado, é o desvio padrão da amostra e r é a precisão desejada (ou a metade da largura do intervalo de confiança). Esta fórmula permite determinar quantas observações são necessárias para estimar a média da população com a precisão desejada (JAIN, 1990).

Além de médias, a determinação do tamanho da amostra é crucial ao trabalhar com proporções. A fórmula para calcular o tamanho da amostra para proporções é $\frac{z^2 p(1-p)}{r^2}$, onde p é a proporção estimada da população e r é a precisão desejada. Por exemplo, se estamos interessados em estimar a proporção de páginas ilegíveis de uma impressora e estimamos que essa proporção seja 0,001, desejando uma precisão de 0,000001 a 95% de confiança, precisamos calcular o tamanho da amostra necessário para garantir essa precisão. Utilizando a fórmula apropriada, podemos determinar o número de páginas que devem ser examinadas para obter uma estimativa confiável da proporção de páginas ilegíveis.

A determinação do tamanho da amostra também é essencial ao comparar duas alternativas, como sistemas ou algoritmos. Neste caso, o tamanho da amostra deve ser suficiente para garantir que os intervalos de confiança das médias dos dois sistemas não se sobreponham, indicando uma diferença significativa. Os cálculos para o tamanho da amostra neste contexto envolvem as variâncias e médias das duas amostras. Fórmulas mais complexas são usadas, como aquelas que consideram o erro padrão combinado das duas amostras. Garantir um tamanho de amostra adequado é fundamental para evitar conclusões incorretas devido a amostras insuficientes, o que poderia levar a uma interpretação errônea dos resultados e a decisões inadequadas (JAIN, 1990).

Para ilustrar, suponha que estamos comparando a eficiência energética de dois tipos de motores. Se queremos detectar uma diferença média de 5 unidades de eficiência com 95% de confiança e sabemos que o desvio padrão da eficiência é 10, podemos usar a fórmula para determinar o tamanho necessário da amostra. Considerando a precisão desejada e o nível de confiança, o cálculo nos forneceria o número de medições necessárias para fazer uma comparação estatisticamente significativa entre os dois motores. Este processo de determinação do tamanho da amostra é crucial em muitos campos, desde a engenharia até a medicina, onde a coleta de dados pode ser custosa e demorada. Portanto, a técnica não apenas assegura a precisão e a validade dos resultados, mas também otimiza os recursos utilizados na condução dos estudos (JAIN, 1990).

3 Trabalhos Relacionados

Este capítulo apresenta os trabalhos relacionados com abordagens ou contextos a este estudo. A seleção de trabalhos considerou critérios como Tipo de Modelo, Métricas, uso de RSUs, Múltiplas Filas de Prioridade, uso de Escalonamento e Foco. A Tabela 1 agrupa as contribuições dos trabalhos com maior relevância ao tema desta dissertação.

Tabela 1 – Trabalhos relacionados.

Trabalho	**Tipo de Modelo**	**Métricas**	**Usa RSUs**	**Múltiplas Filas de Prioridade**	**Usa Escalonamento?**	**Foco**
(HOTA et al., 2022)	FRIIS, Two-Ray Ground, Log-Distance, Nakagami	Throughput, taxa de entrega de pacotes, atraso end-to-end, goodput médio, overhead	✓	✗	✗	Protocolos de Roteamento
(KASSIR et al., 2020)	Poisson	Throughput, conectividade, taxa compartilhada por usuário, redundância de serviço	✓	✗	✓	Conectividade
(MAHMOOD; HORVÁTH, 2020)	Markov Chain	Velocidade de propagação de mensagens, distribuição transitória da distância	✓	✗	✗	Infraestrutura e Gerenciamento
(WANG et al., 2023)	Confiança distribuída	Precisão, recall, taxa de entrega de mensagens	✓	✗	✗	Segurança e Veículos Maliciosos
(MOUSA; HUSZÁK; SALMAN, 2021)	Matemático	Taxa de conectividade	✓	✗	✗	Conectividade

Continua na próxima página

Tabela 1 – Continuação da página anterior

Trabalho	Tipo de Modelo	Métricas	Usa RSUs	Múltiplas Filas de Prioridade	Usa Escalonamento?	Foco
(SHAHIN et al., 2023)	Matemático	Cobertura de estrada, demanda computacional satisfeita	✓	✗	✗	Conectividade
(ZHANG et al., 2024)	Matemático	Taxa de transferência média do sistema, probabilidade de sucesso na transmissão	✓	✗	✓	Conectividade
(GHOSH; MISRA; CHAKRABORTY, 2023)	IIA-ORD	Cobertura média, razão de tempo de cobertura, taxa de entrega de pacotes, atraso fim-a-fim	✓	✗	✗	Infraestrutura e Gerenciamento
(SHUHAIMI et al., 2021)	Free Space, Two-Ray Ground, Nakagami	Throughput, perda de pacotes, atraso end-to-end	✗	✗	✗	Conectividade
(BOUKHALFA et al., 2021)	Matemático	Latência, probabilidade de colisão, taxa de sucesso de transmissão	✗	✓	✓	Protocolos de Roteamento
(NIDHI; LOBIYAL, 2021)	Car-Following, Two-Ray Ground	Packet Delivery Ratio, Packet Loss, Routing Overhead, End-to-End Delay	✓	✗	✗	Infraestrutura e Gerenciamento
(RODRIGUES et al., 2021b)	SPN	MRT, probabilidade de descartes, número de descartes, utilização dos RSUs	✓	✗	✗	Infraestrutura e Gerenciamento
(TAHER et al., 2024)	SDN	Packet Delivery Ratio, Delivery Delay Time, Throughput	✓	✗	✗	Protocolos de Roteamento

Continua na próxima página

Tabela 1 – Continuação da página anterior

Trabalho	Tipo de Modelo	Métricas	Usa RSUs	Múltiplas Filas de Prioridade	Usa Escalonamento?	Foco
(ULLAH et al., 2023)	EMR-ICN	Taxa de entrega de mensagens, latência, número médio de saltos	✓	✗	✗	Protocolos de Roteamento
(TAN; CHUNG, 2021)	Matemático	Consumo de armazenamento, tempo durante o processo de autenticação das RSUs	✓	✗	✗	Segurança e Veículos Maliciosos
(ATWA; FLOCCHINI; NAYAK, 2021)	RTEAM	Validação de eventos, detecção de eventos, confiabilidade, estimativa de risco	✓	✗	✗	Segurança e Veículos Maliciosos
(GAO et al., 2020)	Matemático	Taxa de entrega de pacotes, atraso de tempo, saltos sem fio	✓	✗	✗	Protocolos de Roteamento
Este trabalho	SPN	MRT, probabilidade de descartes, utilização	✓	✓	✓	Infraestrutura e Gerenciamento

3.1 Tipo de Modelo

Na modelagem de redes veiculares e RSUs, diferentes tipos de modelos são usados para representar e simular o comportamento do sistema em diversos cenários. A escolha do modelo adequado é fundamental para obter resultados precisos e garantir que as variáveis de interesse sejam bem representadas. A literatura sobre RSUs e VANETs apresenta uma ampla variedade de modelos, cada um com características e aplicações específicas, voltados para a simulação de propagação de sinal, confiabilidade, desempenho de rede e outros aspectos críticos. Entre os modelos de propagação, destacam-se o *Free Space Propagation Model* (FRIIS), o *Two-Ray Ground Propagation Model* (TR), o *Log-Distance Propagation Model* (LD) e o *Nakagami Fading Model* (NK) (HOTA et al., 2022). Esses modelos são amplamente utilizados para analisar a propagação de sinais de rádio em ambientes urbanos e rodoviários, simulando com precisão os efeitos de interferências e obstáculos no caminho de transmissão. Já em cenários que envolvem a movimentação de veículos, o modelo

Car-Following é utilizado para estudar o comportamento do tráfego, enquanto o modelo de Poisson é frequentemente aplicado para descrever a chegada de veículos e eventos em uma rede de maneira probabilística (KASSIR et al., 2020).

Os modelos matemáticos são uma categoria robusta dentro da literatura, servindo para representar uma grande variedade de fenômenos com equações e fórmulas que descrevem tanto a dinâmica de redes quanto a confiabilidade e a eficiência dos sistemas. Modelos como o de *Markov Chain* são fundamentais para a análise de estados de sistemas estocásticos (MAHMOOD; HORVÁTH, 2020; GAO et al., 2020), enquanto modelos de confiança distribuída avaliam a confiabilidade de redes com base em interações entre os nós (WANG et al., 2023), assim como também é abordada em sistemas como o *Risk-Based Trust Evaluation Advanced* (RTEAM), voltado para avaliar o risco e a confiança em redes (ATWA; FLOCCHINI; NAYAK, 2021). Adicionalmente, modelos como o *Intersection Influence Analysis System for Optimal RSU Deployment* (IIA-ORD) são específicos para a otimização do posicionamento de RSUs, garantindo que o sistema funcione com alta eficiência em termos de cobertura de estrada e conectividade entre veículos (GHOSH; MISRA; CHAKRABORTY, 2023). Já as Redes Definidas por Software (SDN) e os modelos Roteamento de Mensagens de Emergência em Redes Intermitentes (EMR-ICN) também se destacam como abordagens inovadoras para redes veiculares, proporcionando soluções escaláveis e eficientes para a gestão de tráfego e comunicação (ULLAH et al., 2023).

Por fim, o modelo SPN, que é o foco deste trabalho, oferece uma poderosa ferramenta para a modelagem de sistemas complexos. A SPN permite a simulação detalhada de processos estocásticos em redes veiculares, como a alocação de recursos e o comportamento de resposta do sistema. A escolha deste modelo se baseia na sua capacidade de capturar a incerteza e a variabilidade inerentes a esses sistemas, proporcionando resultados confiáveis para a análise de desempenho em redes veiculares.

3.2 Métricas

As métricas desempenham um papel crucial na análise e avaliação de sistemas de redes, como as RSUs em VANETs. Na literatura, encontramos uma ampla gama de métricas que ajudam a capturar diferentes dimensões do desempenho. Métricas como atraso fim a fim (BOUKHALFA et al., 2021) e taxa de entrega de pacotes (NIDHI; LOBIYAL, 2021; ULLAH et al., 2023) são amplamente utilizadas para medir a eficiência da transmissão de dados e a conectividade entre os veículos e as unidades de infraestrutura, sendo essenciais para avaliar a qualidade de serviço nessas redes. Além disso, aspectos como a latência e a utilização de recursos permitem uma visão mais abrangente da capacidade de resposta do sistema sob condições variáveis de carga (BOUKHALFA et al., 2021).

Em muitos estudos, também há um foco em métricas voltadas para a avaliação

do desempenho de algoritmos e protocolos, especialmente no que diz respeito ao uso eficiente de recursos e à minimização de atrasos. Isso se reflete em análises que investigam a resiliência e a escalabilidade dos sistemas, fundamentais para garantir que o desempenho seja consistente em cenários reais, com variações de demanda e condições adversas. Assim como em (RODRIGUES et al., 2021b), no nosso trabalho, as métricas utilizadas incluem o MRT, a probabilidade de descartes e a utilização. Cada uma delas é fundamental para fornecer uma visão detalhada do desempenho do sistema. Juntas, essas métricas permitem uma análise abrangente e detalhada, buscando equilíbrio entre custo e capacidade, essencial para garantir que o sistema funcione de maneira eficiente, confiável e dentro dos parâmetros de desempenho esperados.

3.3 Uso de RSUs

O uso de RSUs é uma abordagem amplamente adotada em redes veiculares para garantir melhor cobertura de comunicação, maior confiabilidade e eficiência na transmissão de dados entre veículos e a infraestrutura (MOUSA; HUSZÁK; SALMAN, 2021; GHOSH; MISRA; CHAKRABORTY, 2023). Em cenários onde RSUs são implementadas, é possível estabelecer uma rede mais robusta, permitindo que os veículos mantenham conectividade contínua ao longo de grandes distâncias e em áreas com alta densidade de tráfego. Estudos que utilizam RSUs demonstram frequentemente ganhos significativos em termos de latência e qualidade de serviço, especialmente em redes urbanas e rodoviárias com alta demanda de comunicação (MAHMOOD; HORVÁTH, 2020; BOUKHALFA et al., 2021; NIDHI; LOBIYAL, 2021). No nosso trabalho, a utilização de RSUs também é um elemento chave, permitindo uma análise mais detalhada sobre o impacto das unidades no desempenho do sistema.

3.4 Múltiplas Filas de Prioridade

O conceito de múltiplas filas de prioridade é amplamente utilizado em sistemas de rede para gerenciar diferentes tipos de requisições com base em sua criticidade. Esse mecanismo permite que mensagens mais importantes, como alertas de emergência, sejam processadas antes de requisições de menor prioridade, garantindo maior eficiência no processamento. Em redes veiculares, onde a comunicação em tempo real é crucial, as filas de prioridade ajudam a organizar e otimizar o tráfego de dados, assegurando que mensagens críticas sejam entregues com maior prioridade, enquanto requisições de menor importância são processadas de maneira adequada sem afetar o desempenho geral do sistema (BOUKHALFA et al., 2021). No contexto específico deste trabalho, são estabelecidas três classes de requisições. Cada classe é processada em uma fila distinta, e as disciplinas de fila podem variar entre *First Come, First Served* (FCFS), *Last Come, First*

Served (LCFS) e *Service In Random Order* (SIRO). O sistema funciona de maneira não preemptiva, o que significa que uma mensagem em processamento não é interrompida por outra de maior prioridade.

3.5 Usa Escalonamento entre filas de prioridade

O uso de escalonamento em sistemas de redes é uma prática que visa otimizar o gerenciamento de recursos e o processamento de requisições ao longo do tempo. Um agendamento eficiente permite priorizar tarefas e distribuir a carga de trabalho de maneira equilibrada, garantindo que as operações mais críticas sejam realizadas dentro dos prazos exigidos e que os recursos de rede sejam utilizados de forma eficaz (BOUKHALFA et al., 2021). Em redes veiculares, o escalonamento pode ser aplicado para coordenar a transmissão de mensagens, evitar congestionamentos e minimizar a latência, especialmente em cenários onde há uma abundância de dados a serem processados em tempo real. No nosso trabalho, o uso de escalonamento é fundamental para garantir que as diferentes classes de requisições sejam processadas de acordo com sua prioridade. O agendamento, combinado com o sistema de filas de prioridade, organiza o processamento de tarefas de maneira eficiente, evitando sobrecargas e maximizando o desempenho da rede.

3.6 Foco

No estudo sobre redes veiculares, o foco de cada trabalho está em um aspecto específico do desempenho e da segurança em VANETs e RSUs. Para organizar as abordagens da literatura, os trabalhos foram agrupados em categorias temáticas com base em seus principais objetivos.

Protocolos de Roteamento – Os trabalhos focados em protocolos de roteamento analisam como otimizar a disseminação de pacotes em VANETs e melhorar a eficiência da comunicação entre veículos e infraestruturas. O trabalho (HOTA et al., 2022) apresenta uma análise comparativa de protocolos de roteamento e modelos de propagação para otimizar a disseminação de pacotes em VANETs, utilizando cenários realistas simulados. Já o (BOUKHALFA et al., 2021) propõe um protocolo MAC TDMA baseado em sinalização ativa (AS-DTMAC), que visa reduzir a latência e a probabilidade de colisão em redes veiculares, garantindo uma comunicação mais eficiente. O (TAHER et al., 2024) avalia o desempenho do uso de SDN em comparação com protocolos de roteamento ad hoc tradicionais, demonstrando melhorias na Taxa de Entrega de Pacotes (PDR), tempo de entrega e *throughput*. O (ULLAH et al., 2023) propõe um esquema de seleção de relé confiável para o roteamento de mensagens de emergência, utilizando comunicação híbrida V2V e V2I, além de predição de posição e métricas de mobilidade para garantir a estabilidade da rota. Por fim, o (GAO et al., 2020) apresenta um esquema de roteamento híbrido que

combina transmissões V2V e V2R, utilizando RSUs para aumentar a confiabilidade e o desempenho da rede.

Conectividade – Os estudos com foco em conectividade investigam maneiras de melhorar a conectividade entre veículos e infraestrutura, otimizando a comunicação em redes veiculares. O (RODRIGUES et al., 2021b) propõe um modelo para avaliar o desempenho de redes veiculares que utilizam V2V e V2I, destacando os benefícios do *multihoming* em rodovias *multilane* para melhorar a conectividade e a estabilidade da taxa de dados. O (MOUSA; HUSZÁK; SALMAN, 2021) propõe um modelo para determinar as melhores posições para implantar RSUs em rodovias, visando maximizar a conectividade da rede. O (SHAHIN et al., 2023) sugere uma estratégia baseada em *fog computing* para otimizar a configuração e localização de RSUs em VANETs, utilizando técnicas de Teorias de Módulos de Satisfatibilidade (SMT) para minimizar custos e maximizar a qualidade de serviço. O (ZHANG et al., 2024) apresenta uma abordagem de otimização para técnicas de MAC cooperativo em VANETs equipadas com múltiplas RSUs, propondo um esquema distribuído baseado no protocolo CSMA/CA para maximizar a taxa de transferência média. Por fim, o (SHUHAIMI et al., 2021) faz uma análise comparativa do desempenho de diferentes modelos de propagação de rádio em VANETs, avaliando o *throughput*, perda de pacotes e atraso fim-a-fim através de simulações.

Segurança e Veículos Maliciosos – Os trabalhos com foco em segurança e veículos maliciosos discutem como lidar com ameaças e proteger a comunicação nas redes veiculares. O (WANG et al., 2023) apresenta um modelo de confiança distribuída que inclui um mecanismo de apelação para proteger veículos benignos de punições injustas por RSUs comprometidas, melhorando a precisão e confiabilidade na detecção de veículos maliciosos. O (TAN; CHUNG, 2021) propõe um design de associação de grupo de Veículos Aéreos Não Tripulados (UAVs) para disseminação de mensagens V2V em ambientes remotos, aprimorando a conectividade e segurança em VANETs. Por fim, o (ATWA; FLOCCHINI; NAYAK, 2021) apresenta o modelo RTEAM, que utiliza confiança baseada em risco para tomar decisões sobre a validade de eventos em VANETs, considerando a autenticidade do remetente e o risco associado a cada ação.

Infraestrutura e Gerenciamento – Trabalhos com foco em infraestrutura e gerenciamento tratam de soluções para otimizar a implantação e operação de RSUs e o gerenciamento de redes veiculares. O (MAHMOOD; HORVÁTH, 2020) apresenta um modelo para caracterizar o processo de propagação de mensagens em VANETs, determinando a velocidade de propagação e a distribuição transitória da distância onde a mensagem está disponível em uma infraestrutura. O (GHOSH; MISRA; CHAKRABORTY, 2023) propõe um sistema para otimizar a implantação de RSUs em interseções influentes, utilizando um modelo baseado em K-shell modificado e Técnica para a Ordem de Preferência por Similaridade à Solução Ideal (TOPSIS), validado por um modelo de previsão de tráfego

baseado em Memória de Longo e Curto Prazo (LSTM) bidirecional empilhado. O (NIDHI; LOBIYAL, 2021) sugere uma estratégia para implantar RSUs em áreas urbanas densas, visando melhorar a comunicação V2I. O (RODRIGUES et al., 2021b) propõe um modelo SPN para analisar o desempenho de uma infraestrutura VANET com múltiplos semáforos inteligentes cooperativos, avaliando métricas como tempo médio de resposta, utilização de recursos e número de descartes de requisições.

A proposta deste trabalho se enquadra no agrupamento de infraestrutura e gerenciamento, sendo focada na avaliação de desempenho de uma infraestrutura de monitoramento inteligente em redes veiculares, com o uso de RSUs e um modelo SPN para otimizar a comunicação e o gerenciamento de recursos. O objetivo é determinar a configuração onde as requisições tenham menor tempo de resposta, bem como equilíbrio entre a alocação de recursos e custos, garantindo maior desempenho no gerenciamento da arquitetura. Os resultados de desempenho encontrados servem como *input* para trabalhos que tenham foco em protocolos de roteamento, conectividade, segurança da comunicação, QoS, dentre outros contextos na temática de VANETs.

4 Metodologia

O objetivo principal deste trabalho é desenvolver um modelo para avaliar o processamento de mensagens em um sistema de filas, com múltiplas filas de prioridade. A Figura 5 apresenta um fluxograma que resume a estratégia usada neste trabalho.

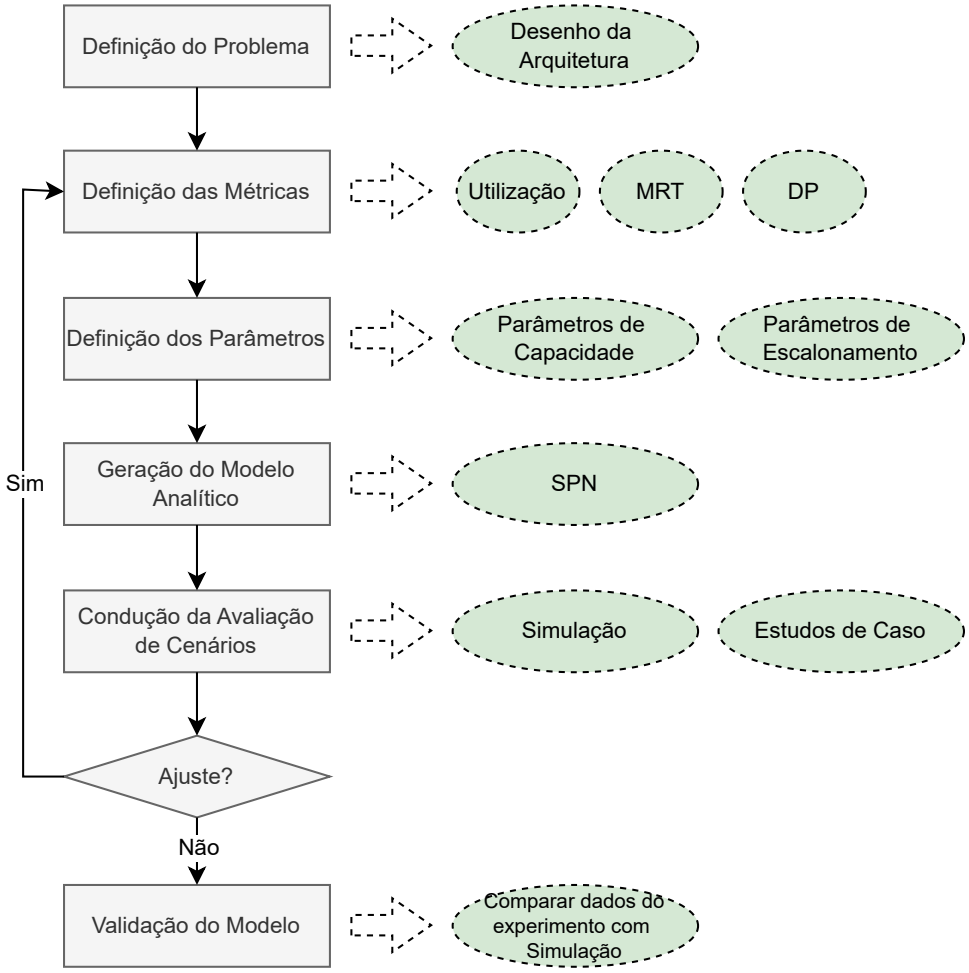


Figura 5 – Metodologia de desenvolvimento do modelo analítico.

- **Definição do Problema:** A etapa inicial foi dedicada à identificação do problema, onde o foco se centrou na necessidade de um modelo que avalie o desempenho de um sistema de filas com múltiplas prioridades. Tal análise levou à formulação de um sistema composto por RSUs capazes de processar requisições de diferentes classes. O que levou a elaboração de uma arquitetura do problema.

- **Definição das Métricas:** As métricas de interesse são identificadas, levando em consideração as informações que o modelo pode fornecer para diagnosticar o desempenho do sistema. As métricas selecionadas podem ser importantes na percepção do

usuário final e úteis para administradores de sistema, são elas: Utilização da RSU (URSU), Utilização do Servidor de *Fog* (UFS), Utilização da Fila na RSU (UQRSU), Utilização da Fila no Servidor de *Fog* (UQFS), Probabilidade de Descarte (DP) e Tempo Médio de Resposta (MRT).

- **Definição dos Parâmetros:** Os parâmetros que serão inseridos no modelo são definidos. Esses parâmetros definem o comportamento e capacidade de recursos para os componentes no sistema de RSUs

- **Geração do Modelo Analítico:** Utilizando o ambiente de modelagem Mercury, o modelo foi gerado com base em Redes de Petri Estocásticas. Para cada cenário, métricas de desempenho foram definidas, incluindo tempo de resposta médio, probabilidade de descarte e utilização das RSUs. As classes de requisições foram configuradas de forma a priorizar mensagens críticas, como alertas de desastre, sobre requisições de menor prioridade, como conforto dos passageiros.

- **Condução da Avaliação de Cenários:** Os cenários construídos são avaliados usando SPN por meio de simulação. A simulação permite modelar o funcionamento do sistema em diferentes situações, variando os níveis dos fatores relevantes em cada cenário. As métricas escolhidas serão analisadas, permitindo perceber com quais configurações o sistema possui desempenho satisfatório.

- **Ajuste:** Após a concepção do modelo, o experimento é realizado através de Simulação Estacionária. Os dados são coletados e analisados inicialmente em gráfico de planilha

- **Validação do Modelo:** Um ambiente de sistema distribuído em máquinas virtuais é utilizado e parametrizado conforme parte do modelo SPN. Os dados reais são coletados para comparar e validar os dados do modelo analítico. Os resultados serão comparados com os valores retornados pelo modelo através do intervalo de confiança e teste T. Se os valores forem semelhantes, o modelo estará validado, caso contrário, haverá a necessidade de ajustes no modelo. Se após a validação for detectada a necessidade de realizar ajustes no modelo, deve-se retornar à etapa de geração do modelo analítico.

Essas etapas garantem uma estrutura sólida para o desenvolvimento e validação do modelo, alinhando as necessidades teóricas com as soluções práticas oferecidas pela arquitetura proposta.

5 Arquitetura e Modelos SPN

Este capítulo apresenta o projeto de arquitetura e o modelo transposto em SPN. Observando o contexto, foram definidos os componentes que atendem às funcionalidades do sistema. Partindo da definição da arquitetura, três modelos SPN foram criados e seus parâmetros definidos.

5.1 Visão Geral

Esta seção apresenta uma arquitetura VANET com RSU, que servirá de referência para a modelagem e análise de desempenho. A Figura 6 representa a arquitetura proposta, que é dividida em três camadas chamadas *(i)* Admissão, *(ii) Edge* e *(iii) Fog*. A camada de Admissão representa os veículos ao longo da via, que enviam requisições *wireless* para a próxima camada. A segunda camada possui uma RSUs, de forma que ao receber uma mensagem, processa e encaminha para próxima camada. O servidor na camada de *Fog* recebe a requisição e realiza o processamento final. A arquitetura abstrai o envio de dados entre os veículos (V2V), bem como da infraestrutura para os veículos. Em um ITS completo, haveria uma ou mais RSUs na camada de *Edge*, assim como um ou mais servidores na camada de *Fog* realizariam o processamento intermediário, sendo que cada um poderia ser responsável por um bairro, distrito ou região. Por exigir maior capacidade de processamento e armazenamento, o sistema de gerenciamento geral ficaria na camada de *Cloud*. Para fins deste estudo, não foi abordada a transmissão de dados para camadas mais altas como a *Cloud*.

Nesta arquitetura foram consideradas três classes de requisições prioritárias, identificadas por `C1`, `C2` e `C3`. A classe `C1` reúne as requisições de maior prioridade, a classe `C2` agrupa requisições de média prioridade e a classe `C3` contém as requisições de menor prioridade. A classe `C1` pode abranger principalmente mensagens como alertas de desastres naturais, evacuação, sincronização entre os componentes da VANET, dentre outros. A classe `C2`, pode agrupar mensagens do tipo alerta de colisão, alerta de passagem de veículos dos serviços de emergência e segurança. As mensagens de classe `C3` englobariam os demais tipos de requisições relacionadas a congestionamento, conforto e entretenimento dos passageiros. Na camada *Edge*, a RSU classifica cada requisição e colocada na fila de prioridade apropriada. As requisições mantém a mesma classificação ao serem encaminhadas para a camada de *Fog*. As filas de prioridades podem trabalhar com disciplinas diferentes entre si, tais como *First Come, First Served* (FCFS), *Last Come, First Served* (LCFS) e *Service In Random Order* (SIRO). A disciplina adotada em uma fila não interfere nas demais filas de prioridade. Esta arquitetura considera considera um sistema de filas não preemptivo, ou

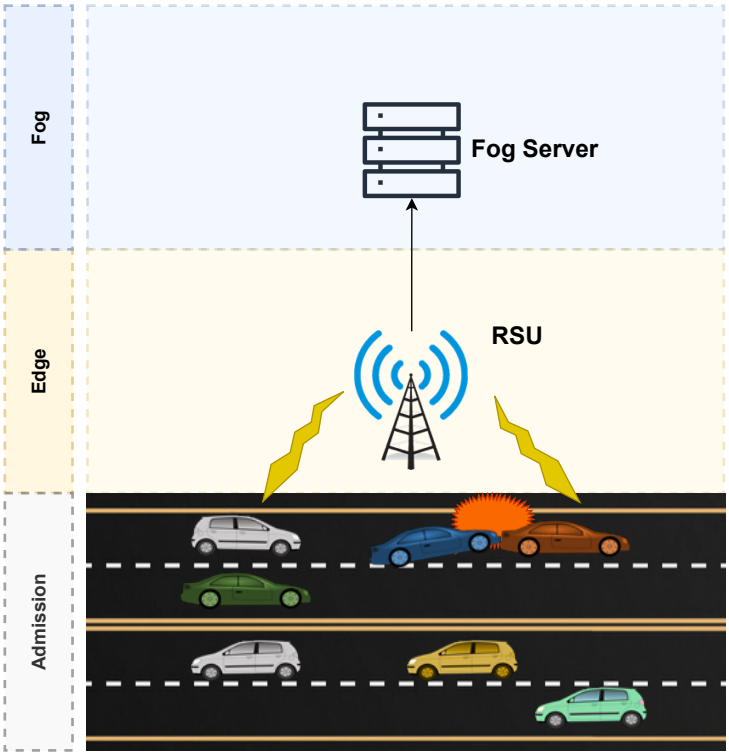


Figura 6 – Arquitetura proposta

seja, uma mensagem de maior prioridade nunca interrompe um processamento iniciado.

5.1.1 Suposições e discussões

Para a simplificação da modelagem, várias suposições sobre a arquitetura e as operações da infraestrutura em questão são dadas a seguir:

- Pontos de acesso de configurações heterogêneos são desconsiderados. No modelo a representação é de um dispositivo genérico e novos serão sempre do mesmo tipo.
- Os protocolos de comunicação entre as camadas e dispositivos não foram levados em consideração.
- O aumento na quantidade de veículos em determinados horários aumenta o envio de mensagens a partir do servidor central, o que é representado pela taxa de envio de mensagens.
- Considerou-se que os veículos são equipados com *On-board Unit* (OBU). Esse equipamento é responsável pela comunicação com as RSUs.
- Mecanismos de transmissão e roteamento entre RSUs não são necessários para este estudo.
- O modelo não considera o tempo de vida de uma requisição. Neste caso uma requisição aceita continuará na fila até ser processada.

- Desconsiderou-se o uso de preempção, pois aumentaria a complexidade do modelo SPN. Um *token* (requisição) não guarda informações sobre a mensagem, apenas representa o momento ou estado da requisição (está na fila, está processando). Assim seriam necessárias mais informações para determinar que uma requisição deve interromper o processamento de outra requisição, tal como uma situação de desastre natural. No contexto deste trabalho, uma mensagem de desastre natural já está englobada na classe `C1`, de maior prioridade.

- A disciplina de fila é indiferente para o nível de detalhes do modelo SPN proposto. Detalhar a disciplina de fila possui as mesmas implicações quanto ao uso de preempção. Assim, considera-se que a disciplina empregada em cada fila é FCFS.

5.2 Modelo SPN Base, com 1 Classe de Prioridade Genérica

Redes de Petri permitem modelar propriedades de concorrência e sincronia de sistemas (SANTOS et al., 2021). A Figura 7 representa um modelo SPN, para a arquitetura proposta na Figura 6. O modelo foi gerado na ferramenta *Mercury* V5.02 (OLIVEIRA et al., 2017), que também gera a avaliação conforme cenários propostos. A interpretação da transmissão de dados ocorre da esquerda para a direita. O Modelo possui três entradas, caracterizando o envio de requisições de diferentes classes de prioridade pelos veículos. As transições temporizadas `ArrivalC1`, `ArrivalC2` e `ArrivalC3` representam a geração de requisições para as classes de prioridade `C1`, `C2` e `C3`, respectivamente. A geração de *tokens* nas três transições seguem uma distribuição exponencial. O lugar `ArrivedQ` representa as requisições que chegam para entrar no sistema.

A Taxa de Chegada (AR) na camada de admissão varia conforme as requisições são enviadas pelos veículos, que dependem do volume de veículos em um dado momento, bem como das ocorrências que impactam no fluxo da via. Uma RSU admite e processa requisições de diversas classes de prioridade. Todavia, o modelo base trata as requisições admitidas de forma genérica, ou seja, não há distinção de prioridade. Todas as requisições ocuparão a mesma fila, seguindo a disciplina de fila FCFS. A RSU possui uma fila e vários núcleos de processamento. O lugar `RQAvailable` é a capacidade da fila de requisições, que tem tamanho máximo `RQCapacity`. Os *tokens* migram de `RQAvailable` para `RQ`, que representa a fila crescendo até o tamanho `RQCapacity`. O lugar `RPAvailable` representa a capacidade de processamento, onde possui até `RPCapacity` núcleos disponíveis para processamento. Um *token* em `RQ` migrará para `RP` se houver núcleo (*token*) disponível em `RPAvailable`. Esse *token* representa que há um núcleo processando uma requisição. O tempo médio de processamento é representado pelo disparo da transição temporizada `TR`, que segue uma distribuição exponencial.

Os dados processados são enviados através um *gateway* para um servidor na camada

de *Fog*. De forma similar a RSU, o lugar `FSQAvailable` é a capacidade da fila, que tem tamanho máximo `FSQCapacity`. Quando uma requisição é processada na RSU, um *tokens* em `RP` e outro em `FSQAvailable` migram para `FSQ`, que representa a fila crescendo até o tamanho `FSQCapacity`. O lugar `FSPAvailable` representa a capacidade de processamento, onde possui no máximo `FSPCapacity` núcleos disponíveis para processamento. Um *token* em `FSQ` migrará para `FSP` se houver núcleo (*token*) disponível em `FSPAvailable`. O disparo da transição temporizada `TFS` significa que uma requisição foi processada. O tempo médio de serviço das transições `TR` e `TFS` foram extraídos da literatura sobre o tema (SILVA et al., 2024). Os valores usados no modelo estão dispostos nas Tabelas 2 e 3. Os tempos de chegada `ADC1`, `ADC2` e `ADC3` foram definidos na calibração do modelo.

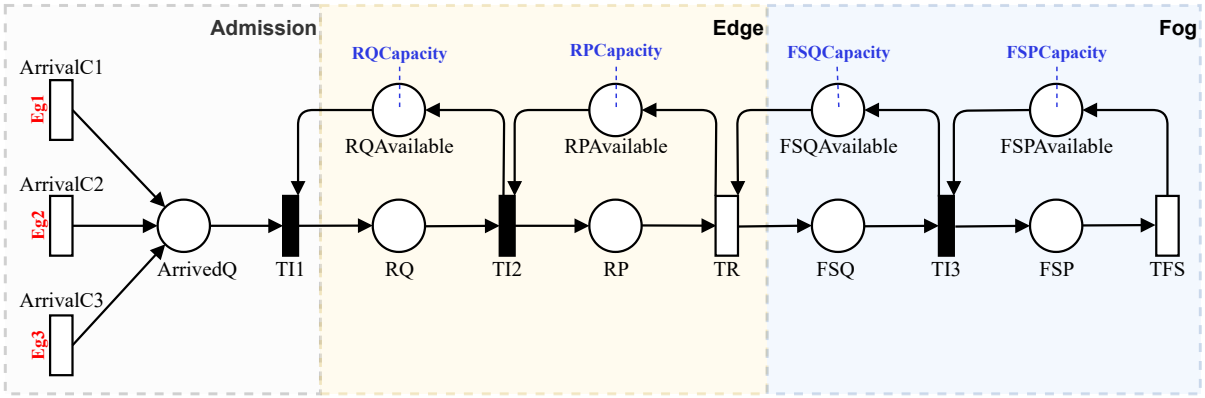


Figura 7 – Modelo com duas máquinas, contendo uma única fila, que pode conter requisições prioritárias e não prioritárias

Tabela 2 – Definições utilizadas no modelo SPN da Figura 7

Elemento	Descrição	Valor
`ADC1`	Tempo médio entre chegadas da classe `C1`	1/AR ms
`ADC2`	Tempo médio entre chegadas da classe `C2`	5.0 ms
`ADC3`	Tempo médio entre chegadas da classe `C3`	12.5 ms
`RST`	Tempo médio de serviço da RSU	15.7 ms
`FSST`	Tempo médio de serviço do Servidor de *Fog*	15.7 ms
`RQCapacity`	Capacidade máxima da fila na RSU	40
`RPCapacity`	Núcleos de processamento da RSU	8, 12, 16, 20, 24
`FSQCapacity`	Capacidade máxima da fila do Servidor de *Fog*	40
`FSPCapacity`	Núcleos de processamento do Servidor de *Fog*	20

Tabela 3 – Principais transições temporizadas utilizadas no modelo SPN da Figura 7

Elemento	Descrição	Valor
`ArrivalC1`	Tempo associado às chegadas da classe `C1`	`ADC1`
`ArrivalC2`	Tempo associado às chegadas da classe `C2`	`ADC2`
`ArrivalC3`	Tempo associado às chegadas da classe `C3`	`ADC3`
`TR`	Tempo associado ao processamento de uma requisição na RSU	`RST`
`TFS`	Tempo associado ao processamento de uma requisição no Servidor de *Fog*	`FSST`

A AR mede a frequência de mensagens que chegam ao sistema por unidade de tempo. O modelo usa tempos em milissegundos, assim as medições foram expressas em mensagens por milissegundos (msg/ms). A AR é calculada como o inverso do Atraso de Chegada (AD), que representa o tempo médio entre as chegadas consecutivas de mensagens e é expresso em milissegundos (ms). Como as métricas são relacionadas e inversamente proporcionais, temos que $AR = 1/AD$ e $AD = 1/AR$. A Tabela 4 apresenta essa relação com base em valores utilizados no modelo, considerando `AR1` variando entre 0,05 msg/ms e 1,2 msg/ms.

Tabela 4 – Relação entre `AR1`, `ADC1` e Interpretação dos Valores

AR (msg/ms)	**AD (ms)**	**Interpretação**
0.05	20.0	Uma mensagem a cada 20 ms
0.10	10.0	Uma mensagem a cada 10 ms
0.20	5.0	Uma mensagem a cada 5 ms
0.50	2.0	Uma mensagem a cada 2 ms
0.75	1.33	Três mensagens a cada 4 ms
1.00	1.0	Uma mensagem a cada ms
1.50	0.67	Três mensagens a cada 2 ms
2.00	0.5	Duas mensagens a cada ms

A transição temporizada `ArrivalC1` envia dados com taxa de chegada (`AR1`) variando de 0,05 a 1,2 mensagens por milissegundos (msg/ms), com incremento de 0,05. O valor utilizado na transição é calculado conforme a expressão: $ADC1 = 1/AR1$. As transições `ArrivalC2` e `ArrivalC3` permanecem com valores fixos, respectivamente `ADC2` e `ADC3` para os intervalos entre chegadas, conforme a Tabela 2. Assim temos as taxas de chegada $AR2 = 1/ADC2$ e $AR3 = 1/ADC3$. Cada uma das três transições é ativada somente se atender a respectiva equação de guarda, onde $Eg1 = Eg2 = Eg3$ e $Eg1 = (ArrivedQ <= 2)$. A condição de guarda é necessária para que as transições `ArrivalC1`, `ArrivalC2` e `ArrivalC3` possam disparar, mesmo que existam *tokens* em ArrivedQ. A taxa de chegada total do sistema é dada por:

$$AR_{Total} = AR1 + AR2 + AR3 \tag{5.1}$$

5.2.1 Métricas

As métricas adotadas nesse trabalho são: Tempo Médio de Resposta (MRT), Utilização da RSU (URSU), Utilização do Servidor de *Fog* (UFS), Utilização da Fila na RSU (UQRSU), Utilização da Fila no Servidor de *Fog* (UQFS) e Probabilidade de Descarte (DP). O cálculo do MRT é efetuado conforme a Lei de Little (LITTLE, 1961). Conforme esta lei, o MRT é dado pelo total de requisições dentro do sistema multiplicado pelo Atraso de Chegada (AD). Um sistema estável é pré-requisito para a Lei de Little, ou seja, que a taxa de chegadas seja menor do que a taxa de serviço. Desta forma o

sistema não ficará sobrecarregado. MRT é igual ao produto da quantidade de requisições em progresso, multiplicado pelo tempo entre chegadas de requisições, no entanto, o modelo proposto possui três entradas de dados simultâneas. Assim, para calcular o MRT no modelo proposto foi necessário utilizar o tempo da última transição, denominado `FSST`, ou seja, o tempo médio de serviço do servidor da camada de *Fog*. Para encontrar o número de requisições em progresso, é necessário obter a quantidade momentânea total de *tokens* que estão sendo processados desde o início do modelo até o final. Para esse cálculo basta somar a esperança estatística de existir *tokens* em todos os lugares onde a requisição passará (RODRIGUES et al., 2021a). Por não utilizarmos o tempo médio entre chegadas, o cálculo do MRT deverá ser multiplicado pela probabilidade de se ter ao menos um *token* no lugar `FSP`.

A métrica URSU calcula a porcentagem da capacidade processamento da RSU, que está sendo consumida a uma uma determinada taxa de chegada de requisições. A métrica UFS, se refere a utilização da capacidade de processamento do servidor da *Fog*. A métrica UQRSU está associada à utilização da capacidade de fila na RSU, ou seja, a ocupação da fila por requisições aguardando processamento. De forma análoga, a métrica UQFS analisa a utilização da fila no servidor da *Fog*. Já a DP se refere às requisições que chegam ao sistema, mas que podem ser ser descartadas devido a elevada ocupação do sistema causada por sobrecarga na utilização. O descarte pela camada *Fog* não é abordado, pois o modelo considera que uma requisição admitida será processada.

Tabela 5 – Expressões das métricas utilizadas no modelo SPN Base

Métrica	Expressão
DP	$(P\{\#RQAvailable = 0\}) * 100$
MRT	$(((E\{\#RQ\}) + (E\{\#RP\}) + (E\{\#FSQ\}) + (E\{\#FSP\})) * (FSST)) * (P\{(\#FSP > 0)\})$
URSU	$(E\{\#RP\}/RPCapacity) * 100$
UFS	$(E\{\#FSP\}/FSPCapacity) * 100$
UQRSU	$(E\{\#RQ\}/RQCapacity) * 100$
UQFS	$(E\{\#FSQ\}/FSQCapacity) * 100$

5.2.2 Estudo de Caso 1

O primeiro estudo de caso analisa o desempenho do sistema de filas da Figura 7, conforme as equações das métricas listadas na Tabela 5. Dois cenários de testes foram idealizados, cujas configurações para os Cenário 1 e Cenário 2 estão listadas nas Tabelas 6 e 7, respectivamente. O estudo de caso visa compreender como diferentes configurações de capacidade afetam o desempenho do modelo, onde a variação na taxa de chegada reflete situações de baixa e alta demanda no trânsito, que podem afetar a comunicação em casos de desastes naturais ou atendimentos de emergência. A análise também permite determinar quais configurações permitem alcançar um equilíbrio entre custo e desempenho

ao implantar o sistema. O termo **núcleo** é uma abstração para representar o valor da capacidade de processamento na RSU, bem como no servidor de *Fog*. Em outros contextos, a capacidade de processamento pode significar *threads*, serviços, computadores, contêineres, dentre outros. A estratégia de descarte da fila é *drop* ao atingir o limite estabelecido para o tamanho da fila, que é de 40 em cada máquina. A política de fila utilizada nos componentes das duas camadas é Não-Preemptiva e usa a estratégia FCFS.

Tabela 6 – Parâmetros para RSU - Cenário 01

Configuração	`RPCapacity`	`FSPCapacity`
A	8 núcleos	20 núcleos
B	12 núcleos	20 núcleos
C	16 núcleos	20 núcleos
D	20 núcleos	20 núcleos
E	24 núcleos	20 núcleos

Tabela 7 – Parâmetros para *Fog* - Cenário 02

Configuração	`RPCapacity`	`FSPCapacity`
F	20 núcleos	8 núcleos
G	20 núcleos	12 núcleos
H	20 núcleos	16 núcleos
I	20 núcleos	20 núcleos
J	20 núcleos	24 núcleos

5.2.2.1 Cenário 1

O gráfico da Figura 8(a) mostra a taxa de utilização da RSU para as quatro configurações. A AR_{Total} é dada em função de `AR1`, que inicia baixa, mas `AR2` e `AR3` possuem valores iniciais maiores. Assim é possível verificar que a configuração com 8 núcleos possui alta taxa de utilização inicial, de aproximadamente 64,76% para o `AR1` de 0,05 msg/ms e AR_{Total} = 0,33 msg/ms. A partir deste ponto, a taxa de chegada citada será referente ao `AR1`. O AR_{Total} será especificado quando necessário. A RSU satura rapidamente com taxas de chegada relativamente baixas de `AR1`, alcançando 99,96% de utilização com `AR1` de 0,3 msg/ms. Conforme a quantidade de núcleos aumenta, a RSU precisa de maiores taxas de chegada para atingir a saturação. Este comportamento é esperado, pois a disponibilidade de mais núcleos de processamento permite lidar com maiores cargas de trabalho. A saturação com 12 núcleos ocorre a partir de 0,6 msg/ms de `AR1`, com 99,98% de utilização. É interessante observar que o aumento de 8 para 12 núcleos faz com que a taxa de utilização inicial seja de 43,17%, ou seja 33,34% menor do que a utilização com 8 núcleos. Esse decréscimo na utilização inicial também é observado à medida que aumenta a quantidade de núcleos. A configuração com 24 núcleos apresenta um pequeno aumento na utilização por volta de 0,9 msg/ms, mas sofre saturação apenas com taxas de chegada a partir de 1.15 msg/ms quando atinge 99,92% de utilização.

O gráfico da Figura 8(b) mostra a taxa de utilização da *Fog*. Na configuração de 8 núcleos, a taxa de utilização estabiliza em aproximadamente 40% da capacidade de processamento, com taxa de chegada a partir de 0,3 msg/ms. Essa estabilização corresponde a saturação da RSU. A configuração com 12 núcleos atinge o platô a partir de 0,6 msg/ms de `AR1` e utilização 59,98%, momento em que a RSU sobrecarrega. É possível verificar que nas cinco configurações as taxas de utilização seguem alinhadas até que convergem para seus respectivos platôs. Na RSU as taxas de utilização possuem crescimento acentuado para configurações com menos núcleos de processamento. Esse comportamento é ligeiramente diferente com 24 núcleos, que sofre pequeno aumento na utilização antes de sobrecarregar. Esse comportamento se dá em razão da *Fog* demorar mais a processar as requisições, fazendo com que a RSU precise aguardar capacidade de processamento na *Fog*. O servidor da *Fog* é menos impactado em configurações onde a RSU possui menos núcleos. Configurações com menos núcleos na RSU farão com que a *Fog* fique subutilizada. Na configuração com 20 núcleos as máquinas nas duas camadas passam a ter as mesmas características, mesmo assim o servidor na *Fog* não sobrecarrega.

O gráfico da Figura 8(c) apresenta utilização da fila para a RSU. A medida que a taxa de chegada aumenta e os núcleos ficam ocupados, as novas requisições começam a esperar na fila para entrarem em processamento. Para a configuração com 8 núcleos, a utilização da fila passa de 7,52% para 87,96%, com `AR1` igual a 1,5 msg/ms e 0,3 msg/ms, respectivamente. Nesta taxa de chegada a utilização da fila converge para o platô de saturação. Ressalta-se que a partir desta taxa de chegada, a utilização do processamento também satura. Na configuração com 12 núcleos a utilização da fila é quase nula para baixas taxas de chegada. Todavia a utilização da fila salta de 11,91% para 89,27% com taxas de chegada de 0,4 msg/ms e 0,6 msg/ms, respectivamente. A utilização do processamento satura a esta taxa de chegada. Quando a taxa de utilização do processamento se aproxima da sobrecarga, em todas as configurações a utilização da fila cresce rapidamente devido ao acumulo de requisições.

É importante destacar que RSU capta o AR_{Total} e processa a uma Taxa de Serviço (SR) tal que $SR = 1/ServiceTime$. Assim a taxa de chegada para *Fog* pode ser diferente de AR_{Total}. Considerando que a RSU sobrecarrega a partir de determinado AR_{Total} e que só pode enviar requisições para a *Fog* no máximo a SR, conforme os conceitos de Teoria das Filas é possível afirmar que a AR'_{Total} para a *Fog* é:

$$AR'_{Total} = min(AR_{Total}, SR) \tag{5.2}$$

O gráfico da Figura 8(d) mostra a ausência de filas para taxas de chegada menores do que 0,7 msg/ms, ou seja, com utilização a baixo de 1%. As configurações com 8 e 12 núcleos permanecem com utilização a baixo de 1% para taxas de chegada maiores. A configuração com 16 núcleos começa a utilizar fila, mas ainda fica a baixo de 3%. A

configuração de 20 núcleos utiliza no máximo 26,56% da fila e a configuração de 24 núcleos alcança o máximo de 46,7%. Em ambos os caso ocorre a permanência em um platô. Observa-se que a *Fog* consegue processar em tempo suficiente para que a utilização da fila fique sempre a baixo de 50%. Isto corrobora a afirmação referente a AR'_{Total}.

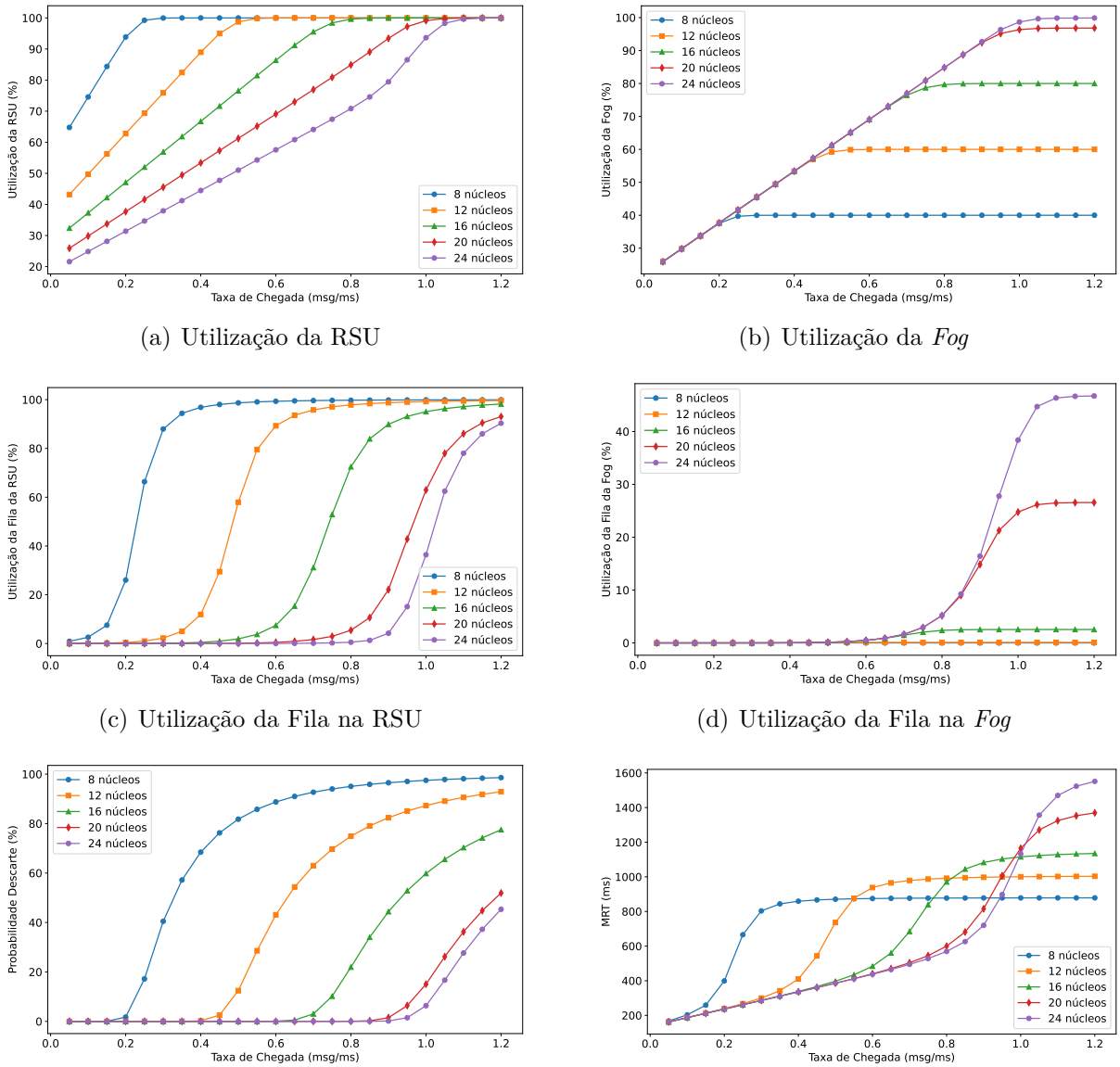
O gráfico da Figura 8(e) mostra a probabilidade de descarte de novas requisições que chegam ao sistema. As filas começam a descartar mensagens quando estão cheias, e a probabilidade de descarte aumenta drasticamente à medida que a taxa de chegada cresce. Na configuração com 8 núcleos, observa-se que com 0,2 msg/ms a probabilidade de descarte deixa de ser nula e se eleva abruptamente, onde atinge o máximo de 98,59% com `AR1` de 1,2 msg/ms. Essa elevação é condizente com a rápida ocupação da fila na RSU, tal qual é o mesmo momento em que a taxa de utilização do processamento inicia a convergência para a saturação. Comportamento análogo é observado para as demais configurações, entretanto mais núcleos permitem lidar melhor com taxas mais altas antes de começarem a descartar mensagens. A configuração de 24 núcleos chega no máximo 45,32% de probabilidade de descarte. Assim a configuração de 8 núcleos é 117% pior do a configuração de 24 núcleos.

O MRT é mostrado no gráfico da Figura 8(f). O tempo médio de resposta aumenta em razão do acúmulo de mensagens na fila, de tal forma que eleva o tempo de espera antes que as requisições sejam processadas. As configurações com menores quantidades de núcleos atingem elevados tempos de resposta rapidamente. As configurações com 8 e 12 núcleos estabilizam em aproximadamente 878 ms e 1003 ms, respectivamente. Configurações com 16, 20 e 24 núcleos mantém tempos de resposta mais baixos por mais tempo. Todavia quando se aproximam da capacidade máxima, o tempo de resposta se eleva mais rápido. Assim, quanto mais rápida for a saturação da RSU, menor será a taxa de utilização da *Fog*, bem como serão maiores o MRT e a probabilidade de descarte.

5.2.2.2 Cenário 2

Neste Cenário a capacidade de processamento da RSU permanece fixa e a variação ocorre com a capacidade de processamento da *Fog*. A Figura 9(a) mostra taxa de utilização para RSU. As configurações de 8 e 12 núcleos da *Fog* possuem crescimento linear conforme aumenta a taxa de chegada. Entretanto as configurações menores apresentam crescimento abrupto para taxas de chegada razoavelmente baixas, com consequente saturação. A exemplo da configuração com 8 núcleos, a utilização da RSU vai de 42,82% com 0,2 msg/ms para 92,36% com 0,25 msg/ms. Com 16 núcleos esse salto é menor, ao passo que nas configurações maiores a convergência para a saturação é mais suave.

A Figura 9(b) apresenta o gráfico de utilização da *Fog*. A configuração com 8 núcleos inicia com utilização de 64,76% e apresenta crescimento linear até 0,2 msg/ms, onde utiliza 94,24% do processamento. A partir desta taxa de chegada a *Fog* sobrecarrega



(a) Utilização da RSU

(b) Utilização da *Fog*

(c) Utilização da Fila na RSU

(d) Utilização da Fila na *Fog*

(e) Probabilidade de Descarte

(f) Tempo Médio de Resposta (MRT)

Figura 8 – Resultados da análise do modelo considerando diferentes números de núcleos para RSU no Cenário 1

e acelera a utilização na RSU. A configuração com 12 núcleos utiliza 43,17% da capacidade inicial, sendo 33% menor do que a configuração anterior. As configurações com 12 e 16 núcleos tem crescimento menos acentuado, demorando mais a saturar. As configurações com 20 e 24 núcleos dispõe de capacidade suficiente para não saturar, respectivamente usando no máximo 96,83% e 83,27%. Maiores capacidades da *Fog* retardam a sobrecarga da RSU.

A configuração com 8 núcleos apresenta baixa capacidade em relação a RSU, de tal forma que a RSU precisa aguardar e precisa ocupar por mais tempo os núcleos de processamento. Com taxa de chegada de 0,2 msg/ms, é possível verificar que a RSU sofre acumulo de requisições na fila, conforme a Figura 9(c). Da mesma forma a *Fog* começa

a acumular requisições enviadas pela RSU, de acordo com a Figura 9(d). Verifica-se que mesmo com a capacidade de processamento saturada, a *Fog* consegue processar de forma que as requisições ocupam apenas 80% da fila. Entretanto a configuração de 8 núcleos torna a *Fog* o gargalo do sistema. Como consequência, observa-se na Figura 9(e) que a taxa de descarte era nula e começa a se elevar. Como implicação temos o aumento súbito do MRT, pois agora as requisições precisam passar mais tempo dentro do sistema. Neste cenário, as baixas configurações da *Fog* fazem com que a RSU sobrecarregue rápido, aumentando a taxa de descarte e o tempo de resposta do sistema, conforme a Figura 9(f).

5.3 Modelos SPN estendidos, com 3 Classes de Prioridades

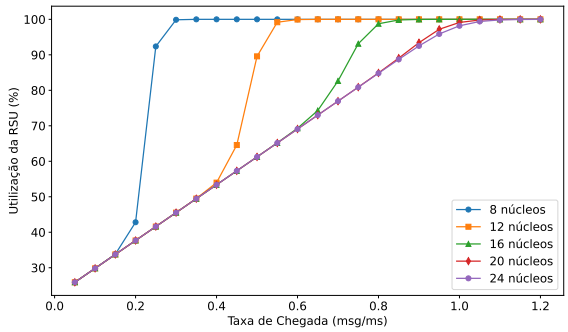
O modelo SPN da Figura 7 discutido na seção anterior, trabalha com admissão de três classes prioritárias de requisição, identificadas por `C1`, `C2` e `C3`. A geração de requisições para as três classes é representada pelas transições temporizadas `ArrivalC1`, `ArrivalC2` e `ArrivalC3`. Uma requisição alta prioridade sem o devido tratamento pode ocasionar prejuízos em cadeia. Diversas vidas podem se perder caso um alerta de desastre natural seja descartado. Incêndios podem se alastrar caso os bombeiros não sejam alertados sobre congestionamentos. Ambulâncias podem não chegar a tempo para socorrer acidentados se os veículos não foram notificados para abir passagem. Esses e outros casos levam a necessidade de tratamento diferenciado para classes de requisições.

Esta seção aborda dois modelos SPN para tratar as classes de requisições prioritárias do modelo anterior.

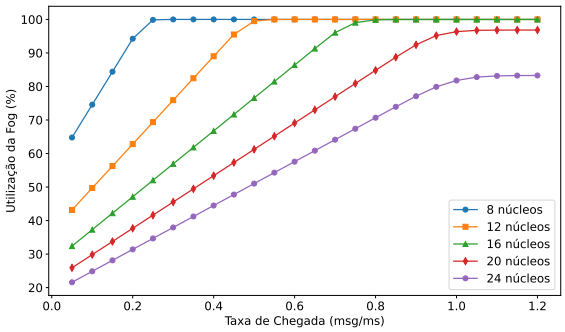
5.3.1 Modelo com Bloqueio entre Classes

O modelo SPN da Figura 10 expande o lugar `RQC1` acrescentando os lugares `RQC2` e `RQC3` para enfileirar as requisições que são geradas pelas transições `ArrivalC1`, `ArrivalC2` e `ArrivalC3`, respectivamente. Os *tokens* disponíveis no lugar `RQAvailable` agora são compartilhados entre `RQC1`, `RQC2` e `RQC3`. Também são acrescentados os lugares `RPC2` e `RPC3`, que recebem os *tokens* migrados de `RQC2` e `RQC3` respectivamente. Tal qual ocorre com `RQAvailable`, os *tokens* disponíveis no lugar `RPAvailable` agora são compartilhados entre `RPC1`, `RPC2` e `RPC3`. Analogamente, na *Fog* foram acrescentados os lugares `FSQC2`, `FSQC3`, `FSPC2` e `FSPC3`. A migração de *tokens* entre os lugares `ArrivalC1`, `RQC1`, `RPC1`, `FSQC1` e `FSPC1` representa o fluxo de uma requisição de classe `C1` dentro do sistema. Os lugares com sufixo `C2` e `C3` são referentes aos fluxos das classes `C2` e `C3`, respectivamente. A Tabela 8 apresenta os componentes adicionados em relação a Tabelas 2 e 3.

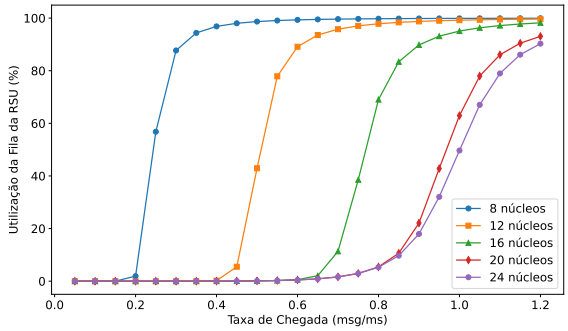
O modelo SPN da Figura 10 foi elaborado para que as classes prioritárias tivessem precedência para usar o processamento. As requisições da classe `C1` podem bloquear o processamento de requisições aguardando nas filas de classes `C2` e `C3`, se houver pelo



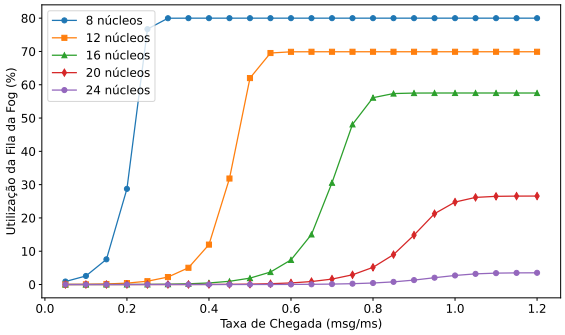
(a) Utilização da RSU



(b) Utilização da *Fog*



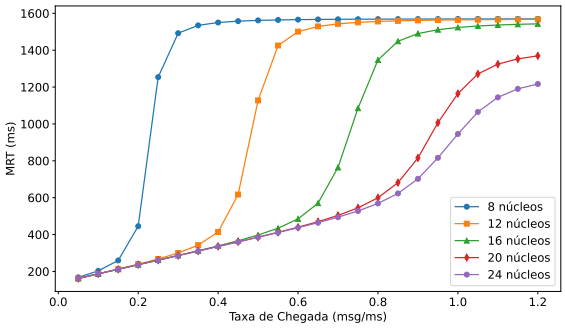
(c) Utilização da Fila na RSU



(d) Utilização da Fila na *Fog*

DissertaÃğÃčo/figs/arquitetura/cs1/cs1_

(e) Probabilidade de Descarte



(f) Tempo Médio de Resposta (MRT)

Figura 9 – Resultados da análise do modelo considerando diferentes números de núcleos para *Fog*, no Cenário 2

menos uma requisição no lugar `RQC1`. Da mesma forma, as requisições da classe `C2` podem bloquear requisições da classe `C3`, se houver pelo menos uma requisição no lugar `RQC2`. Neste caso não devem existir requisições de classe `C1` aguardando processamento em `RQC1`. Na *Fog* as requisições em `FSQC1` bloqueiam as requisições em `FSQC2` e `FSQC3`. Ao tempo que `FSQC2` bloqueia `FSQC3`, desde que `FSQC1` esteja vazio. Essa regra de bloqueio é incluída ao inserir os arcos inibidores, que estão destacados em vermelho.

Para calcular o comportamento das filas de prioridade, o modelo utiliza as métricas listadas na Tabela 9. O Cenário 1, estudado anteriormente na Seção 5.2.2.1 foi usado como referência para esta avaliação. As configurações com 8 e 20 núcleos foram selecionadas.

5.3.2 Estudo de Caso 2

O segundo estudo de caso analisa o desempenho do sistema de filas da Figura 10. Os gráficos da Figura 11 tratam da configuração com 8 núcleos para a RSU. No gráfico da Figura 11(a), verifica-se que as taxas de chegada das classes `C2` e `C3` são responsáveis pela alta utilização inicial da RSU. As classes `C2` e `C3` possuem taxas de chegada fixas, assim as taxas de utilização de `C2` e `C3` diminuem gradativamente conforme a taxa de chegada da classe `C1` aumenta. O gráfico também mostra taxa de utilização total é compatível com a taxa de utilização base, que foi calculada no Estudo de Caso 1. A linha Base mostra o ponto de saturação em 0,3 msg/ms, onde a taxa de utilização de `C2` apresenta decréscimo evidente. A classe `C3` tem a menor taxa de chegada, por isso concorre menos por recurso. O gráfico da Figura 11(b) mostra que a *Fog* fica subutilizada e consumindo somente 40% da capacidade, pois a RSU já está sobrecarregada e não consegue mais enviar requisições.

A Figura 11(c) mostra a utilização da fila da RSU, onde as requisições de classes `C2` e `C3` passam a se acumular na fila, em detrimento das requisições de classe `C1`. O acúmulo se incia após o processamento saturar. Esse acumulo é ocasionado pela prioridade da classe `C1` sobre as classes `C2` e `C3`, bem como pela prioridade da classe `C2` sobre a classe `C3`. Figura 11(d) mostra picos de utilização da fila na *Fog*, entretanto essa utilização fica a baixo de 0,001%, haja vista que a *Fog* consegue processar rapidamente. Conforme a Figura 11(e), a probabilidade de descarte se eleva de forma condizente com o aumento da taxa de chegada, bem como da ocupação da fila na RSU. A Figura 11(f) mostra que MRT também é afetado pelo acumulo de mensagens na fila, ou seja, quanto mais mensagens aguardando processamento, maior o tempo que elas passarão no sistema. O MRT da classe `C3` ultrapassa os 350 ms e para a classe `C1` fica no patamar de 150 ms, ainda se eleva devido ao crescimento da taxa de chegada.

Os gráficos da Figura 12 mostram os resultados para a configuração em que a RSU possui 20 núcleos de processamento. O aumento da capacidade de processamento permite manter o processamento constante para as classes `C2` e `C3`, que possuem baixas taxas de chegada, conforme as Figuras 12(a) e 12(b). O melhor balanceamento entre as máquinas faz com que as requisições de classe `C2` e `C3` sejam impactadas apenas em taxas de chega superiores a 0,9 msg/ms. Assim como observado na configuração com 8 núcleos, as requisições de classes mais baixas passam a se acumular na fila, em detrimento das requisições de classe maior, como visto na Figura 12(c). A classe `C2` passa a utilizar aproximadamente 12% da fila com o uso de bloqueio, ao tempo que a classe `C3` utiliza por volta de 68,45% da fila. Mesmo bloqueando as demais classes, a elevada taxa de chegada

da classe C1 faz com que ela utilize mais de 7% da fila.

Na Figura 12(d) observamos que a fila da *Fog* acumula menos requisições de classe C3 do que na RSU. Isto ocorre porque a RSU está limitando o envio de requisições classe C3 em detrimento da classe C1. É importante ressaltar que tamanho reduzido da fila na *Fog* é em razão da capacidade de processar rapidamente a demanda oriunda da RSU. Observa-se na Figura 12(e), que a probabilidade de descarte da classe C3 é baixa devido taxas de chegada reduzidas, mesmo com o bloqueio imposto pelas classes C1 e C2. Verifica-se na 12(f) que o MRT da classe C3 cresce no momento de saturação da RSU, pois as requisições começam a se acumular na fila. Quando a RSU satura, a classe C1 diminui o ritmo de crescimento pois agora passa a dominar o uso do processamento. Em relação às classes C2 e C3, haja vista que suas taxas de chegadas são relativamente baixas, a capacidade de processamento é suficiente para minimizar o descarte de requisições, bem como que MRT seja baixo.

Fazendo o exercício de inverter as classes de prioridade, onde $C1 < C2 < C3$ teríamos que o aumento de requisições de baixa prioridade afetariam minimamente serviços de urgência e segurança pública.

Tabela 8 – Componentes adicionados no modelo sem bloqueio entre classes

Tipo	Componentes	Descrição
Lugares	ArrivedQC2, ArrivedQC3	Geração de requições para classes C2 e C3
	RQC2, RQC3	Ocupação das filas C2 e C3 na RSU
	RPC2, RPC3	Ocupação do processamento de classes C2 e C3 na RSU
	FSQC2, FSQC3	Ocupação das filas C2 e C3 na *Fog*
	FSPC2, FSPC3	Ocupação do processamento de classes C2 e C3 na *Fog*
Transições	TR1C2, TR1C3	Transições temporizadas de saída da RSU
	TR2C2, TR2C3	Transições temporizadas de saída da *Fog*

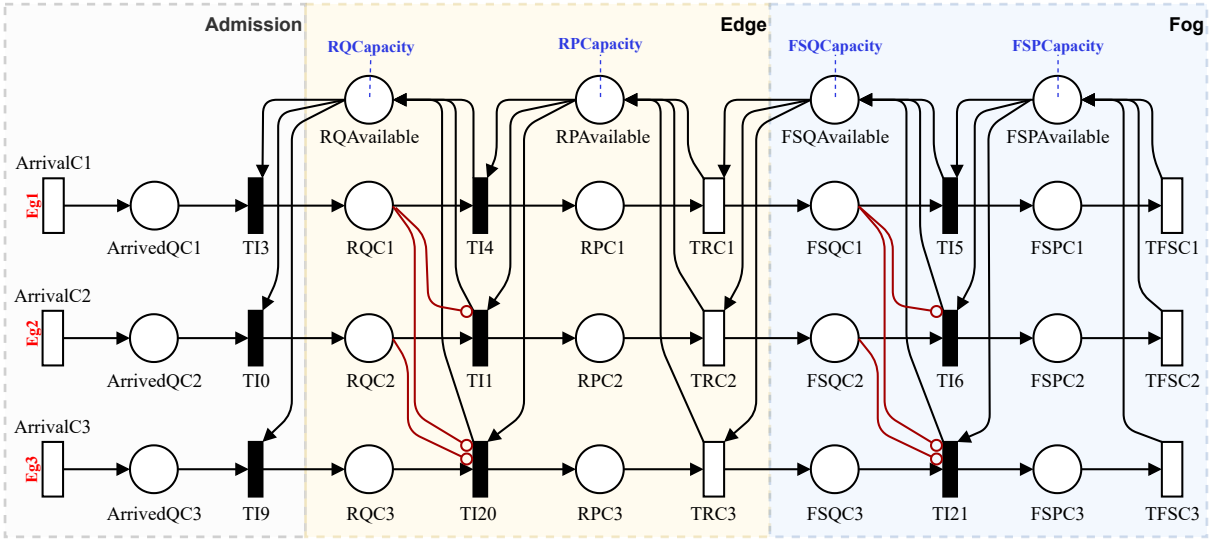


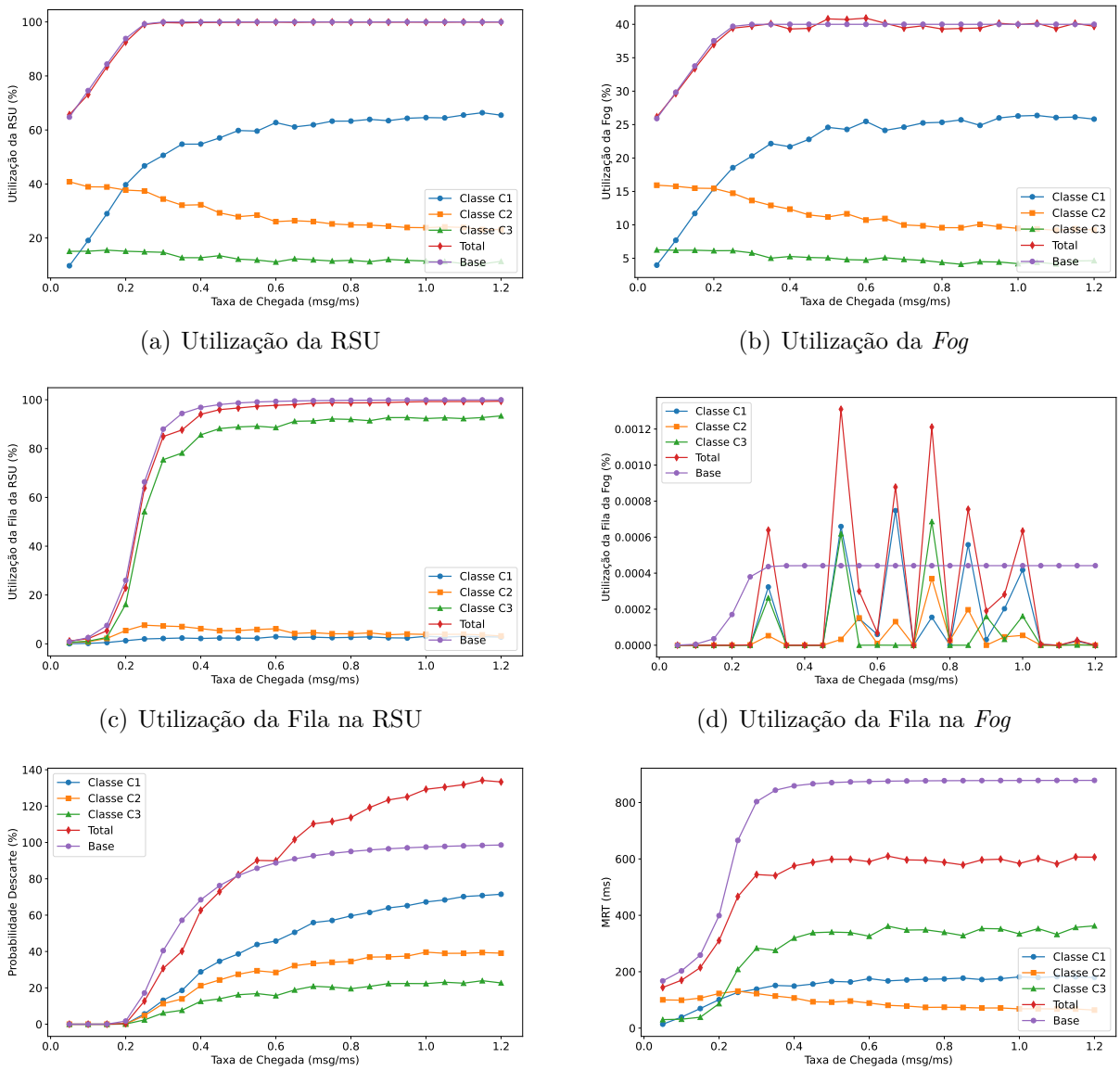
Figura 10 – Modelo contendo três filas de prioridades, usando bloqueio entre as filas

Tabela 9 – Métricas utilizadas no modelo SPN com três filas (Figura 10)

Métrica	Expressão	Descrição
DPC1	$(P\{(\#ArrivedQC1 > 0)AND(\#RQAvailable = 0)\}) * 100$	Probabilidade de descarte da classe `C1`
DPC2	$(P\{(\#ArrivedQC2 > 0)AND(\#RQAvailable = 0)\}) * 100$	Probabilidade de descarte da classe `C2`
DPC3	$(P\{(\#ArrivedQC3 > 0)AND(\#RQAvailable = 0)\}) * 100$	Probabilidade de descarte da classe `C3`
DPTotal	$((DPC1 + DPC2 + DPC3) * 100)$	Probabilidade de descarte na entrada do sistema
MRTC1	$(((E\{\#RQC1\}) + (E\{\#RPC1\}) + (E\{\#FSQC1\}) + (E\{\#FSPC1\})) * (FSST)) * (P\{(\#FSPC1 > 0)\})$	MRT da classe `C1`
MRTC2	$(((E\{\#RQC2\}) + (E\{\#RPC2\}) + (E\{\#FSQC2\}) + (E\{\#FSPC2\})) * (FSST)) * (P\{(\#FSPC2 > 0)\})$	MRT da classe `C2`
MRTC3	$(((E\{\#RQC3\}) + (E\{\#RPC3\}) + (E\{\#FSQC3\}) + (E\{\#FSPC3\})) * (FSST)) * (P\{(\#FSPC3 > 0)\})$	MRT da classe `C3`
MRTTotal	$(MRTC1 + MRTC2 + MRTC3)$	MRT do sistema
UQRC1	$(E\{\#RQC1\}/RQCapacity) * 100$	Uso da fila pela classe `C1` na RSU
UQRC2	$(E\{\#RQC2\}/RQCapacity) * 100$	Uso da fila pela classe `C2` na RSU
UQRC3	$(E\{\#RQC3\}/RQCapacity) * 100$	Uso da fila pela classe `C3` na RSU
UQRTotal	$(((E\{\#RQC1\}) + (E\{\#RQC2\}) + (E\{\#RQC3\}))/RQCapacity) * 100$	Uso da fila na RSU
UQFSC1	$(E\{\#FSQC1\}/FSQCapacity) * 100$	Uso da fila pela classe `C1` na *Fog*
UQFSC2	$(E\{\#FSQC2\}/FSQCapacity) * 100$	Uso da fila pela classe `C2` na *Fog*
UQFSC3	$(E\{\#FSQC3\}/FSQCapacity) * 100$	Uso da fila pela classe `C3` na *Fog*
UQR2Total	$(((E\{\#FSQC1\}) + (E\{\#FSQC2\}) + (E\{\#FSQC3\}))/FSQCapacity) * 100$	Uso da fila na *Fog*
URC1	$(E\{\#RPC1\}/RPCapacity) * 100$	Processamento da classe `C1` na RSU
URC2	$(E\{\#RPC2\}/RPCapacity) * 100$	Processamento da classe `C2` na RSU
URC3	$(E\{\#RPC3\}/RPCapacity) * 100$	Processamento da classe `C3` na RSU
URTotal	$(((E\{\#RPC1\}) + (E\{\#RPC2\}) + (E\{\#RPC3\}))/RPCapacity) * 100$	Processamento na RSU
UFSC1	$(E\{\#FSPC1\}/FSPCapacity) * 100$	Processamento da classe `C1` na *Fog*
UFSC2	$(E\{\#FSPC2\}/FSPCapacity) * 100$	Processamento da classe `C2` na *Fog*
UFSC3	$(E\{\#FSPC3\}/FSPCapacity) * 100$	Processamento da classe `C3` na *Fog*
UFSTotal	$(((E\{\#FSPC1\}) + (E\{\#FSPC2\}) + (E\{\#FSPC3\}))/FSPCapacity) * 100$	Processamento da *Fog*

5.3.3 Modelo com Escalonamento entre Classes

O modelo SPN da Figura 13 foi criado para contemplar escalonamento circular entre as classes de prioridade. Esta abordagem evita que as classes de maior prioridade



(a) Utilização da RSU

(b) Utilização da *Fog*

(c) Utilização da Fila na RSU

(d) Utilização da Fila na *Fog*

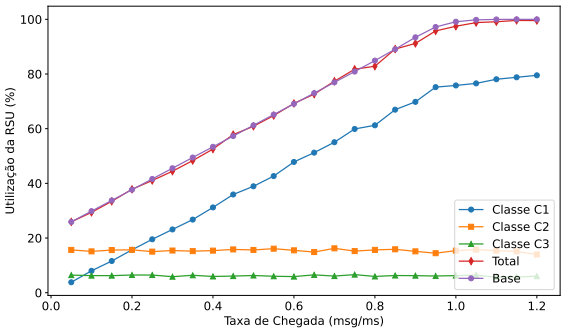
(e) Probabilidade de Descarte

(f) Tempo Médio de Resposta (MRT)

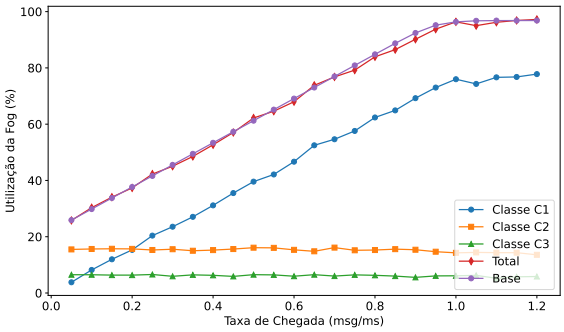
Figura 11 – Resultados da análise Caso 2, para RSU com 8 núcleos

bloqueiem as classes de menor prioridade, tal como ocorre com o modelo da Figura 10. Neste modelo cada classe envia uma determinada quantidade de requisições para o processamento, mas se as requisições acaberem antes disso, a próxima classe de menor prioridade estará habilitada a enviar. Quando a classe de menor prioridade finaliza seu envio, a classe de maior prioridade estará habilitada novamente. Quando não houver núcleos de processamento disponíveis, a classe habilitada enviará as requisições a medida que o processamento se torne disponível. Isso evita que uma classe de menor prioridade fique bloqueada por muito tempo.

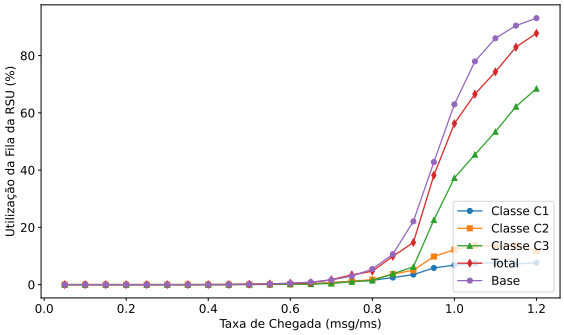
O *token* no lugar `R1C1N` indica que a classe `C1` está habilitada para enviar requisições para processamento. Caso existam requisições em `RQC1` a serem processadas, então `R1C1D` receberá `R1Seq1` *tokens* que representam a quantidade de requisições que a classe `C1` poderá



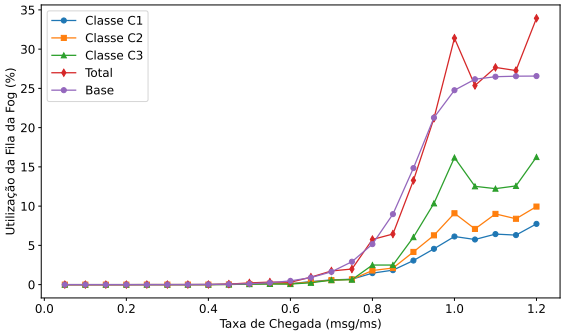
(a) Utilização da RSU



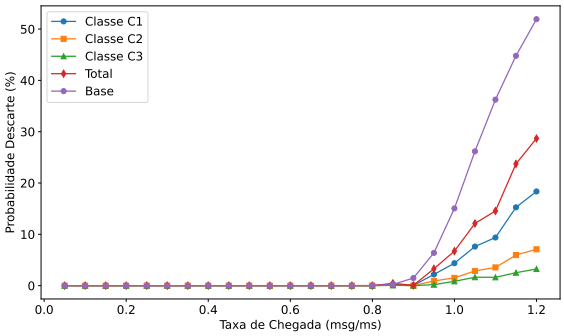
(b) Utilização da *Fog*



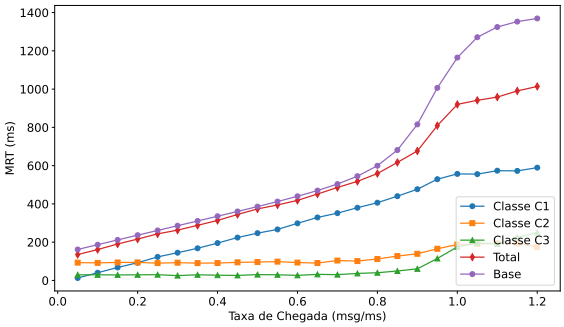
(c) Utilização da Fila na RSU



(d) Utilização da Fila na *Fog*



(e) Probabilidade de Descarte



(f) Tempo Médio de Resposta (MRT)

Figura 12 – Resultados da análise Caso 2, para RSU com 20 núcleos

enviar para processamento. Quando `R1C1D` esvaziar ou não houverem mais requisições em `RQC1`, então o lugar `R1C2N` receberá um *token* indicando que a classe `C2` estará habilitada para processar requisições. Se existirem requisições a serem processadas em `RQC2`, então o lugar `R1C2D` receberá `R1Seq2` *tokens*. O lugar `R1C3N` receberá o *token* de habilitação quando acaberem os *tokens* de `R1C2D` ou não houverem requisições em `RQC2`. O processo se repete para a classe `C3` e retorna para a classe `C1`.

Cada máquina possui sua própria rotina de escalonamento devido às configurações de cada uma, bem como o volume de requisições dadas condições de trânsito. Realizando uma análise qualitativa, o modelo possui as equações de guarda que evitam que o modelo entre em estado *deadlock*. Na situação em que houvesse *tokens* em `R1C1D`, mas não existissem requisições de classe `C1` no lugar `RQC1`, o sistema ficaria indefinidamente aguardando até

que surgisse alguma requisição de classe `C1`. Mesmo que existissem requisições com menor prioridade aguardando, elas não seriam processadas. Para resolver essa situação, a equação de guarda `Eg5` remove os *tokens* remanescentes `R1C1D` e a habilita a `Eg8` para migrar os *tokens* de `R1C1N` para `R1C2N`. Esse comportamento para evitar *deadlock* é tratado em pares de lugares: (`R1C1D`, `RQC1`), (`R1C2D`, `RQC2`), (`R1C3D`, `RQC3`), (`R2C1D`, `FSQC1`), (`R2C2D`, `FSQC2`) e (`R2C3D`, `FSQC3`).

As equações de guarda estão elencadas na Tabela 10. As métricas de desempenho são as mesmas listadas na Tabela 9.

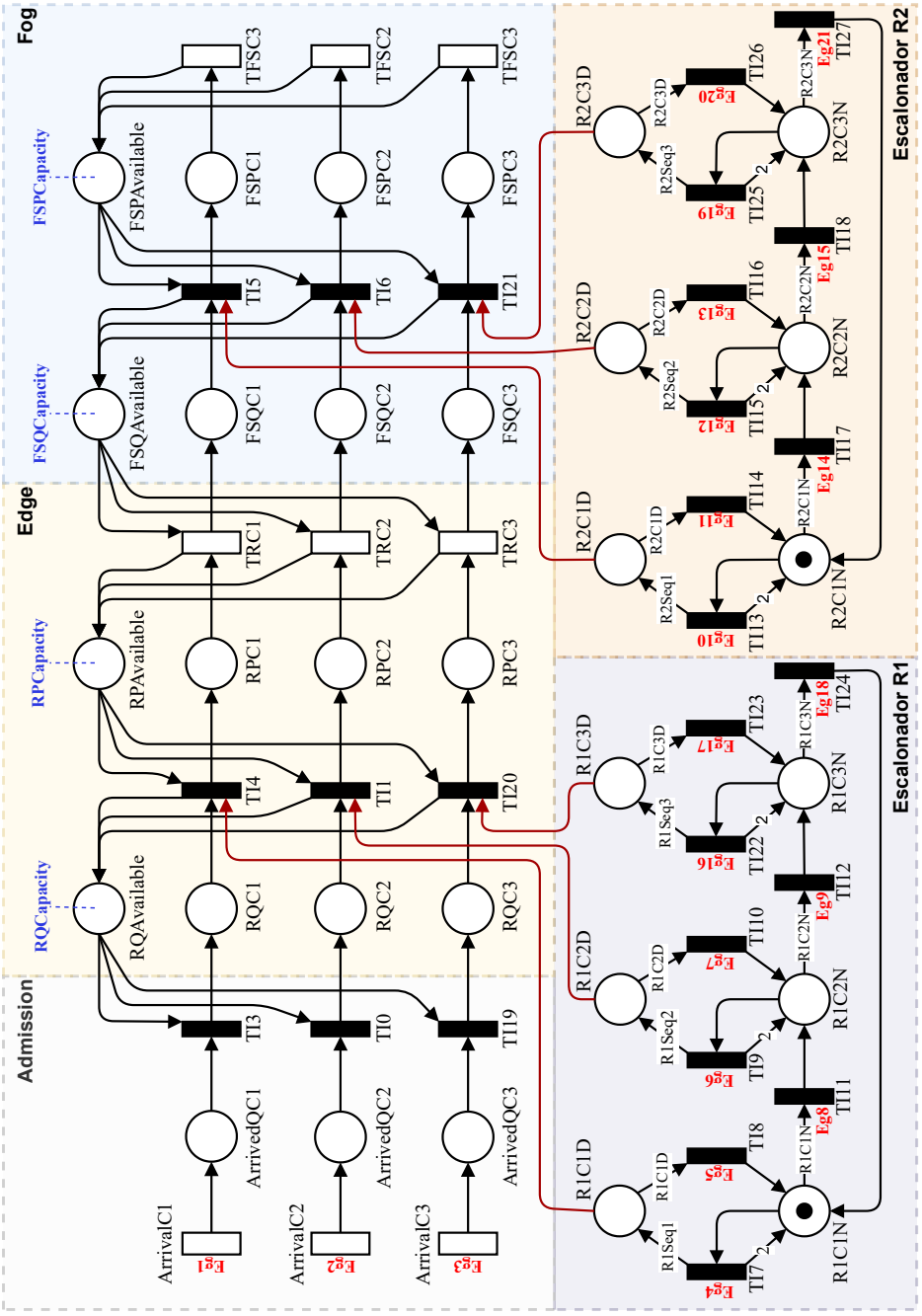


Figura 13 – Modelo contendo três filas de prioridades, usando escalonamento entre as filas

5.3.4 Estudo de Caso 3

O terceiro estudo de caso analisa o desempenho do sistema de filas da Figura 13. Os gráficos da Figura 14 tratam da configuração com 8 núcleos para a RSU deste modelo. No gráfico de utilização da RSU na Figura 14(a), verifica-se que as classes `C2` e `C3` apresentam crescimento mínimo até o limiar de saturação do processamento. Diferente do estudo de caso 2 (Seção 5.3.2), o crescimento inicial é compatível com o escalonamento. Isto quer dizer que em baixas taxas de chegada da classe `C1`, as classes de menor prioridade conseguem maior acesso ao processamento, porque a classe `C1` envia poucas requisições. Comportamento similar é observado na *Fog*, conforme o gráfico da Figura 14(b).

O gráfico da Figura 14(c) mostra a utilização da fila na RSU, onde a classe `C2` alcança o pico próximo de 40% e consequente queda, enquanto que no estudo de caso 2 (Seção 5.3.2), não alcança 10% conforme a Figura 11(c). A classe `C3` tem um crescimento incipiente, mas ainda assim é perceptível. Destaca-se que uma máquina processará um número máximo de requisições de cada classe por ciclo de escalonamento. Assim, o número limitado de processamento por ciclo faz com que a classe `C1` acumule requisições em fila, já que a sua taxa de chegada é maior. A Figura 14(d) mostra a utilização da fila para a *Fog*, que fica a baixo de 0,001% assim como o estudo de caso 2 (Seção 5.3.2). Na Figura 14(e), a probabilidade de descarte é maior para as classes de maior prioridade, por terem taxas de chegada maiores. Na Figura 14(f), o MRT apresenta o mesmo padrão de crescimento para todas as classes, confirmando que o aumento da utilização da fila pela classe `C2`, provoca o aumento do MRT em baixas taxas de chegada.

A Figura 15 mostra o desempenho do sistema para configuração com 20 núcleos na RSU. Devido ao melhor balanceamento entre as máquinas, nas Figuras 14(a) e 14(b) é possível verificar que o desempenho do sistema é similar ao Estudo de Caso 2. É importante observar que a baixa taxa de chegada das classes `C2` e `C3` também influencia na baixa taxa de utilização das filas, conforme as Figuras 14(c) e 14(d). Ressalta-se que a quantidade de requisições que cada classe processará em cada ciclo do escalonamento também interfere no desempenho das classes de menor prioridade. Na Figura 14(e) a probabilidade de descarte cresce proporcional à taxa de chegada. Figura 14(f), a classe `C1` apresenta o MRT que é condizente com o aumento da sua taxa de chegada. Ou seja, quanto maior a fila, maior o tempo que a requisição passa no sistema. Observa-se que o MRT da classe `C2` passa por degradação na performance, seguido por uma recuperação.

5.4 Conclusões Capítulo

Este capítulo apresentou a análise de desempenho de uma arquitetura VANET com duas camadas (*Edge* e *Fog*), onde três modelos SPN e três estudos de caso foram usados para avaliar as métricas MRT, URSU, UFS, UQRSU, UQFS e DP.

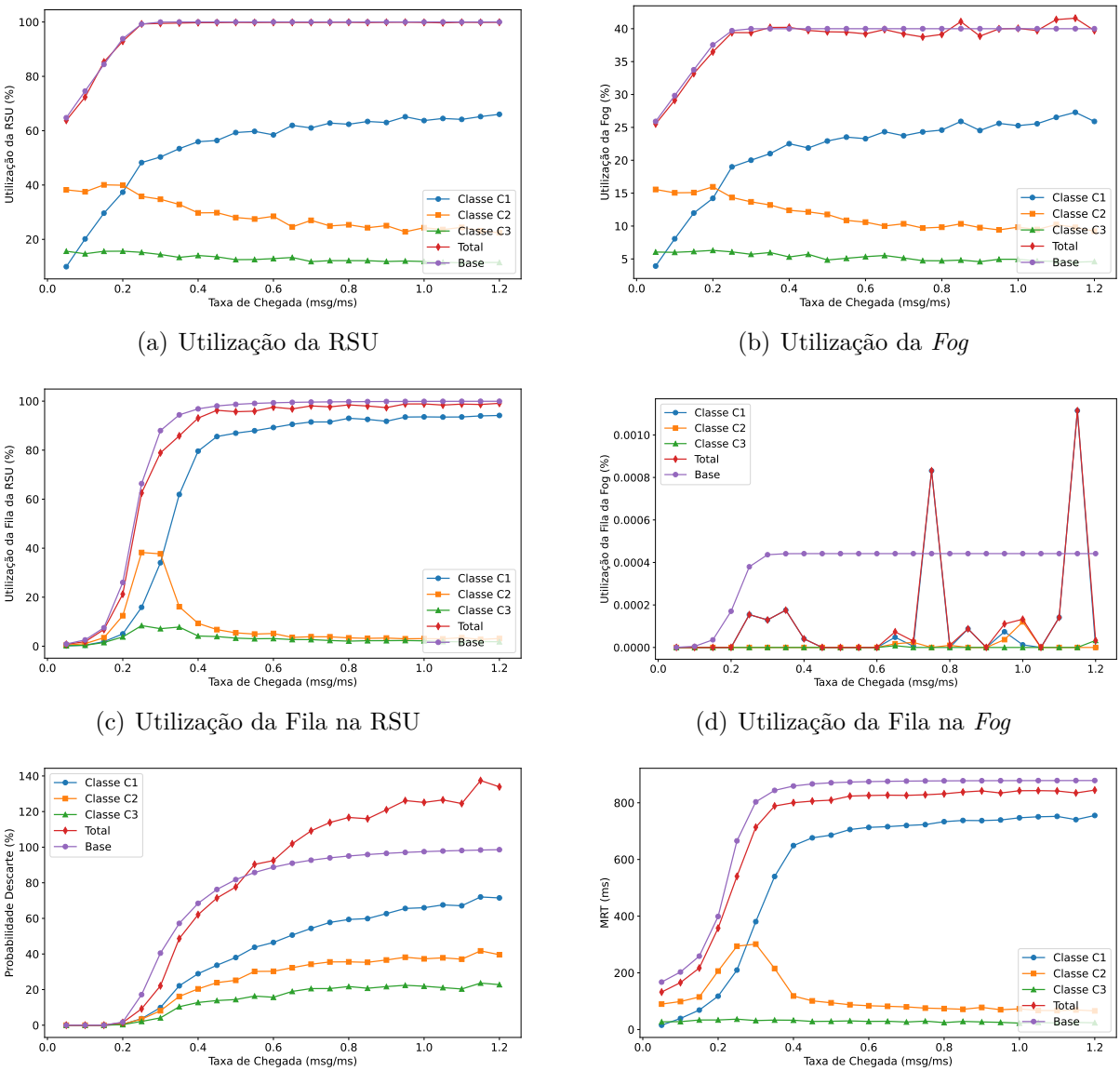
Tabela 10 – Expressões de Guarda

Índice	**Expressão de Guarda**
`Eg1`, `Eg2` e `Eg3`	$ArrivedQ <= 2$
`Eg4`	$(RQC1 > 0)AND(R1C1D = 0)AND(R1C1N = 1)$
`Eg5`	$(RQC1 = 0)AND(R1C1D > 0)$
`Eg6`	$(RQC2 > 0)AND(R1C2D = 0)AND(R1C2N = 1)$
`Eg7`	$(RQC2 = 0)AND(R1C2D > 0)$
`Eg8`	$((R1C1D = 0)AND(R1C1N >= 2))OR(((RQC1 = 0)AND(R1C1N = 1))AND((RQC2 > 0)OR(RQC3 > 0)))$
`Eg9`	$((R1C2D = 0)AND(R1C2N >= 2))OR(((RQC2 = 0)AND(R1C2N = 1))AND((RQC1 > 0)OR(RQC3 > 0)))$
`Eg10`	$(FSQC1 > 0)AND(R2C1D = 0)AND(R2C1N = 1)$
`Eg11`	$(FSQC1 = 0)AND(R2C1D > 0)$
`Eg12`	$(FSQC2 > 0)AND(R2C2D = 0)AND(R2C2N = 1)$
`Eg13`	$(FSQC2 = 0)AND(R2C2D > 0)$
`Eg14`	$((R2C1D = 0)AND(R2C1N >= 2))OR(((FSQC1 = 0)AND(R2C1N = 1))AND((FSQC2 > 0)OR(FSQC3 > 0)))$
`Eg15`	$((R2C2D = 0)AND(R2C2N >= 2))OR(((FSQC2 = 0)AND(R2C2N = 1))AND((FSQC1 > 0)OR(FSQC3 > 0)))$
`Eg16`	$(RQC3 > 0)AND(R1C3D = 0)AND(R1C3N = 1)$
`Eg17`	$(RQC3 = 0)AND(R1C3D > 0)$
`Eg18`	$((R1C3D = 0)AND(R1C3N >= 2))OR(((RQC3 = 0)AND(R1C3N = 1))AND((RQC1 > 0)OR(RQC2 > 0)))$
`Eg19`	$(FSQC3 > 0)AND(R2C3D = 0)AND(R2C3N = 1)$
`Eg20`	$(FSQC3 = 0)AND(R2C3D > 0)$
`Eg21`	$((R2C3D = 0)AND(R2C3N >= 2))OR(((FSQC3 = 0)AND(R2C3N = 1))AND((FSQC1 > 0)OR(FSQC2 > 0)))$

No primeiro estudo de caso foram discutidos dois cenários para um modelo SPN Base. No Cenário 1, a capacidade de processamento da RSU variava e a capacidade da *Fog* permanecia fixa. Verificou-se que a RSU sobrecarrega com baixa demanda de requisições, quando é configurada com poucos núcleos de processamento. Isto ocasiona subutilização da *Fog*. O desempenho do sistema é melhor quando a RSU e *Fog* possuem configurações próximas. A configuração com 20 núcleos na RSU apresenta melhor balanceamento para este cenário. No Cenário 2, a capacidade de processamento da RSU permanece fixa e capacidade da *Fog* varia. Configurações com baixa capacidade deixam a *Fog* sobrecarregada rapidamente e ela se torne um gargalo, fazendo com que a RSU sobrecarregue também, haja vista que precisa manter as requisições enquanto aguarda disponibilidade da *Fog*. Observa-se que a configuração com mais núcleos de processamento subutilizará a *Fog*, pois seu uso efetivo está atrelado à taxa de serviço da RSU, conforme AR'_{Total} da Equação 5.2. Neste cenário a configuração com 20 núcleos apresenta o melhor equilíbrio.

Foi proposto um modelo SPN com três classes de prioridade, onde as classes de maior prioridade bloqueiam as classes de menor prioridade. No segundo estudo de caso foi discutida variação de capacidade na RSU, mantendo fixa a capacidade do servidor da *Fog*. Verificou-se que quanto maior a taxa de chegada das classes prioritárias, pior seria o desempenho das classes de menor prioridade. As classes de menor prioridade passariam mais tempo em fila, o que aumentaria a probabilidade de descarte e elevaria seu MRT.

Para minimizar esse problema, foi proposto um modelo SPN que acrescenta um mecanismo de escalonamento ao modelo com classes de prioridade. Nesta abordagem,



(a) Utilização da RSU

(b) Utilização da *Fog*

(c) Utilização da Fila na RSU

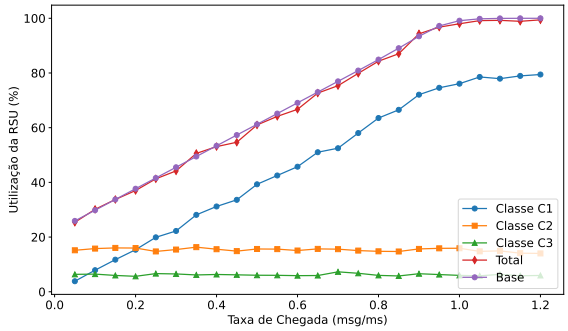
(d) Utilização da Fila na *Fog*

(e) Probabilidade de Descarte

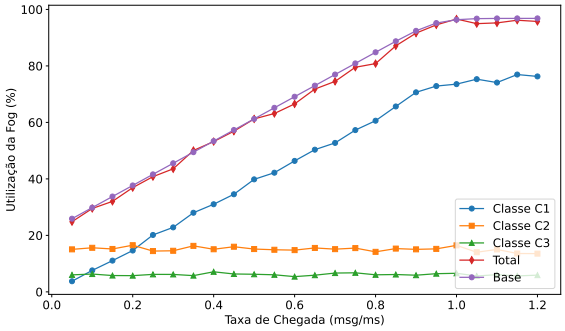
(f) Tempo Médio de Resposta (MRT)

Figura 14 – Resultados da análise Caso 3, para RSU com 8 núcleos

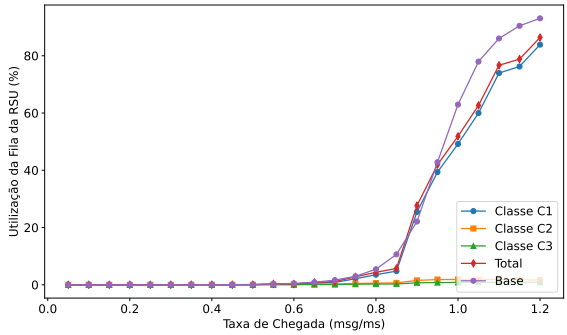
verificou-se que as classes de menor prioridade tiveram menos tempo de fila, bem como seu MRT melhorou, principalmente em configurações com baixa capacidade da RSU. Este estudo aponta que o escalonamento melhora a distribuição da capacidade de processamento. Ressalta-se que a quantidade de requisições que cada classe processará (em um ciclo do escalonamento) deve ser ajustada para otimizar o balanceamento entre as classes. Em um ITS esse ajuste deve ser feito periodicamente ou dinamicamente.



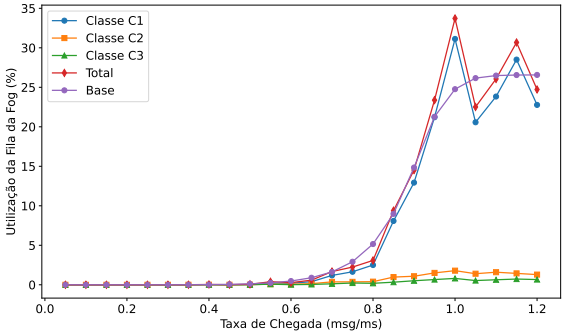
(a) Utilização da RSU



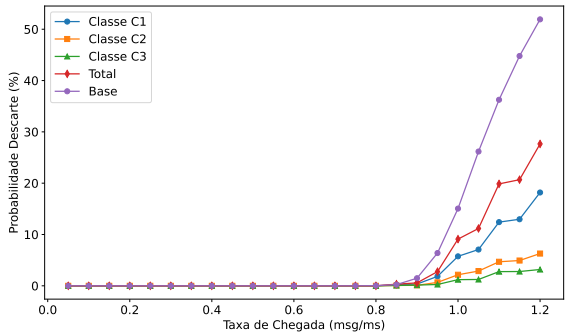
(b) Utilização da *Fog*



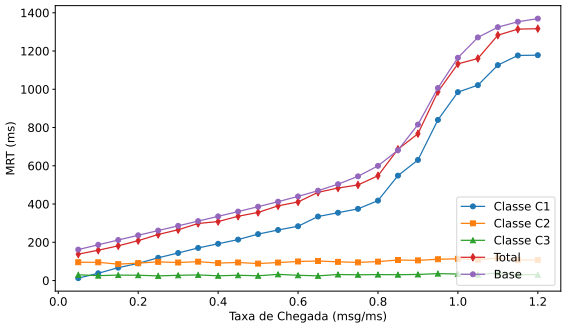
(c) Utilização da Fila na RSU



(d) Utilização da Fila na *Fog*



(e) Probabilidade de Descarte



(f) Tempo Médio de Resposta (MRT)

Figura 15 – Resultados da análise Caso 3, para RSU com 20 núcleos

6 Validação

Este capítulo apresenta a validação do modelo SPN proposto, onde compara os dados de desempenho do modelo, com os dados de um experimento de comunicação entre os componentes em uma rede.

6.1 Processo de Validação

Um modelo SPN representa a simplificação de um conceito ou problema do mundo real. Esses modelos permitem simular diversas situações relacionadas ao problema em análise, bem como estimar o comportamento esperado através de métricas. Os dados obtidos são usados como base para a tomada de decisões, seja quanto ao desenvolvimento de uma nova solução, seja para a adaptação da solução vigente. Tempo de resposta e custo de implementação da solução são fatores que pesam na decisão. Algumas soluções podem usar serviços em nuvem, cujos valores podem ser elevados, mas permitem ajustes para diminuir os custos. Todavia, outras soluções requerem aquisição e implantação de estrutura física.

Em projetos de grande escala o custo de implementação pode ser proibitivo. Mesmo com dados adquiridos a partir das simulações, surge a necessidade de validar as informações obtidas, para mitigar os riscos e custos envolvidos. Implementar uma solução em pequena escala envolve menores custos, ao passo que permite comparar com as métricas obtidas e validar o modelo. Construir um modelo simplificado também é outra possibilidade, pois permite validar com poucos recursos software e hardware. Assim, a validação consiste em criar um experimento para determinar se um modelo SPN representa minimamente o comportamento esperado de sistema real ou simulado.

A Figura 16 apresenta o fluxograma que descreve a estratégia usada nesta validação para assegurar que o experimento seja validado corretamente:

- Montagem do experimento conforme o modelo original: Esta etapa consiste em configurar o experimento conforme a arquitetura base do modelo. O experimento refletirá informações oriundas do modelo, tais como tamanhos de filas, capacidades de processamento, tempos de serviço, dentre outros. Esta etapa também envolve a configuração de computadores, máquinas virtuais e eventualmente o modelo terá maior complexidade, o que inviabilizará a execução do experimento. Desta forma é necessário simplificar o modelo proposto.

- Simplificação do modelo: Verificando-se a inviabilidade de gerar o experimento

a partir do modelo original, deriva-se um novo modelo que encapsular algumas características, mas mantém a essência da arquitetura. Desta forma, reduz-se a quantidade de componentes replicáveis apresentados no cenário e no modelo original. Devido à restrição de recursos computacionais, a simplificação do modelo permitirá que o experimento reflita componentes, parâmetros de quantidade e de processamento. Do ponto de vista estatístico, validar uma parte do modelo validará também o todo.

- Sincronização dos relógios: As máquinas físicas ou virtuais precisam de sincronia, pois minimiza erros no cálculo do MRT. Máquinas virtuais podem ser sincronizadas usando uma ferramenta *Network Time Protocol* (NTP) (MILLS, 1991). Máquinas virtuais hospedadas em serviços de *cloud* utilizam os mecanismos de sincronia providos pelos serviços. A *Amazon Web Services* (AWS) disponibiliza o Serviço de Sincronização Temporal da Amazon local [1], que usa o NTP.

- Execução do experimento com baixa carga (0 drop): Esta etapa avalia se a comunicação entre os componentes ocorre conforme esperado. As requisições são enviadas com baixa taxa de chegada, pois minimiza o tempo de fila, bem como a perda de dados ou eventos (0 drop).

- Coleta dos tempos intermediários do experimento: A execução do experimento coleta os Tempos Intermediários Externos (TIEs) entre os componentes. Um tempo intermediário é a diferença entre o momento que requisição sai de um componente e chega em outro componente. Os Tempos Intermediários Internos (TIIs) ao componente correspondem aos processos internos prévios ao efetivo processamento.

- Alimentação do modelo SPN com os tempos intermediários do experimento: Os TIEs e TIIs obtidos alimentarão o modelo, seja o original, seja o simplificado. Os TIIs fazem parte do tempo de resposta do sistema, mesmo que sejam valores baixos para serem descartados no contexto do modelo. Valores elevados de TIIs podem indicar gargalo no modelo ou no experimento. Nesse caso é importante reavaliar a etapa "Simplificação do modelo".

- Execução do experimento com variação de capacidade de processamento: Cada componente é configurado com até quatro instâncias de processamento paralelo, conforme o modelo SPN. No experimento, o componente alvo terá variação de 1 e 4 instâncias de processamento. Nesta etapa o experimento é executado com alta carga, onde o MRT e o Desvio Padrão são coletados. O experimento realiza quatro ciclos de envio, de forma que em cada ciclo a capacidade de processamento é incrementada no componente alvo.

[1] https://docs.aws.amazon.com/pt_br/AWSEC2/latest/UserGuide/configure-ec2-ntp.html

- Execução do modelo SPN com os mesmos parâmetros de processamento e capacidade do experimento: O modelo simplificado é configurado para ser executado sob as mesmas condições de carga (taxa de chegada). Nesta etapa também são coletados o MRT e Desvio Padrão correspondente para cada variação na capacidade de processamento.

- Cálculo do p-value em relação aos MRTs (experimento e modelo): A partir dos dados obtidos da simulação e do experimento, calcula-se o p-value entre os MRTs em cada cenário, afim de que essa confiança estatística seja maior/igual a 0,05.

- Verificação estatística dos MRTs: Verificando-se que os MRTs são estatisticamente iguais, o processo é finalizado com sucesso. Caso contrário, retorna-se à etapa de coleta dos tempos intermediários do experimento para realizar novos ajustes no modelo.

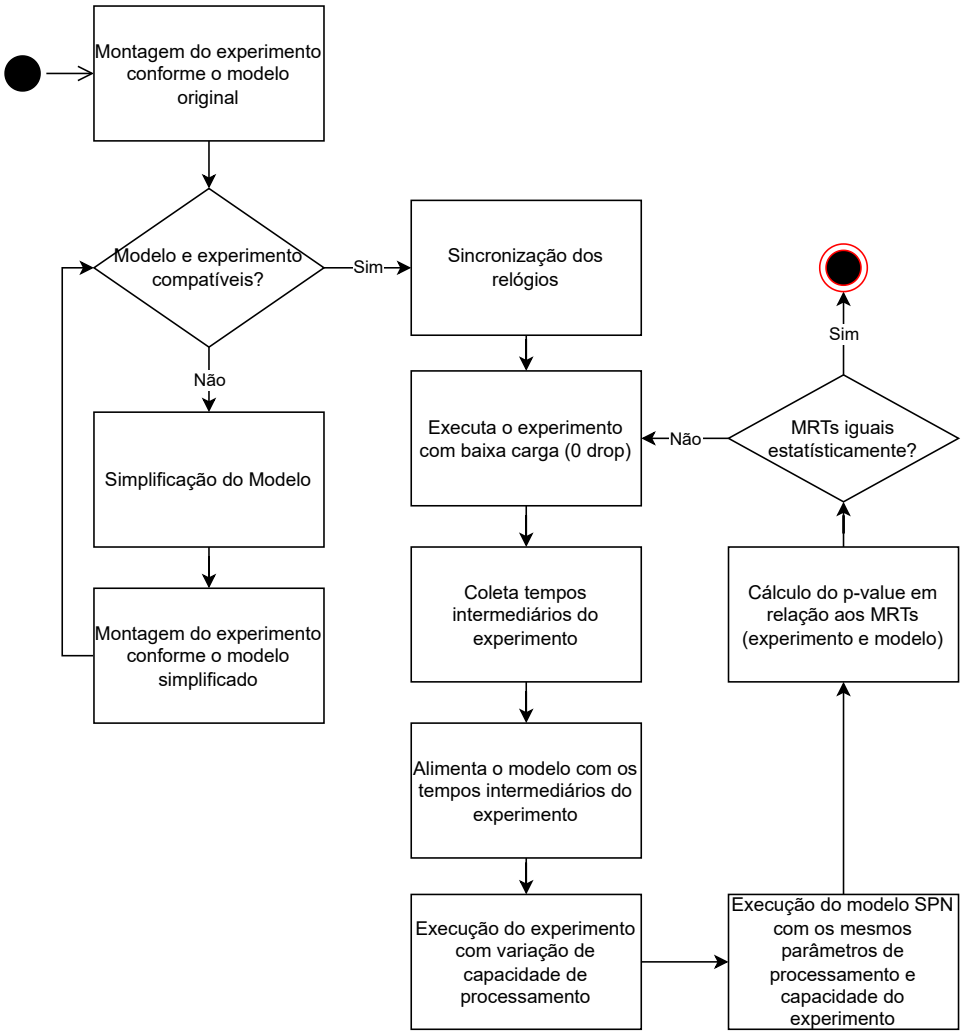


Figura 16 – Fluxo de Validação

6.2 Modelo, Ambiente do Experimento e Ferramenta de Validação

6.2.1 Modelo

A Figura 17 apresenta o modelo SPN simplificado, que foi derivado do modelo base da Figura 7. Diferente do modelo base, o modelo simplificado utiliza três camadas ao invés de duas para se assemelhar a um sistema com processamento na *Cloud*. Na camada de admissão, as três entradas de dados originais foram unificadas, pois as classes `C2` e `C3` possuíam taxas de chegada constantes. Na admissão foi utilizada uma transição determinística, pois o processo de validação é produto da comparação com números absolutos definidos em experimento, ou seja, não levando-se em conta distribuição estatística de valores.

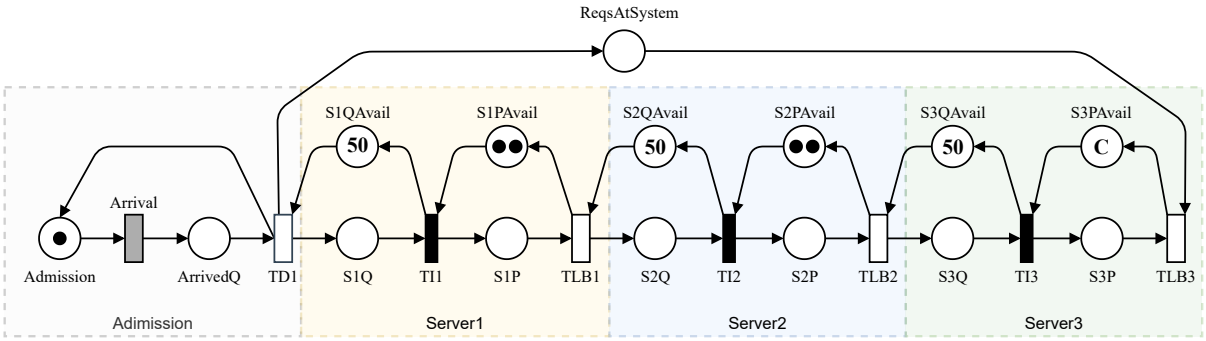


Figura 17 – Modelo SPN da Validação

Conforme o fluxograma da Figura 16, o experimento foi executado visando encontrar variações significativas do MRT. A calibragem do experimento envolve ajustes em diversos parâmetros como tamanho das filas, quantidade de elementos de processamento e tempos de processamento em cada componente. Ressalta-se que a montagem do experimento envolve previamente a estruturação de equipamentos físicos, virtuais, rede de comunicação e quaisquer outros recursos necessários. Após essa calibragem, os tempos coletados alimentaram o modelo simplificado. As simulações do modelo coletam os valores de MRT. Os dados coletados do experimento e da simulação são submetidos à verificação estatística. O fluxo se repetiu até que os valores da Tabela 11 fossem definidos.

Tabela 11 – Principais transições temporizadas utilizadas no modelo SPN da Figura 17

Elemento	Descrição	Valor
Arrival	Intervalo de tempo entre a chegada de requisições	300 ms
TD1	Tempo associado às transmissão da admissão até o Server 1	1 ms
TLB1	Tempo associado ao processamento de uma requisição no Server1	100 ms
TLB2	Tempo associado ao processamento de uma requisição no Server2	100 ms
TLB3	Tempo associado ao processamento de uma requisição no Server3	3000 ms

6.2.2 Ambiente do Experimento

O ambiente de experimento foi montando na plataforma de nuvem AWS [2]. O ambiente é composto por quatro máquinas virtuais *Amazon Elastic Compute Cloud* (Amazon EC2) [3], do tipo `t2.xlarge`, que possuem 4 *vcpu*. As instâncias Amazon EC2 usam sistema operacional Ubuntu, onde as demais características de uma instância estão listadas na Tabela 12. A Figura 18 apresenta a arquitetura do experimento que usou um ambiente *Amazon Virtual Private Cloud* (Amazon VPC) [4], englobando todas as instâncias Amazon EC2. O uso de um ambiente único facilita o gerenciamento dos recursos, reduz a latência e minimiza os riscos de incompatibilidade. Uma instância Amazon EC2 hospeda a aplicação `Source.jar`, responsável pela transmissão das requisições. As demais instâncias hospedam uma aplicação (cada uma) um `LoadBalancer.jar`, onde o primeiro recebe a requisição que segue em cadeia até o último `LoadBalancer`. Após o último nó da cadeia processar a requisição, a resposta é devolvida diretamente para o `Source`, que calcula a métrica de MRT.

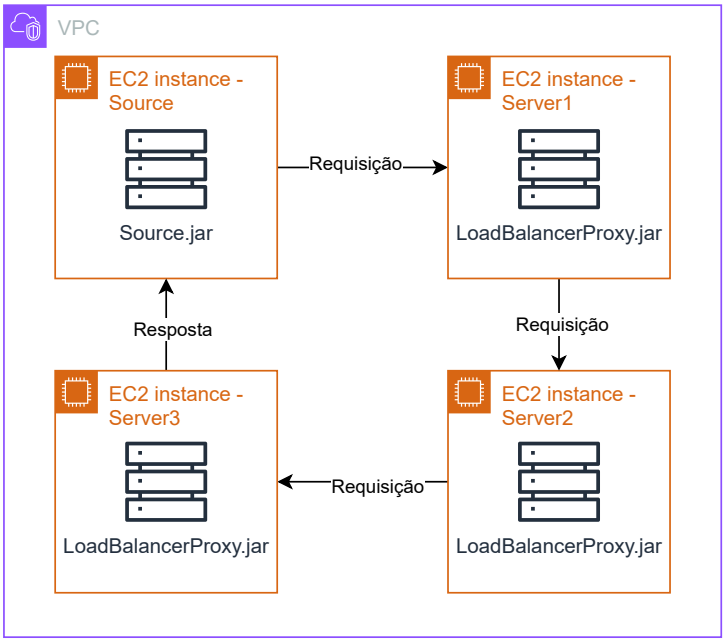


Figura 18 – Ambiente na AWS

Tabela 12 – Configuração do experimento no ambiente da AWS

Elemento	Descrição
Sistema Operacional	Ubuntu 24.04.1 LTS
Memória	15Gi
CPU	Intel(R) Xeon(R) CPU E5-2686 v4 @ 2.30GHz

[2] https://aws.amazon.com/pt/what-is-aws/
[3] https://aws.amazon.com/pt/ec2/
[4] https://aws.amazon.com/pt/vpc/

6.2.3 Ferramenta de Validação

O experimento de validação utilizou o software *PASID Validator*. O projeto do software foi desenvolvido usando a linguagem Java na versão 8, mas já foi atualizado para a versão Java 11. Essa ferramenta foi concebida no laboratório PASID, onde seu objetivo é certificar os resultados oriundos dos modelos em SPN. O *PASID Validator* permite configurar parâmetros de execução e de comunicação em rede, conforme o modelo SPN em avaliação.

O *PASID Validator* consistem em dois artefatos executáveis que são colocados no ambiente de rede do experimento: `Source.jar` e `LoadBalancerProxy.jar`. O artefato `Source` é responsável por enviar as requisições, receber a resposta e calcular as métricas ao final do experimento. O `LoadBalancerProxy` recebe as requisições, enfileira e distribui entre seus `ServiceProxy`. Cada `ServiceProxy` é responsável por processar uma requisição e encaminhar para o próximo `LoadBalancer`. O último `LoadBalancer` da cadeia processa a requisição (via `ServiceProxy`) e devolve a resposta para o `Source`. Os artefatos (`Source` e `LoadBalancerProxy`) se comunicam via *sockets* TCP/IP. Os endereços e portas de cada artefato são configurados durante a montagem do experimento. Cada artefato conhece o endereço e a porta do próximo elemento da cadeia. O uso de *sockets* permite a comunicação direta e contínua entre os elementos do experimento, garantindo uma comunicação confiável.

O `Source` envia requisições em um intervalo de tempo definido durante a configuração. As requisições são enviadas ao primeiro `LoadBalancerProxy`, que distribui para o `ServiceProxy` escolhido através um algoritmo de fila circular. Quando a fila de admissão está lotada, as requisições que chegam são descartadas. O `ServiceProxy` realiza um pseudo processamento da requisição, que se dá por meio de uma pausa chamando comando `Thread.sleep()`. O tempo de espera é dado pelo tempo médio de serviço, que especificado durante a configuração. Quando finaliza, o `ServiceProxy` encaminha para o próximo elemento da rede. Os demais `LoadBalancerProxy` funcionam de forma semelhante, exceto pelo último, que devolve a resposta para o `Source`. Este calcula o tempo de resposta da requisição, que será usado para calcular o MRT do experimento. O *timestamp* de saída de uma máquina e de chegada na próxima máquina, é concatenado ao corpo da requisição. Essa estratégia permite conhecer a rota da requisição e identificar possíveis gargalos.

6.2.4 Cenário de Validação

Neste cenário os `LoadBalancerProxy` foram configurados conforme a Tabela 13. Ressalta-se que o termo `ServiceProxy` é uma abstração para representar um elemento de processamento na capacidade dos `LoadBalancerProxy`. De acordo com o contexto em estudo, esse elemento pode significar núcleo, *thread*, contêiner, dentre outros elementos. Da mesma forma, o `LoadBalancerProxy` pode representar um servidor, um balanceador, um orquestrador de contêineres, dentre outros. A estratégia de descarte da fila é *drop* ao

atingir o limite estabelecido para o tamanho da fila, que é de 50 em cada *load balancer*. A estratégia de fila usada nos *load balancers* é FCFS. Em cada configuração, a quantidade de serviços no último *load balancer* é incrementada em 1. Em cada configuração são enviadas 500 requisições, onde o tempo de resposta de cada mensagem respondida será usado para calcular o MRT e o desvio padrão da configuração. O modelo SPN foi simulado com taxa de erro de 2%, onde o MRT e desvio padrão foram coletados para análise estatística.

Tabela 13 – Parâmetros para validação

Configuração	LoadBalancer1	LoadBalancer2	LoadBalancer3
A	2 Services	2 Services	1 Service
B	2 Services	2 Services	2 Services
C	2 Services	2 Services	3 Services
D	2 Services	2 Services	4 Services

6.3 Resultados

O gráfico da Figura 19 apresenta o comparativo entre os MRTs do experimento e modelo SPN. Em cada configuração, os valores obtidos no experimento são similares aos obtidos no modelo, seguindo a tendência de diminuição conforme a quantidade de *services* aumenta. Observa-se que os valores do experimento apresentam maior variação em relação ao modelo, o que é evidenciado pela ampla faixa do desvio padrão. Essa faixa de variação também diminui com o acréscimo de *services*.

Comparamos os MRTs do modelo e experimento aplicando o T-Test de uma amostra. O *T-Test* é um tipo de teste estatístico usado para comparar as médias de dois grupos, ele pode ser usado quando as amostras satisfazem as condições de normalidade, igual variância e independência (**??**). Os resultados estatísticos estão elencados na Tabela 14. Em cada configuração do experimento foram enviadas 500 requisições, resultando no cálculo do MRT e desvio padrão. O modelo resultou em um valor único da métrica, bem como o seu desvio padrão. Todas as amostras apresentaram distribuição normal.

O valor p é superior a 0,05 em todas as configurações analisadas. Nestes casos, consideramos que não há diferença significativa entre os dois grupos amostrais e, portanto, não podemos refutar a hipótese nula de igualdade nos casos analisados, com 95% de confiança. Considerando o modelo como referência, a análise sugere que o experimento é capaz de reproduzir os valores de referência com precisão razoável, mas apresenta desafios em termos de variabilidade. O foco principal para melhorias deve ser na redução da variabilidade experimental e na calibração para valores menores.

Por fim, esta validação conclui que os resultados gerados pelo modelo são estatisticamente equivalentes ao experimento. O modelo simplificado reflete o ambiente real, o

que também valida o modelo base. O resultado ratifica a utilização do modelo SPN para planejar e avaliar a arquitetura proposta neste trabalho.

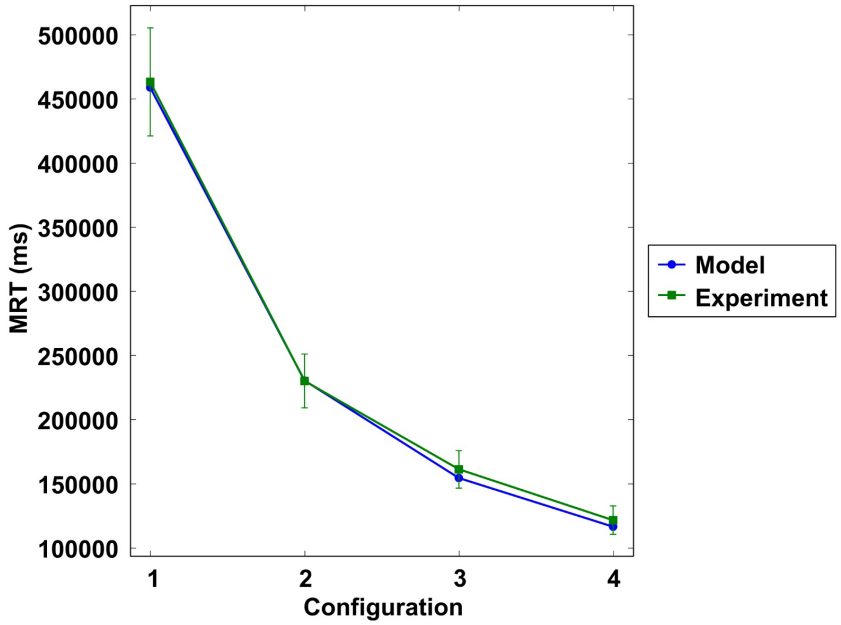


Figura 19 – Resultado da Validação

Conf	MRT Modelo	Desv Modelo	MRT Experim	Desv Experim	t Value	p Value
1	459102.48	1521.20	463489.53	42135.41	-0.47	0.67
2	230696.34	748.49	230384.88	20944.08	0.07	0.95
3	154770.21	502.23	161477.93	14679.81	-2.04	0.11
4	116795.30	382.33	121858.38	11078.03	-2.04	0.11

Tabela 14 – Resultado Test-T

7 Conclusões

Dada a complexidade envolvida no processamento de múltiplas requisições prioritárias dentro da infraestrutura VANET, o modelo proposto com escalonamento oferece uma abordagem coesa para a avaliação de desempenho. Ao considerar três classes de requisições (C1, C2 e C3) e suas respectivas filas de prioridade, o sistema é capaz de gerenciar adequadamente a alocação de recursos sem que haja preempção entre os processos em execução. Ao tempo que evita o bloqueio das classes de menor prioridade, garantindo que uma quantidade mínima de requisições seja atendida em cada classe. A arquitetura demonstra uma organização eficiente, garantindo que as camadas sejam capazes de lidar com o fluxo de mensagens mesmo em cenários de alta demanda, otimizando o tempo de resposta e minimizando a probabilidade de descarte. Com base nos experimentos realizados, é possível concluir que o balanceamento entre as capacidades das *Edge* e *Fog* tem papel central no desempenho geral, destacando a necessidade de uma configuração adequada dos núcleos de processamento. O ajuste apropriado do escalonamento também influencia em como as classes utilizarão a fila e o processamento de cada máquina.

O Estudo de Caso 1 (Cenário 1) mostrou que configurações com 8, 12 e 16 núcleos de processamento na RSU, ocasionam menor utilização no servidor da camada de *Fog*. Essa observação permite propor que equipamentos de menor capacidade sejam instalados em vias de menor fluxo de veículos. Ao passo que um servidor de *Fog* seja responsável por este grupo de RSU. É importante ressaltar que mesmo nestes casos, a quantidade de dados enviados para a *Fog* não deve superar sua capacidade de processamento. Caso isto ocorra, a situação será semelhante ao que ocorre no Cenário 2, onde a *Fog* é o elemento de menor capacidade e se transforma no gargalo do sistema.

A adaptação do modelo SPN para contemplar mais classes de prioridade é um dos fatores limitantes. Considerando apenas uma máquina em cada camada, para cada nova classe de prioridade será necessário acrescentar 9 lugares, 3 transições temporizadas e 9 transições imediatas, totalizando 21 novos elementos. Para incluir uma nova RSU paralela à primeira, seria necessário incluir 17 lugares, 6 transições temporizadas e 15 transições imediatas, totalizando 38 lugares, onde neste caso a maior quantidade se deve ao fato da RSU precisar de um escalonador próprio. Incluir tantos elementos para uma única nova classe ou RSU torna a escalabilidade inviável para este modelo SPN. A análise qualitativa de muitos elementos requer maior atenção, pois é necessário garantir que o modelo apresente o comportamento esperado, bem como não caia em situações de *deadlock*. A adição de muitos elementos também penaliza a análise quantitativa, pois requer maior tempo de simulação. Ajustes nos parâmetros da ferramenta de simulação afetam diretamente a precisão dos resultados.

A análise do problema via modelos SPN só é vantajosa até determinado limite de classes de prioridade ou de máquinas. A inclusão de requisitos como segurança e QoS podem exigir outras formas de analisar e simular o problema. Assim, outras ferramentas de simulação podem ser mais apropriadas para tratar a escalabilidade inerente do problema.

Os resultados obtidos com a análise do modelo base devem ser observados como parâmetro no planejamento de dimensionamento dos componentes reais. Portanto, este trabalho pode auxiliar os administradores de sistemas a identificarem as configurações mais adequada em termos de custo e eficiência.

Na validação apresentada, apesar de ter sido sobre uma simplificação do modelo base (ainda bem próximo do modelo original), foi possível observar com certo grau de confiança estatística que o resultado do modelo simplificado é semelhante ao resultado do experimento e, portanto validando a utilização do modelo base para os devidos fins propostos. A ferramenta de validação apresenta resultados promissores, ainda que em sua fase inicial de desenvolvimento. O acréscimo de novas funcionalidades permitirá que os modelos em avaliação precisem menos ajustes.

7.1 Trabalhos Futuros

A seguir listamos alguns possíveis trabalhos futuros:

- *Análise de desempenho usando preempção:* Aprimorar o modelo para avaliar o desempenho do sistema usando técnicas de preempção.
- *Análise da comunicação e disponibilidade de RSUs:* Avaliar como diversas RSUs são afetadas quando há sobrecarga do sistema por eventual queda de outras RSUs.
- *Cooperação e balanceamento entre RSUs:* Avaliar como as RSUs podem cooperar para balancear a carga vinda dos veículos.

Referências

AHMED, I. Z.; MOHAMED, T. M.; SADEK, R. A. Challenges of vehicular ad hoc networks: Egypt perspective. In: *Second International Conference for Computing and Informatics, ICCI*. [S.l.: s.n.], 2017. Citado na página 10.

AHMED, I. Z.; MOHAMED, T. M.; SADEK, R. A. A low computation message delivery and authentication vanet protocol. In: IEEE. *2017 12th International Conference on Computer Engineering and Systems (ICCES)*. [S.l.], 2017. p. 204–211. Citado 2 vezes nas páginas 11 e 12.

ATWA, R. J.; FLOCCHINI, P.; NAYAK, A. Rteam: Risk-based trust evaluation advanced model for vanets. *IEEE Access*, IEEE, v. 9, p. 117772–117783, 2021. Citado 3 vezes nas páginas 23, 24 e 27.

BALZANO, W.; STRANIERI, S. Data dissemination in vehicular ad hoc network: a model to improve network congestion. In: SPRINGER. *Web, Artificial Intelligence and Network Applications: Proceedings of the Workshops of the 33rd International Conference on Advanced Information Networking and Applications (WAINA-2019) 33*. [S.l.], 2019. p. 851–859. Citado na página 10.

BELLAVISTA, P. et al. A survey on fog computing for the internet of things. *Pervasive and mobile computing*, Elsevier, v. 52, p. 71–99, 2019. Citado 2 vezes nas páginas 9 e 10.

BOUKHALFA, F. et al. Performance evaluation of an active signaling based time-slot scheduling scheme for connected vehicles. *Annals of Telecommunications*, Springer, v. 76, p. 363–374, 2021. Citado 6 vezes nas páginas 1, 10, 22, 24, 25 e 26.

BRIEF, I. L. Internet of things. 2019. Citado na página 7.

CASANOVA, H. et al. Simgrid: a sustained effort for the versatile simulation of large scale distributed systems. *arXiv preprint arXiv:1309.1630*, 2013. Citado na página 2.

CHEN, S.; ZHANG, T.; SHI, W. Fog computing. *IEEE Internet Computing*, IEEE, v. 21, n. 2, p. 4–6, 2017. Citado na página 9.

CUNHA, B. et al. Smart traffic control in vehicle ad-hoc networks: a systematic literature review. *International Journal of Wireless Information Networks*, Springer, v. 28, n. 3, p. 362–384, 2021. Citado na página 10.

DOBRE, C.; POP, F.; CRISTEA, V. A simulation framework for dependable distributed systems. In: IEEE. *2008 International Conference on Parallel Processing-Workshops*. [S.l.], 2008. p. 181–187. Citado na página 2.

FENABRAVE. *Dados de Mercado Fenabrave Ed. 252 - Emplacamentos em 2023*. 2024. Disponível em: <https://www.fenabrave.org.br/portal/files/2023_12_2.pdf>. Citado na página 1.

GAO, H. et al. V2vr: reliable hybrid-network-oriented v2v data transmission and routing considering rsus and connectivity probability. *IEEE Transactions on Intelligent*

Transportation Systems, IEEE, v. 22, n. 6, p. 3533–3546, 2020. Citado 5 vezes nas páginas 10, 11, 23, 24 e 26.

GERMAN, R. *Performance analysis of communication systems - modelling with non-Markovian stochastic Petri nets*. [S.l.]: Wiley, 2000. (Wiley-Interscience series in systems and optimization). ISBN 978-0-471-49258-0. Citado na página 12.

GERMAN, R. *Performance analysis of communication systems with non-Markovian stochastic Petri nets*. [S.l.]: John Wiley & Sons, Inc., 2000. Citado na página 14.

GHOSH, S.; MISRA, I. S.; CHAKRABORTY, T. Optimal rsu deployment using complex network analysis for traffic prediction in vanet. *Peer-to-Peer Networking and Applications*, Springer, v. 16, n. 2, p. 1135–1154, 2023. Citado 5 vezes nas páginas 1, 22, 24, 25 e 27.

HIROFUCHI, T.; LEBRE, A.; POUILLOUX, L. Simgrid vm: Virtual machine support for a simulation framework of distributed systems. *IEEE Transactions on Cloud Computing*, IEEE, v. 6, n. 1, p. 221–234, 2015. Citado na página 2.

HOTA, L. et al. A performance analysis of vanets propagation models and routing protocols. *Sustainability*, MDPI, v. 14, n. 3, p. 1379, 2022. Citado 7 vezes nas páginas 1, 2, 10, 11, 21, 23 e 26.

ISMAIL, N. et al. Enhanced congestion control model based on message prioritization and scheduling mechanism in vehicle-to-infrastructure (v2i). In: IOP PUBLISHING. *Journal of Physics: Conference Series*. [S.l.], 2022. v. 2312, n. 1, p. 012087. Citado 3 vezes nas páginas 1, 3 e 10.

JABALLAH, W. B.; CONTI, M.; LAL, C. A survey on software-defined vanets: benefits, challenges, and future directions. *arXiv preprint arXiv:1904.04577*, 2019. Citado 2 vezes nas páginas 2 e 11.

JAIN, R. *The art of computer systems performance analysis*. [S.l.]: john wiley & sons, 1990. Citado 5 vezes nas páginas 15, 16, 17, 18 e 19.

KASSIR, S. et al. An analytical model and performance evaluation of multihomed multilane vanets. *IEEE/ACM Transactions on Networking*, IEEE, v. 29, n. 1, p. 346–359, 2020. Citado 4 vezes nas páginas 10, 11, 21 e 24.

KHAN, S. et al. Security challenges of location privacy in vanets and state-of-the-art solutions: A survey. *Future Internet*, MDPI, v. 13, n. 4, p. 96, 2021. Citado 2 vezes nas páginas 11 e 12.

KHAN, W. Z. et al. Edge computing: A survey. *Future Generation Computer Systems*, Elsevier, v. 97, p. 219–235, 2019. Citado 2 vezes nas páginas 8 e 9.

LEA, P. *Internet of Things for Architects: Architecting IoT solutions by implementing sensors, communication infrastructure, edge computing, analytics, and security*. [S.l.]: Packt Publishing Ltd, 2018. Citado na página 7.

LIM, K. L. et al. State of data platforms for connected vehicles and infrastructures. *Communications in transportation research*, Elsevier, v. 1, p. 100013, 2021. Citado 2 vezes nas páginas 1 e 3.

LIMA, J. W. S. d.; CALLOU, G.; ANDRADE, E. Queueing theory and stochastic petri net: A tutorial. *Research, Society and Development*, v. 10, n. 3, p. e2810312826, Mar. 2021. Disponível em: <https://rsdjournal.org/index.php/rsd/article/view/12826>. Citado na página 12.

LITTLE, J. D. A proof for the queuing formula: L= λ w. *Operations research*, INFORMS, v. 9, n. 3, p. 383–387, 1961. Citado na página 35.

MACIEL, P. R. M. *Performance, reliability, and availability evaluation of computational systems, Volume 2: Reliability, availability modeling, measuring, and data analysis*. [S.l.]: CRC Press, 2023. Citado na página 12.

MACIEL, P. R. M. *Performance, reliability, and availability evaluation of computational systems, volume I: performance and background*. [S.l.]: CRC Press, 2023. Citado na página 12.

MAHMOOD, D. A.; HORVÁTH, G. Analysis of the message propagation speed in vanet with disconnected rsus. *Mathematics*, MDPI, v. 8, n. 5, p. 782, 2020. Citado 5 vezes nas páginas 10, 21, 24, 25 e 27.

MARSAN, M. A. et al. *Modelling with Generalized Stochastic Petri Nets*. USA: John Wiley & Sons, Inc., 1994. ISBN 0471930598. Citado na página 12.

MARSAN, M. A. et al. Modelling with generalized stochastic petri nets. *ACM SIGMETRICS performance evaluation review*, ACM New York, NY, USA, v. 26, n. 2, p. 2, 1998. Citado na página 15.

MAZHAR, T. et al. Quality of service (qos) performance analysis in a traffic engineering model for next-generation wireless sensor networks. *Symmetry*, MDPI, v. 15, n. 2, p. 513, 2023. Citado na página 3.

MILLS, D. L. Internet time synchronization: the network time protocol. *IEEE Transactions on communications*, Ieee, v. 39, n. 10, p. 1482–1493, 1991. Citado na página 54.

MOLLOY, M. K. Performance analysis using stochastic petri nets. *IEEE Trans. Comput.*, IEEE Computer Society, Washington, DC, USA, v. 31, p. 913–917, September 1982. ISSN 0018-9340. Citado na página 12.

MOUSA, R. J.; HUSZÁK, Á.; SALMAN, M. A. Enhancing vanet connectivity through a realistic model for rsu deployment on highway. In: IOP PUBLISHING. *Journal of Physics: Conference Series*. [S.l.], 2021. v. 1804, n. 1, p. 012104. Citado 4 vezes nas páginas 11, 21, 25 e 27.

MURATA, T. Petri nets: Properties, analysis and applications. *Proceedings of the IEEE*, IEEE, v. 77, n. 4, p. 541–580, 1989. Citado na página 12.

NIDHI; LOBIYAL, D. Performance evaluation of rsus deployment at dense intersections. *International Journal of Information Technology*, Springer, v. 13, n. 3, p. 1095–1099, 2021. Citado 4 vezes nas páginas 22, 24, 25 e 28.

OLIVEIRA, D. et al. Advanced stochastic petri net modeling with the mercury scripting language. In: *Proceedings of the 11th EAI International Conference on Performance Evaluation Methodologies and Tools*. [S.l.: s.n.], 2017. p. 192–197. Citado na página 33.

PAUL, A.; JEYARAJ, R. Internet of things: A primer. *Human Behavior and Emerging Technologies*, Wiley Online Library, v. 1, n. 1, p. 37–47, 2019. Citado na página 7.

PETERSON, J. L. Petri nets. *ACM Computing Surveys (CSUR)*, ACM New York, NY, USA, v. 9, n. 3, p. 223–252, 1977. Citado na página 12.

RENASET. *Registro Nacional de Sinistros e Estatísticas de Trânsito em 2023*. 2024. Disponível em: <https://www.gov.br/transportes/pt-br/assuntos/transito/arquivos-senatran/docs/renaest>. Citado na página 1.

RITO, P. et al. Aveiro tech city living lab: a communication, sensing, and computing platform for city environments. *IEEE Internet of Things Journal*, IEEE, v. 10, n. 15, p. 13489–13510, 2023. Citado na página 2.

RODRIGUES, L. et al. Performance and availability evaluation of an smart hospital architecture. *Computing*, Springer, v. 103, p. 2401–2435, 2021. Citado na página 36.

RODRIGUES, L. et al. Performance evaluation of smart cooperative traffic lights in vanets. *International Journal of Computational Science and Engineering*, Inderscience Publishers (IEL), v. 24, n. 3, p. 276–289, 2021. Citado 4 vezes nas páginas 22, 25, 27 e 28.

SADIKU, M. N.; MUSA, S. M.; MOMOH, O. D. Cloud computing: opportunities and challenges. *IEEE potentials*, IEEE, v. 33, n. 1, p. 34–36, 2014. Citado na página 8.

SANTNER, T. *The Design and analysis of computer experiments*. [S.l.]: Springer, 2003. Citado 2 vezes nas páginas 15 e 16.

SANTOS, L. et al. Data processing on edge and cloud: a performability evaluation and sensitivity analysis. *Journal of Network and Systems Management*, Springer, v. 29, n. 3, p. 1–24, 2021. Citado na página 33.

SEEMA, S. K. et al. Simulation-based performance evaluation of vanet routing protocols under indian traffic scenarios. *ICIC Express Letters*, v. 16, n. 1, p. 67–74, 2022. Citado na página 1.

SHAHIN, R. et al. Fog-rocl: A fog based rsu optimum configuration and localization in vanets. *Pervasive and Mobile Computing*, Elsevier, v. 94, p. 101807, 2023. Citado 4 vezes nas páginas 10, 11, 22 e 27.

SHI, W. et al. Edge computing: Vision and challenges. *IEEE internet of things journal*, IEEE, v. 3, n. 5, p. 637–646, 2016. Citado na página 8.

SHUHAIMI, N. I. et al. Performance analysis of radio propagation models in vanet application. In: IEEE. *2021 IEEE 11th IEEE Symposium on Computer Applications & Industrial Electronics (ISCAIE)*. [S.l.], 2021. p. 372–377. Citado 2 vezes nas páginas 22 e 27.

SILVA, L. G. et al. Desvendando a elasticidade de máquinas virtuais em vanets: Uma estratégia para aperfeiçoar o planejamento de capacidade em rsus. *Anais do XLII Simpósio Brasileiro de Redes de Computadores e Sistemas Distribuídos (SBRC 2024)*, 2024. Disponível em: <https://api.semanticscholar.org/CorpusID:273054728>. Citado na página 34.

SINHA, A.; MISHRA, S. K. Queue limiting algorithm (qla) for protecting vanet from denial of service (dos) attack. *International Journal of Computer Applications*, Foundation of Computer Science, v. 86, n. 8, p. 14–17, 2014. Citado 2 vezes nas páginas 1 e 3.

STRZĘCIWILK, D. Timed petri nets for modeling and performance evaluation of a priority queueing system. *Energies*, MDPI, v. 16, n. 23, p. 7690, 2023. Citado na página 3.

TAHER, Y. H. et al. Performance evaluation of software defined networking into vanets system. *Bulletin of Electrical Engineering and Informatics*, v. 13, n. 5, p. 3770–3778, 2024. Citado 3 vezes nas páginas 10, 22 e 26.

TAN, H.; CHUNG, I. Rsu-aided remote v2v message dissemination employing secure group association for uav-assisted vanets. *Electronics*, MDPI, v. 10, n. 5, p. 548, 2021. Citado 4 vezes nas páginas 1, 2, 23 e 27.

ULLAH, S. et al. Rsu assisted reliable relay selection for emergency message routing in intermittently connected vanets. *Wireless Networks*, Springer, v. 29, n. 3, p. 1311–1332, 2023. Citado 4 vezes nas páginas 1, 23, 24 e 26.

WANG, Y. et al. Appeal-based distributed trust management model in vanets concerning untrustworthy rsus. In: IEEE. *2023 IEEE Wireless Communications and Networking Conference (WCNC)*. [S.l.], 2023. p. 1–6. Citado 3 vezes nas páginas 21, 24 e 27.

WIRE, B. *U.S. Autonomous Car Market 2018-2023 - There Will Be Some 20.8 Million Autonomous Vehicles in Operation in the U.S. by 2030 - ResearchAndMarkets.com*. 2018. Disponível em: <https://www.businesswire.com/news/home/20180831005410/en/U.S.-Autonomous-Car-Market-2018-2023---There-Will-Be-Some-20.8-Million-Autonomous-Vehicles-in-Operation-in-the-U.S.-by-2030---ResearchAndMarkets.com>. Citado na página 1.

YU, H. et al. An rsu deployment strategy based on traffic demand in vehicular ad hoc networks (vanets). *IEEE Internet of Things Journal*, IEEE, v. 9, n. 9, p. 6496–6505, 2021. Citado na página 10.

ZHANG, Z. et al. Optimal cooperative mac strategies for wireless vanets with multiple roadside units. *IEEE Transactions on Vehicular Technology*, IEEE, 2024. Citado 2 vezes nas páginas 22 e 27.