UNIVERSIDADE FEDERAL DO CEARÁ

CENTRO DE TECNOLOGIA

DEPARTAMENTO DE ENGENHARIA DE TELEINFORMÁTICA

PROGRAMA DE PÓS-GRADUAÇÃO EM ENGENHARIA DE TELEINFORMÁTICA

MESTRADO ACADÊMICO EM ENGENHARIA DE TELEINFORMÁTICA

GUILHERME ALVES DE ARAÚJO

ALOCAÇÃO DE RECURSOS BASEADO EM PRIORIDADE DE TAREFAS E CONSUMO DE RECURSOS NA COMPUTAÇÃO EM BORDA

FORTALEZA

2023

GUILHERME ALVES DE ARAÚJO

ALOCAÇÃO DE RECURSOS BASEADO EM PRIORIDADE DE TAREFAS E CONSUMO DE RECURSOS NA COMPUTAÇÃO EM BORDA

Dissertação apresentada ao Curso de Mestrado Acadêmico em Engenharia de Teleinformática do Programa de Pós-Graduação em Engenharia de Teleinformática do Centro de Tecnologia da Universidade Federal do Ceará, como requisito parcial à obtenção do título de mestre em Engenharia de Teleinformática. Área de Concentração: Engenharia de Teleinformática. Sinais e sistemas.

Orientadora: Prof(a). Dra. Atslands Rego da Rocha.

FORTALEZA

2023

Dados Internacionais de Catalogação na Publicação
Universidade Federal do Ceará
Sistema de Bibliotecas
Gerada automaticamente pelo módulo Catalog, mediante os dados fornecidos pelo(a) autor(a)

A689a Araújo, Guilherme Alves de.
Alocação de recursos baseado em prioridade de tarefas e consumo de recursos na computação em borda / Guilherme Alves de Araújo. – 2023.
82 f. : il. color.

Dissertação (mestrado) – Universidade Federal do Ceará, Centro de Tecnologia, Programa de Pós-Graduação em Engenharia de Teleinformática, Fortaleza, 2023.
Orientação: Profa. Dra. Atslands Rego da Rocha .

1. Computação em borda. 2. Gerenciamento de recursos. 3. Alocação de recursos. 4. Modelos de classificação. 5. Prioridade de tarefas. I. Título.

CDD 621.38

GUILHERME ALVES DE ARAÚJO

ALOCAÇÃO DE RECURSOS BASEADO EM PRIORIDADE DE TAREFAS E CONSUMO DE RECURSOS NA COMPUTAÇÃO EM BORDA

Dissertação apresentada ao Curso de Mestrado Acadêmico em Engenharia de Teleinformática do Programa de Pós-Graduação em Engenharia de Teleinformática do Centro de Tecnologia da Universidade Federal do Ceará, como requisito parcial à obtenção do título de mestre em Engenharia de Teleinformática. Área de Concentração: Engenharia de Teleinformática. Sinais e sistemas.

Aprovada em: 21/09/2023.

BANCA EXAMINADORA

Prof(a). Dra. Atslands Rego da Rocha (Orientadora)
Universidade Federal do Ceará (UFC)

Prof(a). Dra Michela Mulas
Universidade Federal do Ceará (UFC)

Prof. Dr. Rafael Lopes Gomes
Universidade Estadual do Ceará (UECE)

Dedico este trabalho a todos que, de alguma forma, contribuíram para que eu pudesse chegar a este momento.

AGRADECIMENTOS

Agradeço primeiramente a Deus, por essa oportunidade e por ter me proporcionado mais uma vitória em minha vida. É ele o maior pilar que nos sustenta nesse mundo.

A minha mãe, Claudineide Alves, que foi o maior pilar para ser quem sou hoje, que abriu mão de muitas coisas na vida, para que eu pudesse realizar os meus sonhos, e que mesmo já não estando mais comigo em vida, tenho certeza que se orgulha muito do legado que deixou aqui.

Ao meu pai e irmão Francisco Haroldo e Ruan Alves, pela força, carinho e compreensão nos momentos de ausência, e que contribuíram significativamente para que eu conseguisse realizar mais uma etapa.

A Prof. Dra. Atslands Rego da Rocha, pela excelente orientação, confiança, ensinamentos, pelo tempo dedicado e por contribuir com minha formação acadêmica, profissional e pessoal.

Aos membros da banca examinadora, por aceitarem o convite e cederem parte dos seus tempos para contribuírem com esta pesquisa.

Aos colegas da turma de mestrado e laboratório, pelas reflexões, críticas e sugestões recebidas, em especial Dávila, Sandy Lucas, Noronha e Davi Querioz com quem tive o prazer de dividir esses dois anos de estudo.

A todos meus amigos e familiares, pelas conversas, carinho e apoio. Que estiveram comigo durante minha jornada, e partilharam vitórias e derrotas, e onde quer que estejam sempre terão um pedaço no meu coração em especial: Claúdia Alves, Lúcia Pires, Caio Alves, Sarah Alves, Pedro Ribeiro, Weyner Bezerra.

À Instituição FUNCAP, Fundação Cearense de Apoio ao Desenvolvimento Científico e Tecnológico, pelo apoio financeiro com a manutenção da bolsa de mestrado, e com recursos do projeto através do processo MLC-0191-00164.01.00/22.

A todos professores que já passaram pela minha vida. Meu muito obrigado a todos que fizeram com que esse dia fosse possível.

A minha noiva e amiga Carolina Alves Ribeiro, por toda força e amor durante esses anos de estudo. Agradeço por sempre está ao meu lado e contribuir com meu crescimento pessoal e profissional, mesmo nas mais variadas adversidades que a vida me proporcionou.

Quero expressar minha gratidão a todas as pessoas que, direta ou indiretamente, contribuíram para a realização desta pesquisa ou que tiveram um impacto positivo em minha

formação. A cada um de vocês, meus mais sinceros agradecimentos.

"A mente que se abre a uma nova ideia jamais voltará ao seu tamanho original." (Albert Einstein.)

RESUMO

Em dispositivos de Internet das Coisas (do inglês, *Internet of Things*, IoT), com baixo poder computacional, os dados precisam ser processados e informações relevantes extraídas em dispositivos com maior capacidade de processamento. A Computação em Borda se apresenta como um complemento à computação em nuvem tradicional, sendo também conhecido como *Cloud-to-Thing Continuum*. Os dispositivos na borda da rede possuem recursos computacionais para lidarem com o processamento e a análise de dados que os restritos dispositivos IoT eventualmente não tem capacidade para realizar. Com a Computação em Borda, é possível processar dados próximo aos dispositivos IoT (finais), e não apenas na nuvem frequentemente distante, reduzindo a latência para aplicações de Internet das Coisas. No entanto, a limitação de recursos dos nós de borda, que em geral têm menos poder computacional do que a nuvem, torna desafiador o processamento massivo de requisições. Neste trabalho é proposto um mecanismo de alocação de recursos baseado na classificação de prioridades de tarefas de dispositivos IoT e no consumo de recursos pelos dispositivos de borda. Para isso, foi avaliado o desempenho de diferentes tipos de algoritmos de classificação através da validação de métricas de modelos classificadores. O classificador mais eficiente apresentou uma acurácia de 92% e uma precisão de 90%. Os resultados indicam um bom desempenho da utilização desse classificador no abordagem avaliada neste trabalho, com médias de erro absoluto médio (MAE) de 0.07 e erro quadrático médio (MSE) de 0.08, respectivamente. Além disso, a utilização do classificador selecionado demonstrou ser uma abordagem eficiente na alocação de recursos para atendimento das requisições de dispositivos IoT. Os resultados mostram que, com o mecanismo de alocação proposto neste trabalho, o gerenciamento de recursos ocorre de forma mais controlada e eficiente, com uma utilização bem menor de recursos em comparação com uma alocação apenas baseado na distância. Foram testados diferentes cenários em relação à quantidade de requisições de dispositivos IoT e números de nós de borda, além de teste de mecanismo de falha. Portanto, o método proposto pode ser uma ferramenta valiosa para o gerenciamento eficiente de recursos na borda, com um custo computacional reduzido e uma alocação de recursos eficiente.

Palavras-chave: computação em borda; gerenciamento de recursos; alocação de recursos; modelos de classificação; prioridade de tarefas.

ABSTRACT

With low computational power in Internet of Things (IoT) devices, data must be processed and relevant information extracted in devices with higher processing capacity. Edge Computing emerged as a complementary solution to cloud computing, also known as Cloud- to-Thing continuum. Devices at the network edge have computational resources to handle the data processing and analysis that constrained IoT devices eventually cannot perform. Edge Computing allows data processing close to the end devices, and not only in the often far away Cloud, reducing latency for Internet of Things applications. However, the resource constraints of edge nodes, which have lower computational power than the cloud, make massive request processing challenging. The approach of this study evaluates the performance of different classification algorithms through the validation of metrics for classifying models. The most efficient classifier achieved an accuracy of 92% and a precision of 90%. The results indicate good performance of using this classifier in the evaluated approach, with mean absolute error (MAE) and mean squared error (MSE) averages of 0.07 and 0.08, respectively. Additionally, using this selected classifier efficiently allocates resources for IoT requests. The results demonstrate that resource management occurs more efficiently with our proposed mechanism, with significantly lower resource utilization compared to the allocation method based on distance. Different scenarios were tested regarding the number of requests, edge nodes, and failure mechanism. Therefore, the proposed method can become a valuable tool for efficient resource management, with reduced computational cost and efficient resource allocation.

Keywords: edge computing; resource management; resource allocation; classification models; task priority.

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

FCFS	First Come First Serve (Primeiro a chegar, primeiro a servir)
kNN	k-Nearest Neighbors (k-Vizinhos Mais Próximos)
MQTT	Message Queuing Telemetry Transport
RSSI	Received Signal Strength Indication (Indicador de Intensidade do Sinal Recebido)
SVM	Support Vector Machine (Máquina de vetores de suporte)

SUMÁRIO

1 INTRODUÇÃO

A Computação em Borda surge como um paradigma inovador que busca atender às demandas de aplicações com acesso massivo e latência crítica, aproximando a capacidade de processamento dos usuários finais. Este paradigma tem sido amplamente aplicado em ambientes de Internet das Coisas, representando uma infraestrutura descentralizada composta por nós de processamento distribuídos estrategicamente entre os dispositivos finais e a nuvem. Essa abordagem permite um processamento mais ágil e eficiente, transformando a borda em uma extensão da nuvem (MARTINEZ *et al.*, 2021). Dessa forma, a Computação em Borda se apresenta como um complemento à computação em nuvem tradicional, sendo também conhecido como *Cloud-to-Thing continuum*.

O paradigma da IoT tem sido amplamente adotado em muitas aplicações práticas, como cidades inteligentes (HAJAM; SOFI, 2021). Isso é possível por meio da interconexão e interoperabilidade de entidades físicas e virtuais com a IoT, possibilitando a criação de serviços inteligentes e a tomada de decisão informada para fins de monitoramento, controle e gerenciamento (TRAN-DANG; KIM, 2018). Hoje, a combinação de borda e nuvem oferece benefícios mútuos e permite que os sistemas *Cloud-to-Thing continuum* resultantes forneçam aos usuários finais serviços contínuos com diferentes requisitos de qualidade de serviço (QoS) em todo o *continuum* coisa-nuvem.

Diferentemente dos nós da nuvem, os recursos de computação (como processamento, armazenamento e energia) nos nós de borda são limitados e dinâmicos, resultando em cargas de trabalho em constante mudança, além de aplicações que competem por esses recursos restritos. Diante dessa situação, o gerenciamento eficiente desses recursos de computação limitados nos nós de borda torna-se um desafio significativo para uma utilização mais otimizada desses recursos (SHARIF *et al.*, 2022).

A arquitetura da Computação em Borda, em sua essência, consiste em múltiplas camadas interligadas. Na camada mais baixa, encontram-se os dispositivos IoT responsáveis por coletar, pré-processar e encaminhar os dados para os nós de borda. A camada intermediária é composta pelos nós de borda, que possuem uma capacidade de processamento superior em comparação aos dispositivos IoT. Esses nós desempenham funções de processamento de dados, tomada de decisões e realização de ações. Por fim, temos a camada superior, representada pela nuvem, que assume a responsabilidade de armazenar informações permanentes e processar os dados que os nós de borda não possuem capacidade computacional suficiente para realizar

(ARENA; PAU, 2020).

Nos últimos anos, uma área que tem se destacado significativamente dentro da ampla esfera da Computação em Borda é o gerenciamento de recursos. Essa área é de suma importância, pois se dedica à administrar os recursos disponíveis na rede, garantindo a alocação eficiente desses recursos na borda da rede e que os nós possam executar suas atividades de forma eficiente. O gerenciamento adequado dos recursos é essencial para evitar situações em que alguns nós consumam uma quantidade desproporcional de recursos, deixando outros nós com escassez, o que pode resultar em problemas operacionais e de desempenho (ZHOU *et al.*, 2021).

A alocação de recursos envolve a seleção criteriosa de um nó de borda, também conhecido como nó névoa (*"fog node"*), capaz de fornecer os recursos necessários para a execução de uma determinada tarefa ou solicitação proveniente de um dispositivo final (LIU *et al.*, 2019). Essa escolha se torna um dos principais desafios enfrentados na Computação em Borda, uma vez que os recursos são limitados. Além disso, devido ao compartilhamento dos recursos entre os dispositivos IoT, é possível que ocorram eventos imprevisíveis, indisponibilidade dos recursos nos nós da rede e tempos de resposta prolongados (DLAMINI; VENTURA, 2019).

Uma abordagem já amplamente explorada para solucionar esse problema é o uso de técnicas de aprendizado de máquina, buscando automatizar cada vez mais diferentes soluções para o desafio do gerenciamento de recursos na borda, como demonstrado em (MARTINS, 2021). A alocação eficiente de recursos representa um ponto de partida crucial para essa solução, visando evitar o consumo excessivo dos recursos dos nós e garantindo um gerenciamento mais efetivo (WANG *et al.*, 2022).

Entretanto, como as aplicações IoT possuem diferentes requisitos (como sensibilidade ou tolerância a atrasos), é interessante que a alocação eficiente de recursos leve em consideração esses aspectos. A classificação de prioridades de requisições das aplicações é uma abordagem em potencial desde que os recursos da borda podem ser alocados para a execução de tarefas com base em sua relevância e urgência. Alguns exemplos de aplicações práticas que podem ser beneficiados com a classificação de prioridades de tarefas são: Saúde Digital e Sistemas Industriais e Manufatura Inteligente. Em sistemas de saúde digital, priorizar as requisições de dispositivos médicos conectados em tempo real, pode garantir que dados críticos, como sinais vitais, sejam processados e enviados com alta prioridade para melhorar a tomada de decisões médicas em situações de emergência. Enquanto no contexto da Indústria 4.0, para Sistemas Industriais, priorizar as requisições de sensores e dispositivos em ambientes de manufatura

inteligente, pode garantir que as demandas críticas, como a monitorização de parâmetros de produção e de segurança, sejam tratadas com prioridade para evitar paradas ou falhas no processo produtivo.

Nesse cenário, é apresentado nesta dissertação um mecanismo para alocação de recursos em ambientes de computação em borda. O mecanismo proposto se baseia no monitoramento contínuo dos recursos disponíveis nos nós de borda e utiliza um classificador de prioridade de tarefas. Com isso, possibilita-se um gerenciamento eficiente desses recursos e, ao mesmo tempo, a seleção do nó mais apropriado para atender às diversas requisições solicitadas. Esse mecanismo visa otimizar o desempenho, a eficiência e a capacidade de resposta do ambiente de computação em borda, permitindo que as tarefas sejam processadas de forma ágil e eficaz, de acordo com suas necessidades específicas e com uso eficiente dos recursos dos nós.

O presente trabalho oferece importantes contribuições, sendo as principais:

- Um mecanismo de classificação baseado em aprendizado de máquina que estabelece prioridades para as tarefas de dispositivos IoT de acordo com a sua relevância e urgência de execução.
- Um conjunto de regras e definição de pesos para a priorização das tarefas que levam em consideração diversos fatores, como o tipo de requisição, latência, tipo de conexão, tempo de solicitação, distância e recursos disponíveis, proporcionando um tratamento mais adequado e personalizado para cada tarefa.
- Um mecanismo de alocação dos recursos dos nós de borda, considerando diferentes critérios, como a capacidade do nós e a distância, que visa garantir o uso apropriado desses recursos levando em consideração a prioridade das tarefas e o monitoramento da utilização de recursos de nós de borda.
- Um mecanismo de controle de falha de nós de borda.

1.1 Objetivos Gerais

Neste trabalho, o objetivo principal é propor um mecanismo de alocação de recursos de borda com base na classificação de prioridade de requisições de aplicações IoT e no monitoramento da utilização de recursos dos nós de borda para escalonamento de tarefas/requisições. Assim, monitorando continuamente os recursos disponíveis nos nós, o nó mais adequado na borda pode ser selecionado para atender aos requisitos exigidos pelas aplicações, levando em consideração os recursos necessários para executar os serviços solicitados.

1.1.1 Objetivos Específicos

Os objetivos específicos deste trabalho são:

1. Propor e desenvolver um modelo de classificação de prioridades para requisições provenientes de dispositivos IoT.
2. Estabelecer regras para definir prioridades baseadas em seis fatores: Recurso, Distância, Tipo de conexão, Tipo de requisição e Latência.
3. Propor um mecanismo eficiente de alocação de recursos para gerenciar a utilização de recursos computacionais na borda na rede.
4. Propor e desenvolver um mecanismo capaz de detectar falhas e lidar com elas nos nós de borda.
5. Avaliar o mecanismo proposto com o modelo de alocação aleatória, que leva em consideração apenas a distância para escolha do nós que serão alocadas as tarefas.

1.2 Produção Científica

No decorrer da pesquisa, foi realizada a seguinte contribuição científica:

– ARAÚJO, Guilherme A. de; BEZERRA, Sandy F. C.; ROCHA, Atslands R. da. Um Classificador de Prioridade de Requisições para Alocação de Recursos na Computação em Borda. In: SIMPÓSIO BRASILEIRO DE COMPUTAÇÃO UBÍQUA E PERVASIVA (SBCUP), 15. , 2023, João Pessoa/PB. Anais [...]. Porto Alegre: Sociedade Brasileira de Computação, 2023 . p. 131-140. ISSN 2595-6183. DOI: https://doi.org/10.5753/sbcup.2023.230787.

1.3 Estrutura da Dissertação

Esta dissertação é composta por sete capítulos que foram estruturados da seguinte forma. No Capítulo 1 é apresentada a contextualização do problema, juntamente com as motivações da pesquisa e os objetivos deste trabalho. No Capítulo 2 são apresentados os trabalhos relacionados que serviram como base para esta pesquisa. No Capítulo 3 é apresentada a fundamentação teórica para um melhor entendimento do trabalho. No Capítulo 4 é apresentado a metodologia adotada. No Capítulo 5 são apresentados os experimentos realizados. Os resultados obtidos são discutidos no Capítulo 6. Finalmente, no Capítulo 7 são apresentadas as conclusões obtidas a partir da pesquisa realizada, destacando as limitações encontradas, e possíveis trabalhos futuros que podem ser desenvolvidos para expandir e aprofundar o tema abordado.

2 TRABALHOS RELACIONADOS

Neste capítulo são apresentados alguns trabalhos relacionados, permitindo um melhor entendimento do estado da arte da posposta de estudo apresentada nesta Dissertação.

Um grande foco está sendo dado pelos acadêmicos à gestão de recursos na Computação em borda. Isso ocorre principalmente devido à busca por melhorias de desempenho, que é amplamente impactado pela alocação ineficiente dos recursos disponíveis na borda da rede. A partir desses esforços, diversas soluções algoritmos e metodologias têm sido propostas, e um progresso significativo já foi alcançado.

No trabalho (TRAN-DANG; KIM, 2021), os autores apresentam um algoritmo de alocação de recursos para a computação em borda chamado TPRA (*Task Priority-based Resource Allocation*), que se baseia na prioridade das tarefas. Nesse estudo, as tarefas são classificadas em duas categorias: normal e prioritária. As tarefas normais podem ser rejeitadas pelos provedores de recursos, ou seja, não serem atendidas, pois admite-se que possam sofrer atrasos. Por outro lado, as tarefas classificadas como prioritárias precisam ser concluídas com sucesso e não podem ser rejeitadas pelos provedores de recursos. As tarefas normais são colocadas em uma fila que calcula o tempo de atraso até o momento em que se tornam prioridade. Nesse ponto, elas precisam ser atendidas imediatamente, sem possibilidade de rejeição. No entanto, na proposta desse trabalho, essa classificação é expandida e apresenta-se três níveis de prioridade (Alta, Média e Baixa). Mesmo as requisições classificadas com prioridade baixa são atendidas, embora possam levar um pouco mais de tempo. Diferentemente do algoritmo proposto em (TRAN-DANG; KIM, 2021), no qual tarefas podem ser rejeitadas ou colocadas em uma fila, o algoritmo de classificação proposto usa a prioridade das requisições e considera os recursos disponíveis pelos nós de borda. Isso evita possíveis custos decorrentes de filas de espera extensas por prioridade ou por recursos.

No trabalho (BHUSHAN; MAT, 2021), é proposta uma arquitetura de computação em borda baseada em fila de prioridades, na qual os nós de borda são alocados dinamicamente com base na carga do sistema. A alocação de tarefas é realizada considerando a prioridade, que é definida com base no provisionamento de serviços e divide as tarefas em duas classes: Alta prioridade (sensível ao atraso) e Baixa prioridade (tolerante ao atraso). O trabalho também assume um parâmetro, que representa a proporção média do número de tarefas de alta prioridade em relação ao número total de tarefas, para calcular o tempo médio de atraso de cada classe. Na proposta apresentada, além de levar em conta as prioridades, um fator crucial para a alocação das tarefas é determinar se os nós de borda possuem recursos suficientes para executar uma

tarefa específica. A alocação de recursos considera o estado dos nós de borda, a partir da disponibilidade de recursos. Dessa forma, possibilita-se que as tarefas sejam alocadas somente em nós de borda que tenham capacidade suficiente para executá-las, levando em consideração tanto as prioridades das tarefas quanto a disponibilidade de recursos.

O trabalho citado (MADEJ *et al.*, 2020) apresenta diferentes técnicas de escalonamento. A primeira é uma estratégia *First Come First Serve (Primeiro a chegar, primeiro a servir) (FCFS)*, na qual as tarefas são escalonadas na ordem em que são recebidas. Além disso, o artigo propõe três estratégias adicionais: orientada aos clientes, orientada a prioridades e uma abordagem híbrida que considera tanto os clientes quanto as prioridades das tarefas. A estratégia orientada aos clientes trata todos os clientes como tendo a mesma prioridade, sem levar em consideração as prioridades individuais das tarefas. Já a abordagem orientada a prioridades considera as prioridades das tarefas em si, independentemente dos clientes. Por fim, a estratégia híbrida leva em conta tanto os clientes quanto as prioridades das tarefas, buscando garantir uma alocação equitativa e eficiente dos recursos da borda. Essas técnicas de escalonamento propostas no artigo têm como objetivo lidar com a questão da disponibilidade limitada de recursos na borda, garantindo a equidade entre os clientes e considerando as prioridades das tarefas. Entretanto, o trabalho (MADEJ *et al.*, 2020) supõe que os recursos necessários estão sempre disponíveis quando uma solicitação de requisição/tarefa é feita. Nem todas as solicitações de um servidor podem ser escalonadas em um nó de borda, portanto, garantir que os nós que irão executar a tarefa possuam os recursos necessários enquanto considera as prioridades das tarefas se torna crucial. Na proposta, há o conhecimento sobre o estado de cada um dos nós de borda disponíveis no momento da seleção dos nós para execução da tarefa. Esse conhecimento é utilizado para garantir que as tarefas sejam alocadas apenas nos nós de borda que possuam recursos suficientes para executá-las, considerando suas capacidades e disponibilidades específicas. Essa abordagem visa otimizar a alocação de recursos e garantir a eficiência do sistema.

Com o objetivo de alcançar uma alocação eficaz de recursos e uma utilização ideal dos mesmos, o trabalho (SHARIF *et al.*, 2023) apresenta um mecanismo adaptativo de alocação de recursos para a Computação em Borda. Esse mecanismo visa alocar dinamicamente os recursos disponíveis levando em consideração a natureza das solicitações recebidas a fim de obter uma utilização ideal. No esquema proposto pelos autores, o mecanismo se adapta às demandas e prioridades das solicitações recebidas. Após identificar uma solicitação, a prioridade é definida com base na sensibilidade ao atraso da tarefa, que é verificada durante o processo

de identificação. Os recursos disponíveis são então alocados de acordo com as prioridades das solicitações recebidas, para atender às restrições estabelecidas.

É importante mencionar que o mecanismo proposto em (SHARIF *et al.*, 2023) é adaptável a um número máximo de solicitações recebidas e visa otimizar a utilização de recursos limitados no nó de borda. No entanto, o autor recorre a técnicas de alocação de recursos dinâmicas, considerando apenas a natureza das solicitações, o que pode não ser a melhor abordagem, uma vez que a alocação sem levar em consideração os recursos disponíveis nos diferentes nós de borda pode ser problemática. Em contraste, o classificador de prioridades, proposto nessa dissertação, leva em consideração uma variedade de fatores, como os serviços requeridos, o tempo decorrido desde a solicitação, a distância em relação ao nó, a latência e os recursos disponíveis. Esses aspectos são considerados para definir a prioridade das tarefas e, posteriormente, alocá-las de maneira mais eficiente. Na proposta, é implementado um mecanismo de gerenciamento de falhas para lidar com possíveis interrupções ou problemas durante a execução das tarefas. Ao contrário do mecanismo apresentado em (SHARIF *et al.*, 2023), o enfoque é mais abrangente e considera múltiplos aspectos para determinar as prioridades de alocação.

Com base nessas informações, o mecanismo é capaz de realizar uma alocação de recursos eficiente, considerando não apenas as necessidades da tarefa, mas também a disponibilidade e as condições dos recursos na borda. Dessa forma, busca-se otimizar a utilização dos recursos disponíveis, garantindo um melhor desempenho e confiabilidade do sistema de alocação de tarefas.

O artigo (BUI *et al.*, 2021) apresenta uma melhoria do algoritmo *"Score Based Match-Making"* com objetivo de otimizar a distribuição de tarefas nos nós de borda e qualidade de serviço na alocação de recursos. É proposto um algoritmo *"Match-Making"* que utiliza uma pontuação baseada em seis fatores (distância entre os nós, condição da rede, latência, memória, capacidade da CPU e histórico de execução de tarefas do nó) para lidar com desafios relacionados à prioridade na alocação de recursos. Além disso, o algoritmo inclui um fator de preempção para tratar situações de emergência. O fator de preempção permite interromper temporariamente uma tarefa em execução, caso uma tarefa de maior prioridade entre na fila. A intenção é retomar a tarefa interrompida posteriormente. Essa abordagem é implementada por meio de uma plataforma de orquestração em diferentes cenários. O algoritmo proposto é comparado com outros dois tipos de alocação: aleatória e uma abordagem "naive"semelhante

ao "first come first serve". O melhor nó para execução da tarefa é escolhido ao comparar as pontuações dos nós disponíveis.

Em resumo, o objetivo principal do algoritmo proposto pelos autores é utilizar um conjunto de parâmetros importantes (distância entre os nós, condição estimada da rede, latência, memória, capacidade da CPU e histórico de execução de tarefas do nó) para atribuir uma pontuação a cada nó disponível. Essa pontuação é usada para tomar decisões eficientes na alocação de recursos, levando em consideração diversos aspectos relevantes para garantir um balanceamento adequado e a qualidade do serviço. Apesar de os autores considerarem vários fatores para alocar as tarefas, o artigo não menciona explicitamente se foram utilizadas técnicas para recuperação em caso de falhas nos nós.

No entanto, na abordagem proposta, é implementado um mecanismo para lidar com falhas nos nós que disponibilizam recursos. Em caso de falha em um nó, técnicas de recuperação serão acionadas para contornar essa situação e garantir a continuidade da execução das tarefas. Além disso, a estratégia dos autores de (BUI *et al.*, 2021) em relação a preempção, que permite interromper a execução de uma tarefa para alocar recursos para outra tarefa mais prioritária, pode levar a problemas de escalonamento e atrasos, especialmente se não for bem gerenciada. Enquanto na abordagem proposta, as tarefas não são interrompidas durante a execução e permanecem nos nós de borda designados até que sejam concluídas. Esse aspecto é relevante, uma vez que recursos já foram alocados para a execução dessas tarefas, e interrompê-las poderia resultar em desperdício de recursos. A proposta busca maximizar a utilização dos recursos e evitar interrupções desnecessárias, por meio dos mecanismos propostos de recuperação e da estratégia de conclusão das tarefas em seus respectivos nós. Isso proporciona robustez e eficiência na alocação de recursos, levando em consideração a minimização de desperdício e o uso eficiente dos recursos disponíveis.

Em (YIN *et al.*, 2020), o método de otimização concentra-se na busca pelo ponto de atraso médio mínimo para a realização de tarefas com o objetivo de otimizar e alocar recursos do sistema. Para isso, os autores utilizam um limite inferior baseado em uma função quadrática para determinar a quantidade mínima de recursos necessária para processar cada tarefa. Além disso, o modelo implementa uma função de escalonamento que considera três critérios: tempo de atraso, tempo de transferência e consumo de recursos. Essa função de escalonamento permite calcular o tempo de processamento necessário e a quantidade mínima de recursos exigida para cada tipo de tarefa.

Por outro lado, nessa dissertação, a priorização das tarefas visa especificamente a definição e alocação de recursos nos nós de borda com eficiência. O foco vai além de verificar apenas se o nó é capaz de processar a tarefa ou não, buscando otimizar a alocação em termos de eficiência geral.

Além disso, outro aspecto importante ausente no trabalho de (YIN *et al.*, 2020) é a descrição de um mecanismo de controle para identificar e lidar com falhas. Em contraste, na abordagem proposta neste trabalho, é apresentado como lidar com possíveis falhas nos nós de borda, garantindo a robustez e confiabilidade do sistema.

A Tabela 1 lista os trabalhos relacionados e suas comparações:

Tabela 1 – Trabalhos relacionados

Trabalho	Técnica Computacional	Controle de Falha	Critérios para Priorização
(TRAN-DANG; KIM, 2021)	Algoritmo de Priorização de Tarefas	Não	Tempo de Atraso
(BHUSHAN; MAT, 2021)	Priorização Baseado em Provisionamento de Serviço	Não	Sensibilidade ao Atraso e Tempo de Espera
(MADEJ *et al.*, 2020)	Combinação e Priorização Baseado em Pontuação	Não	Distância entre os nós, memória e capacidade de processamento
(SHARIF *et al.*, 2023)	Adaptativo e Baseado em Prioridade	Não	Restrição de Tempo da Requisição
(BUI *et al.*, 2021)	Pontuação Baseada em Seis Fatores	Não	Distância entre os nós, Latência, Condição Estimada da rede, Memória, Processador e Histórico de Conexões dos Nós
(YIN *et al.*, 2020)	Algoritmo de função Quadrática	Não	Tempo de Atraso, Tempo de transferência e Recurso Minimo
Mecanismo Proposto	**Algoritmo Classificação Baseado em Prioridades**	Sim	**Tipos de Serviços, Recursos, Latência, Tipo de Conexão, Tempo e Distância**

Fonte: elaborada pelo autor.

Em síntese, percebe-se que há pesquisas que se propõem a utilizar critérios para priorizar tarefas e equilibrar o uso dos recursos disponíveis. Essas abordagens consideram as tarefas e/ou recursos dos nós de borda como critério de priorização. No entanto, a real contribuição da proposta desta dissertação, e que a diferencia dos demais trabalhos, é o fato de levar em conta não apenas as tarefas/requisições, mas também a limitação dos recursos disponíveis nos nós de borda, além de fatores importantes como a distância do nó, a latência, o tipo de conexão e o tempo decorrido desde a requisição dessa tarefa. Ademais, o controle de falhas é um ponto importante em nossa abordagem, garantindo que todas as requisições sejam

atendidas independentemente de falhas em nós de borda. O objetivo deste trabalho é gerenciar os recursos nos nós de borda, classificando suas prioridades e determinando o melhor nó para alocação, permitindo um uso balanceado dos recursos de borda para atender às necessidades dos serviços solicitantes.

Por fim, os trabalhos investigados apresentaram bons resultados, porém alguns deles presumem que os recursos estarão sempre disponíveis para atender às necessidades das tarefas requisitadas, e não consideram múltiplos aspectos para o escalonamento das tarefas ou são custosos computacionalmente. Nesta pesquisa, busca-se contribuir com avanços enfatizando que os recursos de borda são limitados, o que pode não permitir atender todas as requisições. A abordagem proposta visa otimizar a utilização dos recursos disponíveis, aumentando a eficiência e a confiabilidade dos serviços prestados. Ao levar em conta a limitação dos recursos de borda e priorizar tarefas com critérios adequados, o objetivo é garantir que cada requisição seja atendida de forma eficaz, mesmo em situações de falhas nos nós de borda.

3 FUNDAMENTAÇÃO TEÓRICA

Neste capítulo são abordados os conceitos relevantes para o desenvolvimento desta pesquisa. Inicialmente, será fornecida uma descrição do conceito de computação em borda. Em seguida, serão apresentados os conceitos de gerenciamento de recursos, alocação e virtualização. Por fim, são discutidos os modelos de classificação e são apresentados os algoritmos de Aprendizado de Máquina selecionados para os testes de classificação de prioridades em nossos experimentos.

3.1 Computação em Borda

A Internet das Coisas está transformando a maneira como as pessoas interagem com o mundo físico, proporcionando um aumento de conectividade. A IoT se refere à implantação de vários dispositivos inteligentes interconectados que suportam tarefas diárias. Essa interconexão dos dispositivos introduzirá novas aplicações com um potencial ilimitado e um impacto massivo, permitindo a participação em massa dos usuários e impulsionando especialmente a comunicação entre máquinas e sensores/atuadores (BELLAVISTA *et al.*, 2019).

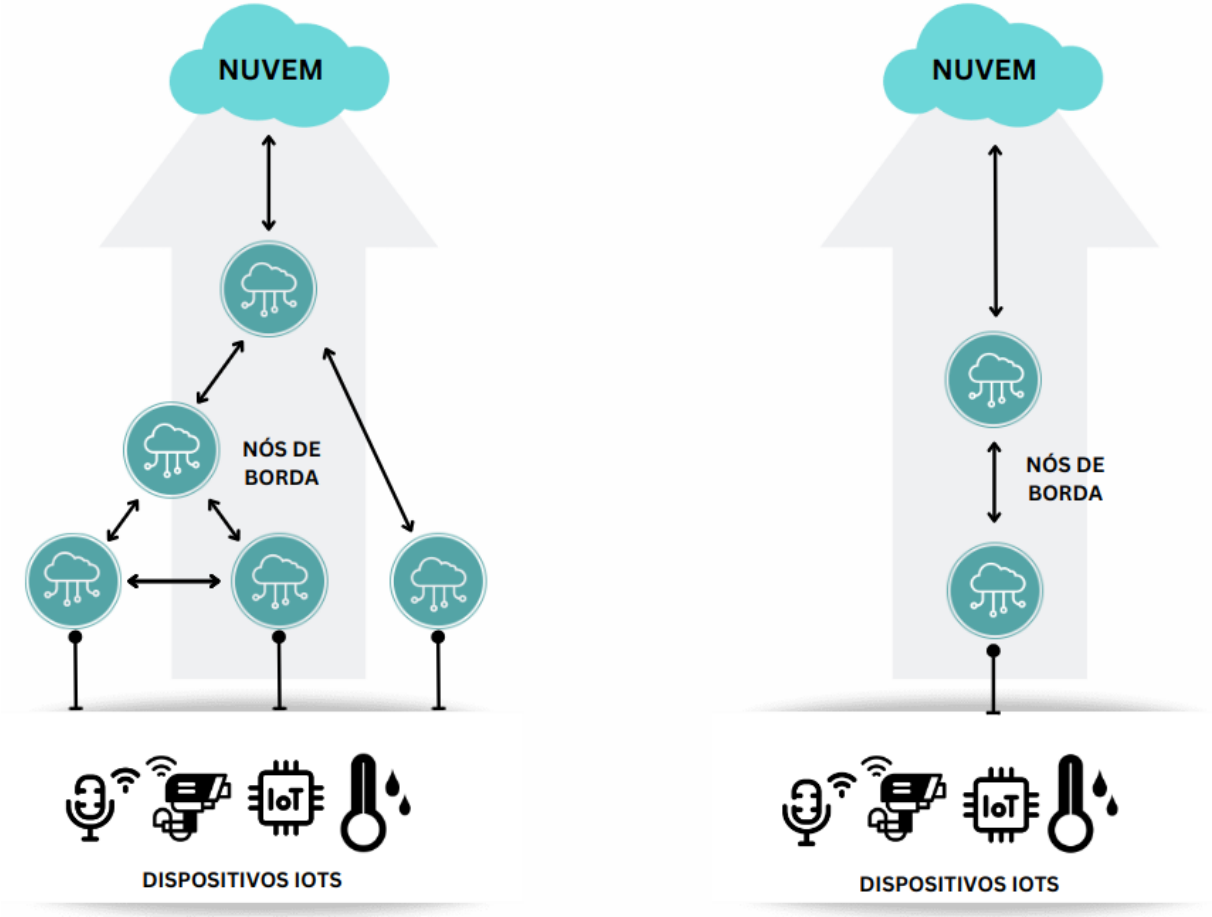
A computação em borda é um paradigma de computação que visa aproximar os recursos de processamento e armazenamento de dados de dispositivos e sensores, reduzir a latência e melhorar a eficiência da transmissão de dados. A borda ou névoa (*fog*), é composta por vários nós chamados nós da borda (*fog nodes*), que estabelecem a comunicação entre a nuvem e os dispositivos inteligentes. Com a implantação desse modelo de computação e o uso extensivo de dispositivos inteligentes, a computação em borda tornou-se um recurso necessário. A utilização da borda permite uma redução significativa no tráfego da rede (que seria usada para transferir grandes volumes de dados para a nuvem), proporcionando serviços na borda da rede, antes restritos à nuvem. Isso pode resultar em uma redução do congestionamento, custos e latência. Além disso, permite maior autonomia e resposta em tempo real, o que é essencial para aplicações críticas como sistemas de monitoramento, controle industrial, veículos autônomos e cidades inteligentes.

De acordo com (BONOMI *et al.*, 2012), a computação em borda pode ser definida como "uma plataforma altamente virtualizada que visa fornecer serviços de computação, armazenamento e comunicação". Esses serviços são disponibilizados entre os dispositivos finais e os centros de dados tradicionais de computação em nuvem, que estão geralmente localizados na

borda da rede. No entanto, eles também podem estar próximos à nuvem, aos usuários ou em qualquer outro ponto ao longo do caminho entre os dispositivos finais e a nuvem. A localização dos centros de dados pode variar de acordo com a arquitetura e a estratégia de implantação.

A Figura 1 ilustra exemplos de ambientes de computação em borda. Na figura pode ser observado que a quantidade, posicionamento e comunicações entre os nós de borda podem variar de acordo com a arquitetura escolhida. A arquitetura selecionada pode influenciar esses aspectos, permitindo uma flexibilidade na configuração da infraestrutura de computação em borda para atender às necessidades e requisitos específicos do sistema.

Figura 1 – Computação em Borda



Fonte: Elaborado Pelo Autor

Ao mover o poder de computação para a borda da rede, a computação de borda permite que tarefas locais sejam executadas com mais eficiência, minimiza a dependência de recursos de nuvem e aproveita a proximidade de dados e dispositivos de borda (DLAMINI; VENTURA, 2019).

3.2 Alocação de Recursos

A computação em borda é uma tecnologia poderosa utilizada em várias aplicações inovadoras, como 5G, jogos online, veículos autônomos, redes elétricas inteligentes, controle inteligente de tráfego, IIoT (Internet das Coisas Industrial), saúde 4.0, IoV (Internet dos Veículos) e outras (KANSAL *et al.*, 2022). Essas aplicações dependem de um eficiente gerenciamento dos recursos, que envolve os processos de alocar, monitorar e otimizar os recursos disponíveis nos nós de borda.

Um dos principais desafios na computação em névoa/borda é a limitação dos recursos disponíveis na rede de borda. Uma gestão eficiente dos recursos de névoa pode ter um impacto significativo na melhoria da qualidade da experiência (QoE), ao mesmo tempo em que reduz os custos para os provedores de serviço e os usuários finais (SAMANTA; TANG, 2020).

No gerenciamento de recursos da borda, são destacadas nove categorias principais: planejamento de recursos, colocação de aplicativos, balanceamento de carga, alocação de recursos, estimativa de recursos, transferência de tarefas, provisionamento de recursos, descoberta de recursos e orquestração de recursos (KANSAL *et al.*, 2022). Neste trabalho em particular, o método de gerenciamento utilizado foi o de alocação de recursos de forma a balancear o uso dos recursos dos nós de borda.

Na computação em borda, o balanceamento de recursos geralmente é usado para garantir que os recursos disponíveis em um sistema, como CPU, memória, largura de banda de rede ou espaço de armazenamento, sejam distribuídos de maneira justa e eficiente entre os processos ou tarefas em execução. O objetivo é evitar gargalos, maximizar o desempenho e evitar a sobrecarga de recursos em determinadas partes do sistema (KANSAL *et al.*, 2022).

A alocação de recursos na computação de borda refere-se ao processo de distribuir e alocar efetivamente os recursos de computação disponíveis para oferecer suporte a serviços e aplicativos na infraestrutura de computação na borda (LAHMAR; BOUKADI, 2020). Na computação em borda, os recursos de computação estão localizados em dispositivos de borda, como roteadores, pontos de acesso Wi-Fi, servidores locais e dispositivos IoT. Esses recursos incluem poder de computação, memória, armazenamento e largura de banda de rede.

A alocação de recursos na computação em borda envolve decidir quais tarefas ou serviços devem ser executados em dispositivos de borda específicos, levando em consideração as capacidades e limitações de cada dispositivo individualmente. Essa decisão é baseada em fatores como a latência exigida pelo serviço, carga atual do dispositivo, capacidade de computação

disponível e requisitos de segurança e privacidade. Portanto, segundo (SARAH *et al.*, 2023), a alocação de recursos pode ser considerada como um problema de otimização, em que os recursos são alocados de maneira a alcançar um objetivo específico em um determinado cenário.

A alocação efetiva de recursos em nós na borda da rede é essencial para garantir o desempenho adequado de serviços e aplicações, além de otimizar o uso dos recursos disponíveis na infraestrutura de borda (BACHIEGA *et al.*, 2023). Ao considerar a distribuição de recursos nos nós de borda, é necessário levar em consideração diversas características dos dispositivos, como capacidade de processamento e armazenamento, disponibilidade de memória, largura de banda de rede e consumo de energia. Esses fatores desempenham um papel fundamental na determinação de como os recursos serão alocados entre as diferentes aplicações em execução. Isso inclui monitorar continuamente os recursos, analisar os requisitos do serviço e ajustar a alocação de recursos conforme necessário para atender às necessidades dos usuários finais.

Além das características dos dispositivos, a localização geográfica dos nós de borda também é um aspecto importante na alocação de recursos. É essencial considerar a proximidade dos dados que serão processados, a fim de minimizar a latência e maximizar o desempenho. Dessa forma, os recursos podem ser alocados estrategicamente, permitindo que o processamento seja realizado o mais próximo possível da fonte dos dados, evitando atrasos indesejados.

3.2.1 Escalonamento de tarefas

O escalonamento de tarefas em computação de borda refere-se ao processo de gerenciamento da alocação de tarefas e recursos computacionais em um ambiente de borda. Isto é importante para garantir a execução eficiente das tarefas, levando em consideração as limitações dos dispositivos na borda da rede. O escalonamento de tarefas envolve a distribuição inteligente de processamento de dados e tarefas de computação para dispositivos específicos com base em vários fatores, como disponibilidade de recursos, latência de rede, desempenho e requisitos de segurança (BENCHIKH; LOUAIL, 2021)

O objetivo do escalonamento de tarefas é otimizar o uso dos recursos disponíveis, garantir tempos de resposta adequados e minimizar o consumo de energia. Isto é especialmente importante em cenários onde a latência é crítica, como aplicações em tempo real, saúde, transporte e muitos outros casos de uso de IoT.

3.3 Virtualização

Uma tecnologia fundamental para os paradigmas de computação em borda e computação em nuvem é a virtualização de recursos, que dissocia recursos de hardware do software para executar vários "locatários"(*tenants*) no mesmo hardware (MANSOURI; BABAR, 2021).

A virtualização com *Docker* é uma tecnologia de virtualização no nível do sistema operacional que permite empacotar e distribuir aplicativos em contêineres, ao contrário da virtualização tradicional, em que cada máquina virtual (VM) executa um sistema operacional completo.

Um contêiner é uma unidade leve e isolada que contém tudo o que um aplicativo precisa para executar, como código, bibliotecas, dependências e variáveis de ambiente. Cada contêiner compartilha o mesmo kernel do sistema operacional host, tornando-os mais eficientes em termos de recursos (POTDAR *et al.*, 2020). Devido a restrição de recursos dos nós de borda, o uso de técnicas de virtualização leves, como contêineres é potencializado, pois exigem menos ciclos de CPU, menor tamanho de memória e menos tempo de instanciação (MANSOURI; BABAR, 2021).

O *Docker* permite que os desenvolvedores criem, implantem e executem aplicativos de forma consistente em vários ambientes, independentemente do sistema operacional subjacente (DOCKER, 2022). O *Docker* fornece uma plataforma para criar, implantar e gerenciar contêineres, usando imagens do *Docker*. As imagens do *Docker* são pacotes independentes contendo tudo o que é necessário para executar um aplicativo, incluindo código, bibliotecas e configurações. Um arquivo de definição chamado *Dockerfile* especifica as instruções para construir as imagens. Desse modo, a virtualização com *Docker* é uma tecnologia que fornece portabilidade, eficiência e fácil gerenciamento de aplicativos.

3.4 Métodos de Classificação

O objetivo de técnicas de Aprendizado de máquina é reconhecer padrões em estruturas de dados que são difíceis de detectar manualmente. Com base nesses padrões, podem ser utilizadas técnicas de Aprendizado de Máquina para classificar novos dados de forma automatizada. Nesse trabalho, os esforços relacionados aos métodos de Aprendizado de Máquina são direcionados para a tarefa específica de classificação de prioridade para requisições ou tarefas originadas de dispositivos finais.

A classificação é uma tarefa fundamental no campo do Aprendizado de Máquina. É um processo no qual um algoritmo é treinado para identificar e atribuir categorias ou rótulos a diferentes instâncias de dados com base em características ou atributos específicos. O objetivo da classificação é criar um modelo capaz de generalizar os padrões identificados durante o treinamento para classificar corretamente novos dados não vistos anteriormente (RAJPUT; OZA, 2017). Um dos métodos de classificação é o aprendizado supervisionado, no qual os dados de treinamento possuem rótulos ou saídas esperadas associadas a cada exemplo. O objetivo é treinar um modelo de Aprendizado de Máquina para aprender a mapear os dados de entrada para as saídas correspondentes.

No aprendizado supervisionado, os dados de entrada e saída já são fornecidos à máquina, também conhecidos como dados de treinamento ou dados rotulados (SUYAL; GOYAL, 2022). Quando uma nova entrada é fornecida à máquina, ela produzirá uma saída com base em sua experiência anterior e nos dados disponíveis. Nesse algoritmo, a máquina utiliza o conhecimento adquirido a partir de seu histórico e exemplos rotulados para prever eventos futuros.

Durante o processo de treinamento, o algoritmo de classificação recebe um conjunto de dados rotulados, onde cada exemplo possui um conjunto de características e uma classe ou categoria conhecida. O algoritmo analisa esses exemplos de treinamento e aprende a relação entre os recursos e as classes correspondentes. Então, o modelo treinado pode ser usado para classificar novos dados não rotulados, ou seja, atribuir uma classe a instâncias desconhecidas com base no conhecimento adquirido durante o treinamento.

Existem vários algoritmos de classificação em aprendizado de máquina (do inglês, *Machine learning*, ML), incluindo *Naive Bayes*, Árvores de Decisão (*Decision Tree*), Árvores Randômicas (*Random Forest*), Máquina de Vetores de Suporte (*Support Vector Machines*, SVM), k-Vizinhos Mais Próximos (*K-Nearest Neighbors*, KNN) entre outros. Cada algoritmo tem suas próprias características e suposições subjacentes, tornando-o adequado para diferentes tipos de problemas de classificação. A escolha do algoritmo de classificação depende do contexto do problema, do tipo de dados e dos requisitos específicos da aplicação.

A seguir são apresentados os métodos de Aprendizado de Máquina usados nessa dissertação.

3.4.1 k-Vizinhos Mais Próximos (kNN)

O algoritmo kNN é amplamente utilizado em Aprendizado de Máquina supervisionado. É considerado simples, mas eficaz e é usado para muitos problemas de classificação. No kNN, a classificação de um novo exemplo é determinada com base em sua proximidade com os exemplos de treinamento. O valor "k"no nome do algoritmo refere-se ao número de vizinhos mais próximos considerados para a tomada de decisão. Desse modo, o algoritmo calcula a distância entre o exemplo de teste e todos os exemplos de treinamento e seleciona os "k"exemplos mais próximos (JABBAR *et al.*, 2013).

Após identificar "k"vizinhos mais próximos, a classificação do novo exemplo é determinada pela maioria das classes desses vizinhos. Por exemplo, se a maioria dos vizinhos "k"pertencer à classe "A", o novo exemplo também será classificado como pertencente à classe "A"(KRAMER, 2011).

O algoritmo k-Nearest Neighbors (k-Vizinhos Mais Próximos) (kNN) é versátil e pode ser aplicado a problemas de classificação binária e multiclasse. Ele não assume nenhuma distribuição de dados específica e pode ser usado para dados numéricos e categóricos.

O kNN é: (i) um classificador simples e intuitivo que se baseia na proximidade dos pontos de dados. (ii) adequado para conjuntos de dados com relações não lineares e não assume uma distribuição específica dos dados. (iii) O KNN pode ser facilmente interpretado e compreendido, e é útil quando a fronteira de decisão é complexa e não pode ser facilmente modelada por métodos lineares (SCIKIT-LEARN, 2023).

3.4.2 Máquina de Vetores de Suporte

A SVM é um algoritmo de Aprendizado de Máquina que pode ser usado para tarefas de regressão, conhecido como SVR, Regressão de Vetores de Suporte (*Support Vector Regression*) (AWAD; KHANNA, 2015).

No contexto da classificação, o SVM busca encontrar um hiperplano ótimo que possa separar as classes de maneira eficiente. O hiperplano é uma representação geométrica de uma fronteira de decisão que separa as classes no espaço de características. A ideia básica do SVM é identificar vetores de suporte, que são os pontos de dados mais próximos do limite de decisão. Esses vetores de suporte são usados para determinar a posição e a orientação do hiperplano. O objetivo é maximizar o *span* entre as classes, ou seja, maximizar a distância entre o hiperplano e

os vetores de suporte (SMOLA; SCHöLKOPF, 2004).

O Support Vector Machine (Máquina de vetores de suporte) (SVM) (SCIKIT-LEARN, 2023) é: (i) um classificador que busca encontrar um hiperplano de separação ótimo entre as classes; (ii) eficaz em conjuntos de dados com separação clara e pode lidar bem com dados de alta dimensionalidade; (iii) O SVM pode lidar com problemas de classificação não lineares usando estratégias de kernel para mapear os dados em espaços de maior dimensão.

3.4.3 Regressão Logística

A Regressão Logística é um modelo estatístico amplamente utilizado para realizar classificação binária. Embora o nome contenha "regressão", a regressão logística é uma técnica de classificação e não a regressão propriamente dita. Na classificação, o objetivo é prever a associação de uma instância a uma das duas ou mais classes possíveis. A regressão logística é usada quando a variável de resposta é categórica e possui apenas duas categorias, geralmente codificadas como 0 e 1. A regressão logística multiclasse, também conhecida como regressão logística multinomial, é uma extensão do modelo de regressão logística para problemas de classificação com mais de duas classes (JR *et al.*, 2013).

Um modelo de regressão logística usa uma função logística (também conhecida como função sigmoide) para estimar a probabilidade de uma instância pertencer a uma determinada classe. A função logística mapeia um valor linear (uma combinação linear de variáveis preditoras) para um valor entre 0 e 1 que representa a probabilidade de pertencer à classe positiva.

Durante o treinamento do modelo de regressão logística, os coeficientes são estimados usando técnicas de otimização como máxima verossimilhança. Esses coeficientes representam a contribuição de cada variável preditora para a previsão da probabilidade de pertencer à classe positiva. Uma vez estimados os coeficientes, o modelo de regressão logística pode ser utilizado para prever novas instâncias e calcular a probabilidade de pertencer à classe positiva. Uma regra de decisão é usada para classificar a instância com base nas probabilidades calculadas (JR *et al.*, 2013).

A regressão logística é: (i) um classificador linear que modela a relação entre as variáveis independentes e a probabilidade de pertencer a uma classe; (ii) eficiente computacionalmente e pode ser facilmente interpretado. (iii) útil em problemas de classificação binária, mas também pode ser estendido para problemas de classificação multiclasses, além de lidar bem com conjuntos de dados com características categóricas ou numéricas, desde que sejam tratadas

corretamente (SCIKIT-LEARN, 2023).

Esses classificadores foram selecionados por sua simplicidade, eficiência computacional e capacidade de lidar com diferentes tipos de dados e problemas de classificação. No entanto, é importante ressaltar que a escolha dos classificadores depende do contexto do problema, das características do conjunto de dados e dos objetivos do desenvolvedor.

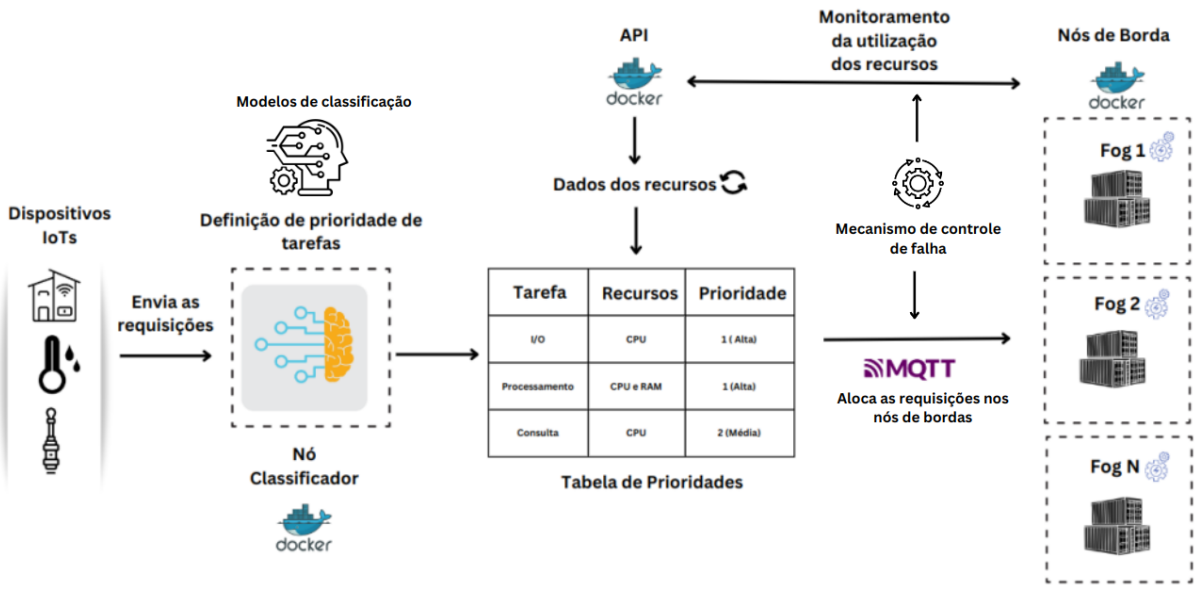
4 METODOLOGIA

Neste capítulo está descrita a arquitetura do mecanismo de gerenciamento de recursos proposto nesta dissertação e seus componentes, além dos critérios de priorização, da escolha de nós e controle de falha dos nós de borda.

4.1 Gerenciamento de Recursos baseado em Prioridades de Tarefas e Consumo de Recursos dos Nós de Borda

O mecanismo de gerenciamento de recursos proposto é baseado em aprendizado de máquina para a classificação e definição de prioridades de requisições vindas de dispositivos finais (IoT) em ambientes de computação de borda. A Figura 2 mostra a arquitetura do mecanismo proposto. Brevemente, as aplicações, presentes nos dispositivos IoT, enviam suas requisições para um nó classificador, cuja função é receber esses dados (requisições) e classificá-los em níveis de prioridade usando aprendizado de máquina. Com base na prioridade da tarefa e no monitoramento da utilização de recursos dos nós de borda, as tarefas são executados no nó de borda escolhido. Nas próximas subseções são descritos os detalhes do mecanismo: requisições, nó classificador, alocação de recursos e mecanismo de controle de falha.

Figura 2 – Arquitetura do Mecanismo Proposto



Fonte: (Elaborado pelo Autor)

4.1.1 Requisições

As requisições dos dispositivos IoT podem abranger uma variedade de tarefas, como acionamento, armazenamento, processamento, entre outras. Para as requisições, são levadas em consideração as seguintes premissas:

1. Cada requisição possui um tipo específico de tarefa, por isso, foram adicionados os tipos de serviços ao conjunto de dados (Armazenamento, Processamento, Consulta e I/O);
2. As requisições são oriundas de dispositivos que podem ter diferentes tipos de conexões de rede;
3. A latência das requisições varia de acordo com o tipo de rede utilizado.

Uma requisição é definida como uma tupla $R_i = (TR_i, L_i, RN_i, H_i, D_i, TC_i)$, em que TR_i representa o tipo da requisição i, L_i indica a latência da requisição i, RN_i se refere ao recurso necessário para atender a requisição i, H_i é o horário em que a requisição i é disparada, D_i representa a distância dos dispositivos em relação ao classificador da requisição i e TC_i representa o tipo de conexão da requisição i.

Esses dados são as variáveis de entrada dos modelos de classificação utilizados, juntamente com a adição de pesos para balanceamento de prioridades dessas tarefas.

4.1.2 Classificador

A Classificação é uma técnica de aprendizado de máquina que consiste em atribuir uma ou mais classes a um conjunto de dados, com base em suas características ou variáveis. Essa classificação pode ser binária (duas classes, 1 ou 0) ou multiclasse (três ou mais classes). Essa técnica é utilizada para identificar padrões e tendências nos dados, permitindo a tomada de decisão automatizada em diversos contextos (SCIKIT-LEARN, 2023). Neste trabalho, foi utilizada a classificação multiclasse (três classes para níveis de prioridades).

O Algoritmo 1 fornece um pseudo-código do modelo de classificação implementado neste trabalho.

Algoritmo 1: Algoritmo de Classificação

Entrada : Dados de treinamento rotulados
Saída : Modelo de classificação treinado
`TreinarModelo();`

Entrada : Dados não rotulados
Saída : Classificações previstas
foreach *instância de dado não rotulada* **do**
  `Prever();`
end

Opcional : Ajuste e otimização do modelo
;

Entrada : Dados de teste rotulados
Saída : Avaliação do modelo
`Avaliar();`

Entrada : Novos dados para classificação
Saída : Classificações dos novos dados
Utilizar o modelo treinado para classificar os novos dados;

Com o objetivo de comparação e seleção de um modelo de classificação apropriado para o mecanismo, foram implementados três classificadores usando aprendizado de máquina: K-Vizinhos Mais Próximos (KNN) (*K-Nearest Neighbors*), Máquina de Vetores de Suporte (SVM) (*Support Vector Machine*) e Regressão Logística. (Ver Seção 4.3).

Foi implementado um nó classificador e um modelo para alocação, com o objetivo de classificar as requisições de aplicações provenientes de dispositivos finais em diferentes níveis de prioridade (classes), permitindo assim uma alocação de recursos com base nas prioridades estabelecidas. Os dispositivos IoT enviam suas solicitações de serviço (dados) para o Nó classificador, também conhecido como Nó Gerente.

Os dados utilizados para treinamento e validação dos classificadores foram extraído do *Kaggle*[1]. Esses dados em questão possuem características valiosas, como latência, tipo de conexão e taxa da rede, e foram enriquecidos com outras informações como peso, distância com base na latência para simular as solicitações de dispositivos IoT com uma quantidade maior de variáveis. Essas informações incluem o tipo de serviço, recursos, tempo, latência, tipo de rede, taxa e coordenadas geográficas, as quais são utilizadas no cálculo da distância euclidiana para

[1] https://www.kaggle.com/datasets/suraj520/cellular-network-analysis-dataset

escolha do nós mais adequada para uma determinada solicitação além do mecanismo de controle de falha.

Esses dados são enviados para o nó classificador, onde são aplicados diferentes pesos aos recursos e atribuídas prioridades para cada tipo de requisição, com base em regras predefinidas. As regras de prioridade são definidas da seguinte forma:
Inicialmente, as prioridades das requisições foram classificadas em três níveis: Alto (2), Médio (1) e Baixo (0). Para estabelecer as regras de prioridade dos serviços, dois tipos de serviço foram categorizados em duas classes: serviços críticos (SC) e serviços não críticos (SNC), conforme apresentado a seguir.

- **Serviço Crítico**
  - I/O (*Input/Output*)
  - Tempo real
- **Serviço não crítico**
  - Consulta
  - Armazenamento

Os serviços da classe crítica recebem prioridade Alta (2), enquanto os serviços da classe não crítica são classificados como prioridade Média (1) ou Baixa (0).

De acordo com os estudos de (CHENG *et al.*, 2015; WANG *et al.*, 2019), o serviço I/O é considerado crítico devido à sua interação direta com o usuário final, o que tem um impacto significativo na experiência geral do usuário. Portanto, a necessidade de uma aplicação de um serviço I/O é um fator importante a ser considerado, sugerindo prioridade em relação a outros recursos. Da mesma forma, o critério para serviços em tempo real também se baseia no princípio usado para serviços I/O, pois esses serviços também influenciam a experiência do usuário e requerem respostas rápidas. Ao contrário, os serviços de consulta (que fornecem apenas informações ao usuário) e de armazenamento de dados (em que os dados são armazenados localmente na borda ou posteriormente enviados para a Nuvem) são considerados não críticos. As solicitações que exigem o uso da Nuvem geralmente são serviços com menor necessidade de resposta imediata ou que envolvem processamento mais complexo, considerando que a Nuvem está geralmente distante dos dispositivos finais (HONG; VARGHESE, 2019).

Além dessa distinção com base na criticidade das tarefas, também é considerado o tempo de requisição e os pesos definidos aos recursos necessários para atender cada requisição. Essa abordagem de adicionar pesos às prioridades dos recursos visa o balanceamento adequado

entre as prioridades dos diferentes tipos serviços e a disponibilidade dos recursos na borda, permitindo um funcionamento eficiente do sistema visto que tais recursos são limitados.

Nesse sentido, serviços que demandam maior quantidade de recursos, como RAM e CPU, têm suas prioridades ajustadas (recebem pesos mais altos) em relação a serviços que exigem apenas CPU. Essa decisão é tomada considerando que serviços prioritários que necessitam de múltiplos recursos podem causar extensas filas de espera e, consequentemente, levar à indisponibilidade do sistema. Enquanto outros serviços prioritários que utilizam apenas um recurso computacional podem ser atendidos mais rapidamente. Além disso, a complexidade das requisições pode aumentar o consumo de energia, contribuindo também para a indisponibilidade dos serviços. Portanto, é necessário atribuir pesos aos recursos para lidar de maneira adequada com esses diferentes situações. Esse aspecto leva em consideração a disponibilidade em tempo real dos recursos nos nós da Borda.

Sendo assim, a regra de atribuição de peso aos recursos é a seguinte:

$$Requisicao_{peso} = (0.3 * Q_R) + (N * T_R) \begin{cases} Q_R = 0, & \text{Requisições necessitam um recurso (R = 1)} \\ Q_R = 1 & \text{caso contrário, (R > 1)} \end{cases} \tag{4.1}$$

O valor N da equação é uma variável de adição que assume diferentes valores dependendo do tipo de serviço: SNC (Serviço Não Crítico) com valores de 0.5 e 0.7 e SC (Serviço Crítico) com valores de 0.1 e 0.3. Esses valores variam com base no tempo decorrido desde a solicitação de cada serviço. Quanto maior o valor resultante da Equação 1, mais prioritária tende a ser a solicitação.

O valor de T_R depende do tipo de conexão, que pode variar entre LTE, 3G, 4G e 5G. Quanto mais alta a taxa de transmissão, a latência tende a ser menor. As requisições com um tipo de rede com uma taxa de transmissão menor, recebem um valor maior de T_R. Esses valores são definidos como 0.3 (LTE), 0.2 (3G) e 0.1 (4G e 5G). Essa abordagem visa priorizar as requisições provenientes de conexões mais lentas, dando a elas um peso maior (em termos de prioridade), a fim de compensar a latência. Dessa forma, é atribuído um peso elevado às requisições com maior prioridade, a fim de compensar a latência. Assim, leva-se em consideração as diferentes características dos serviços e conexões, possibilitando um melhor desempenho.

Os valores dos pesos utilizados foram determinados empiricamente por meio de experimentos. Essa abordagem prática mostrou-se satisfatória para alcançar um balanceamento

adequado entre o tempo decorrido, a quantidade de recursos, os tipos de serviços e os tipos de conexões. Através da atribuição de pesos, é possível ajustar a prioridade das requisições, de modo que os serviços críticos recebam a devida atenção.

É importante ressaltar que os pesos podem variar de acordo com o contexto e as necessidades específicas do problema abordado. Realizar experimentos e ajustar os pesos com base nos resultados observados é uma abordagem válida e comumente adotada na prática. Isso permite adaptar o modelo às características e restrições particulares do sistema, buscando maximizar o desempenho e a eficiência das requisições.

4.1.3 Gerencia de Recursos com Alocação

No mecanismo de alocação de recursos proposto neste trabalho, é empregada a técnica de classificação de prioridade das requisições dos clientes descrita na seção anterior. Ao estabelecer prioridades para as requisições, o sistema pode direcionar os recursos de forma mais aprimorada, garantindo um atendimento imediato às tarefas mais relevantes e urgentes. Isso contribui para melhorar o desempenho das rede, economizando os recursos disponíveis.

Para determinar o nó de borda mais adequado para atender à requisição, considera-se além do monitoramento em tempo real do uso dos recursos, a distância dos nós de borda em relação ao classificador. Neste estudo, a Equação 4.2 é utilizada para calcular a Distância Euclidiana entre o nó de borda e o classificador, considerando um plano bidimensional:

$$Distancia = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2} \tag{4.2}$$

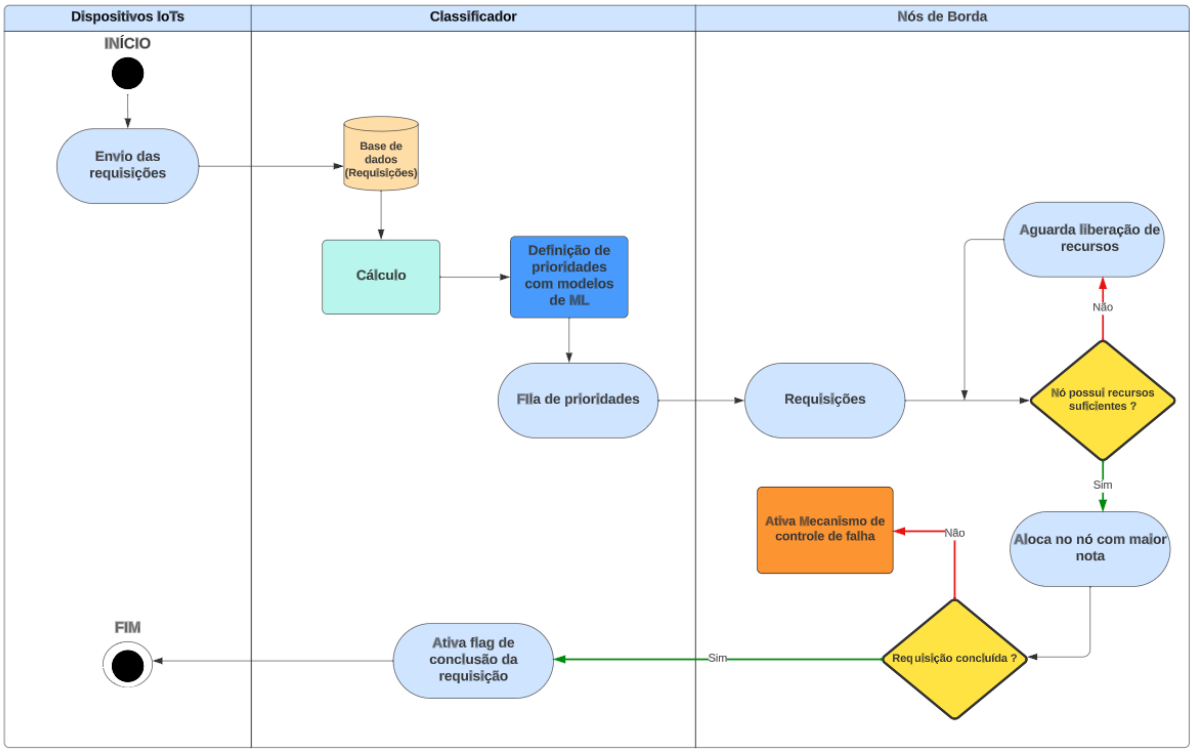
Ao usar um critério que combina o percentual de utilização dos recursos no nó de borda e a distância em relação ao classificador, é atribuída uma nota que servirá como critério para a seleção do nó (COSTA *et al.*, 2020). Quanto maior a nota, maior será a chance de escolher o nó. A Equação 4.3 é utilizada para calcular a nota:

$$Nota_i = (0.3 * Distancia_no) + (0.7 * Recursos_no\%) \tag{4.3}$$

Dessa forma, o nó mais adequado para a execução de uma determinada tarefa prioritária é selecionado, com o objetivo de escolher o nó mais capacitado e mais próximo, buscando minimizar a latência.

Na Figura 3 é apresentado um fluxograma que representa o processo completo desde de o processo de requisição até a alocação da tarefa nos nós de borda.

Figura 3 – Fluxo da Proposta



Fonte: (Elaborado pelo Autor)

4.2 Mecanismo de Controle de Falha

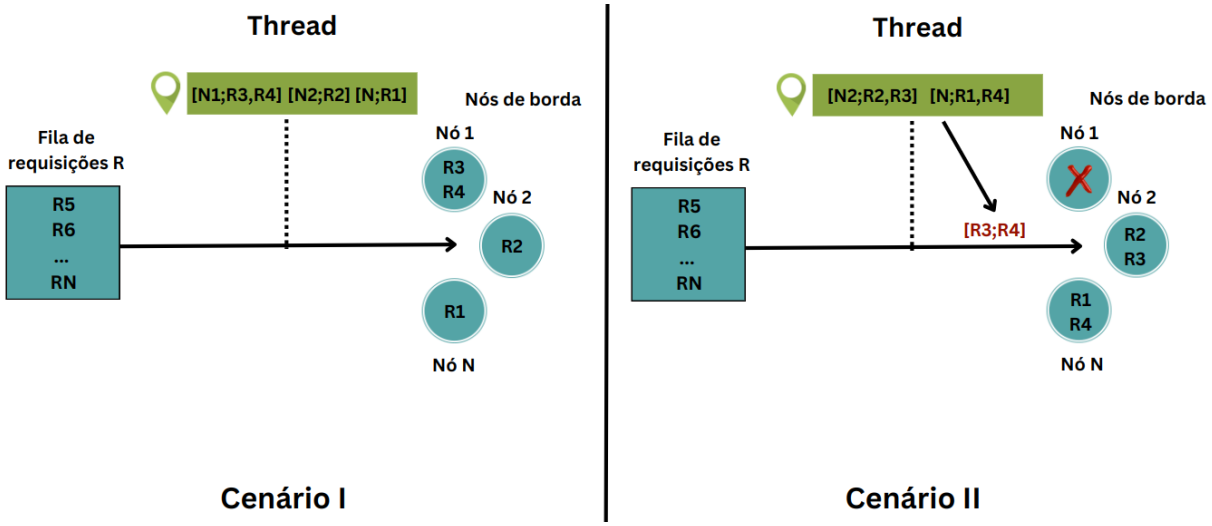
Um ponto crucial e de suma importância nesse trabalho é o controle de falhas. Controlar as falhas em ambientes de computação pode incluir estratégias como redundância, monitoramento contínuo, resiliência de rede, autocorreção e manutenção regular. Esses procedimentos visam garantir a disponibilidade e confiabilidade dos serviços, detectar falhas precocemente, garantir conectividade e capacidade de recuperação em caso de interrupção, além de manter os sistemas atualizados e funcionando corretamente.

Neste trabalho foi implementado um mecanismo de controle e gerenciamento de falhas, baseado em monitoramento contínuo e redundância, com o objetivo de manter a eficiência no processamento das requisições nos nós de borda. Ao definir as prioridades, uma *Thread* é designada para calcular a pontuação de cada nó de borda, a fim de determinar qual será escolhido para alocação. Uma vez feita a seleção, as requisições são alocadas nos respectivos nós correspondentes.

Essa *Thread* assume a responsabilidade de acompanhar as requisições alocadas em um determinado momento e mantê-las em um estado de *checkpoint*, manter uma cópia desses dados, até que sejam completamente resolvidas. Esse estado de *checkpoint* é crucial para o controle de falhas caso algum nó de borda apresente problemas. Por meio do monitoramento contínuo dos estados dos nós de borda, a *Thread* identifica se um determinado nó está inativo. Se essa situação for detectada, com base nas informações armazenadas no estado de *checkpoint*, a *Thread* é capaz de realocar as requisições que estavam no nó falho, desde que não tenham sido finalizadas, para outros nós de borda que possuam recursos suficientes para concluí-las. O estado de *checkpoint* é então atualizado com as novas informações, permitindo que o sistema se adapte e continue processando as requisições mesmo em caso de falha.

A Figura 4 ilustra o funcionamento do mecanismo de controle de falhas. No cenário I é possível observar as requisições alocadas nos nós de borda, enquanto o estado de *checkpoint* armazena essas informações para garantir que elas não se percam, até sua completa execução. Já no cenário II, ocorre a falha do nó 1 e a Thread assume o controle para realocar as requisições que estavam originalmente atribuídas a esse nó para outros nós de borda na rede. Após essa realocação, o estado de *checkpoint* é atualizado com as novas informações.

Figura 4 – Fluxo de controle de Falha



Fonte: (Elaborado pelo Autor)

Esses dois cenários ilustram como o mecanismo de verificação e gerenciamento de falhas funciona em ação. O estado do ponto de verificação desempenha um papel fundamental em manter o controle sobre as solicitações alocadas e permitir a recuperação no caso de falha de um nó de borda. Com essa abordagem, o sistema pode garantir uma operação contínua, mesmo

diante de falhas, e garantir que todas as solicitações sejam processadas adequadamente.

4.3 Implementação do Mecanismo Proposto

Utilizando *Docker* foi possível usar o conceito de conteinerização para a criação da arquitetura apresentada neste trabalho. A utilização de contêineres *Docker* foi fundamental para o desenvolvimento deste estudo, pois possibilitou a criação de um ambiente de borda no qual cada nó de borda é executado em seu próprio contêiner independentemente. Essa abordagem oferece diversas vantagens, como a facilidade de migração de aplicações entre dispositivos com diferentes configurações de hardware. Além disso, os contêineres *Docker* permitem o isolamento completo dos sistemas operacionais, viabilizando a execução simultânea de múltiplos nós de borda na mesma máquina. Isso significa que os recursos de cada nó são isolados e consumidos de forma independente, mesmo quando tarefas estão sendo executadas em outros nós.

Através da API *'docker stats'* é possível obter informações sobre a utilização de recursos de cada nó de borda (container) individualmente. Essas informações são utilizadas para alocar as requisições nos nós de borda, permitindo uma alocação mais dinâmica dos recursos com base nas necessidades das aplicações, das prioridades definidas e dos recursos disponíveis.

Para a classificação das prioridades, foram realizadas os passos a seguir. Após conduzir uma série de testes com diversos classificadores, avaliando suas métricas de desempenho, como acurácia, precisão e possível ocorrência de *overfitting*, foram escolhidos três modelos em particular: KNN, SVM e Regressão Logística. Esses modelos são amplamente estudados e têm sido consistentemente bem-sucedidos em diversas pesquisas, garantindo resultados satisfatórios, conforme (PASSOS *et al.*, 2021), (SILVA *et al.*, 2021) e (RUSMAN *et al.*, 2019).

Os classificadores utilizam um conjunto de dados extraído do *Kaggle*[2]. Kaggle é uma plataforma online de ciência de dados e aprendizado de máquina, ela permite acessar diversos conjuntos de dados. Os dados em questão possuem características valiosas, como latência, tipo de conexão e taxa da rede, e foram enriquecidos com outras informações para simular as solicitações de dispositivos IoT. Essas informações incluem o tipo de serviço, recursos, tempo, latência, tipo de rede, taxa e coordenadas geográficas, as quais são utilizadas no cálculo da distância euclidiana.

Após os testes com esse conjunto de requisições, na etapa de alocação, apenas o classificador que obteve os melhores resultados foi selecionado. As tarefas são priorizadas

[2] https://www.kaggle.com/datasets/suraj520/cellular-network-analysis-dataset

pelo nó classificador, que gera uma fila de prioridades levando em consideração os recursos necessários para a execução dessas tarefas nos nós de borda. Avaliando as métricas apresentadas na Seção 4.4, obtemos a fila de prioridade das requisições que serão alocadas nos nós de borda. Através dessa fila, as requisições serão atendidas de acordo com suas prioridades.

Nesse momento, uma *Thread* na simulação é usada para selecionar um nó de borda adequado para a execução de cada tarefa, considerando dois principais critérios: a quantidade de recursos disponíveis e a distância do nó. Para isso, é atribuída uma nota a cada nó de borda, seguindo uma abordagem semelhante à apresentada em (COSTA *et al.*, 2020), na qual é levado em consideração a quantidade de recursos consumidos e a distância do nó (descrito na Seção 4.1.3). O nó que obtiver a maior nota é selecionado para executar a respectiva requisição, garantindo uma alocação eficiente e baseada em critérios de recursos e proximidade.

A primeira etapa considerada é a quantidade de recursos disponíveis nos nós de borda. É crucial garantir que um nó de borda tenha recursos suficientes para processar a tarefa adequadamente. Portanto, são priorizados os nós que possuem uma quantidade adequada de recursos disponíveis, evitando sobrecarga e garantindo uma execução eficiente das tarefas.

Na segunda etapa, é realizado o cálculo da nota, que representa a distância do nó de borda em relação ao classificador. Esse cálculo é baseado no princípio da menor latência, ou seja, leva em consideração o tempo de comunicação entre o nó classificador e os nós de borda, conforme mostrado na Equação 4.6 na Seção 4.1.3. Dessa forma, os nós de borda mais próximos do classificador recebem notas mais altas, o que aumenta suas chances de serem selecionados. Isso resulta em uma redução do tempo de resposta e proporciona uma melhor experiência para o usuário. Ao combinar essas duas etapa, é selecionado o nó de borda mais adequado para a execução de cada tarefa.

A comunicação entre os nós de borda e o classificador é estabelecida usando o protocolo leve *Message Queuing Telemetry Transport (MQTT)* (MQTT.ORG, 2023), projetado para comunicação dispositivo a dispositivo em redes de baixa potência e largura de banda restrita. É amplamente utilizado na Internet das Coisas (IoT) devido à sua simplicidade, eficiência e capacidade de suportar comunicação assíncrona e bidirecional, e adequação para comunicação entre dispositivos IoT (SASAKI; YOKOTANI, 2019)(MQTT.ORG, 2023). O MQTT permite o transporte de mensagens através do modelo de publicação e assinatura (BAYıLMış *et al.*, 2022). Os dispositivos podem postar mensagens sobre tópicos específicos, enquanto outros dispositivos podem se inscrever nesses tópicos e receber mensagens específicas desse tópico. Isso permite

que os dispositivos IoT se comuniquem, enviem e recebam informações relevantes para suas necessidades de maneira eficaz.

Em um cenário de alocação de recursos, cada nó de borda possui um tópico específico, e as requisições alocadas para um determinado nó de borda são enviadas a esse tópico, permitindo que o nó de borda se inscreva nesse tópico e receba as requisições direcionadas a ele.

Em um cenário real, os nós de borda estão sujeitos a diversas falhas, que vão desde dados incorretos até a indisponibilidade do próprio nó. Por essa razão, foi desenvolvido um mecanismo de controle no qual, caso um nó de borda falhe ou fique offline, todas as suas tarefas serão realocadas para os outros nós de borda disponíveis. Esse mecanismo utiliza a verificação do estado dos contêineres *Docker* para garantir a disponibilidade e continuidade das operações.

4.4 Métricas de Avaliação

Para modelos de aprendizado de máquina, várias métricas de desempenho são utilizadas como indicadores da qualidade dos algoritmos de classificação. Alguns exemplos dessas métricas são a acurácia, recall, F1-score, precisão e outras. Todas essas métricas são derivadas da análise da matriz de confusão, que é uma ferramenta importante para avaliar o desempenho dos modelos. Através da matriz de confusão, pode-se analisar a quantidade de verdadeiros positivos, falsos positivos, verdadeiros negativos e falsos negativos, e com base nesses valores, calcular as diferentes métricas de desempenho mencionadas (VIEIRA, 2021). A seguir, são detalhadas as métricas de avaliação utilizadas nessa dissertação.

4.4.1 Matriz de Confusão

A matriz de confusão é uma ferramenta utilizada na estatística computacional para analisar os resultados de algoritmos de Aprendizado de Máquina. Ela representa de forma organizada os resultados positivos e negativos, mostrando os acertos e erros do modelo. Através da matriz de confusão, é possível calcular métricas precisas e obter dados matemáticos relevantes (FACELI *et al.*, 2021). A matriz de confusão é representada da seguinte forma:

$$\begin{bmatrix} TP & FP \\ FN & TN \end{bmatrix}$$

– TP (*True Positive*) representa as classificações corretas positivas.

– FP (*False Positive*) representa as classificações incorretas positivas.
– FN (*False Negative*) representa as classificações incorretas negativas.
– TN (*True Negative*) representa as classificações corretas negativas.

Dessa forma, utilizando a matriz de confusão, é possível calcular diversas métricas que representam diferentes características do modelo. Algumas dessas métricas incluem a acurácia, precisão (*precision*), recall e F1-score. Essas métricas fornecem informações importantes sobre o desempenho e a qualidade do modelo de classificação (FACELI *et al.*, 2021) (MARIANO, 2021).

4.4.2 Acurácia

A acurácia, em um modelo de classificação é uma métrica que mede a taxa de acerto do modelo em relação ao número total de amostras. Em outras palavras, a acurácia mede a proporção de previsões corretas feitas pelo modelo em relação ao total de amostras. A Equação 4.4 descreve o cálculo da acurácia:

$$Acuracia = \frac{TP + TN}{TP + TN + FP + FN} \tag{4.4}$$

– TP representa os verdadeiros positivos (amostras corretamente classificadas como positivas).
– TN representa os verdadeiros negativos (amostras corretamente classificadas como negativas).
– FP representa os falsos positivos (amostras erroneamente classificadas como positivas).
– FN representa os falsos negativos (amostras erroneamente classificadas como negativas).

A acurácia é uma métrica amplamente usada para avaliar o desempenho de um modelo de classificação porque fornece uma medida geral de quão bem o modelo funciona corretamente na classificação de amostras. No entanto, a acurácia pode não fornecer resultados concretos em certos casos, especialmente se houver classes desbalanceadas no conjunto de dados.

4.4.3 F1-score

O F1-score é uma métrica amplamente utilizada para avaliar o desempenho de modelos de classificação. Ele combina precisão e recuperação em uma única métrica para

fornecer uma medida equilibrada do desempenho do modelo. A Equação 4.5 descreve o cálculo do F1-score:

$$F1\text{-}score = 2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}} \tag{4.5}$$

A pontuação F1 varia de 0 a 1, onde um valor próximo de 1 indica um bom equilíbrio entre precisão e revocação e, portanto, um bom desempenho do modelo de classificação. Valores menores indicam uma baixa capacidade do modelo em realizar classificações corretas.

A pontuação F1 é especialmente útil quando as classes estão desbalanceadas e não queremos dar mais importância a uma classe em detrimento de outra. Ele fornece uma métrica que leva em consideração a capacidade de identificar corretamente amostras positivas e a capacidade de evitar falsos positivos.

Para um problema multiclasse, calcula-se a média para obter uma medida geral do desempenho do modelo em todas as classes. Existem diferentes métodos de média que podem ser utilizados, como a média aritmética simples (média macro) ou a média ponderada pelos números de amostras em cada classe (média ponderada).

4.4.4 Precisão

A precisão é uma das métricas mais comumente utilizadas para avaliar modelos de classificação. Essa métrica é definida como a razão entre a quantidade de exemplos classificados corretamente como positivos e o total de exemplos classificados como positivos, conforme a Equação 4.6 a seguir:

$$\text{Precisão} = \frac{\text{Verdadeiros Positivos}}{\text{Verdadeiros Positivos} + \text{Falsos Positivos}} \tag{4.6}$$

É importante destacar que a precisão enfatiza os erros de falso positivo. Pode-se entender a precisão como uma expressão matemática para responder à pergunta: dentre os exemplos classificados como positivos, quantos realmente são positivos?

4.4.5 Recall

O recall, também conhecido como revocação ou sensibilidade, é uma métrica que mede a capacidade do modelo em prever corretamente as classes positivas reais. Essa métrica é

calculada como a taxa entre os verdadeiros positivos previstos pelo modelo e a quantidade total de exemplos que foram realmente marcados como positivos. A fórmula para o cálculo da recall e exibida na Equação 4.7:

$$\text{Recall} = \frac{\text{Verdadeiros Positivos}}{\text{Verdadeiros Positivos} + \text{Falsos Negativos}} \tag{4.7}$$

O recall revela quantas das classes previstas pelo modelo estão corretas, ou seja, quantos exemplos positivos reais foram corretamente identificados como positivos. É uma medida importante para avaliar o desempenho do modelo na detecção de exemplos positivos e é especialmente útil quando os falsos negativos têm consequências significativas.

É importante destacar que ao avaliar um modelo de classificação, é fundamental considerar as peculiaridades de cada métrica. Em geral, não deve-se considerar uma métrica melhor ou pior do que outra. Em vez disso, é preciso analisar o problema em questão e escolher a(s) métrica(s) que melhor se ajusta(m) a ele. Cada métrica tem sua relevância e interpretação específica, e sua seleção depende das necessidades e características do problema analisado.

4.4.6 Técnicas Experimentais

Para auxiliar na avaliação dos sistemas do cenário avaliado, são realizados alguns testes mais aprofundados para a tomada de decisão sobre o melhor sistema a ser utilizado. A seguir são apresentadas as técnicas utilizadas na condução do experimento.

O teste t de Student, trata-se de um método estatístico que permite realizar comparações simultâneas entre dois grupos , e testar se elas são iguais ou diferentes (GOMES, 2009). A análise é feita através de duas hipóteses, sendo elas:

$H_0 : \beta 1 == 0$ (Não há influência sobre a resposta)

$H_1 : \beta 1 = 0$ (Há influência do sistema sobre o a resposta)

Na hipótese nula, H_0, as médias obtidas são iguais, isto significa que, estatisticamente falando, os tratamentos são iguais. Na hipótese alternativa, H1, as médias obtidas são diferentes, o que significa que um dos sistemas testados apresentou diferenças estatísticas dos outros sistemas (ROCHA; JúNIOR, 2018).

A fórmula matemática do teste t de Student é dada pela equação 4.8:

$$t = \frac{\bar{X}_1 - \bar{X}_2}{\sqrt{\frac{s_1^2}{n_1} + \frac{s_2^2}{n_2}}} \tag{4.8}$$

onde:

- $\bar{X}_1$ e $\bar{X}_2$ são as médias das duas amostras. - s_1^2 e s_2^2 são as variâncias das duas amostras. - n_1 e n_2 são os tamanhos das amostras.

O processo básico do teste t de Student envolve o cálculo do valor t, que é uma medida da diferença entre as médias da amostra dividida pelo erro padrão dessa diferença. A distribuição t segue a distribuição t de Student, que é uma distribuição estatística que leva em consideração tamanhos de amostra e graus de liberdade.

Interpretar o resultado de um teste t envolve comparar o valor t calculado com o valor crítico da distribuição t ou, mais comumente, comparar o valor p (probabilidade) associado ao valor t calculado com um nível de significância (geralmente 0,05). ou 5%)). Se o valor-p for menor que o nível de significância, você rejeita a hipótese nula (que diz que não há diferença significativa) e conclui que há uma diferença significativa entre as médias.

Em resumo, o teste t de Student é uma ferramenta fundamental na análise estatística para ajudar a determinar se as diferenças observadas nas médias entre dois grupos são reais ou apenas devidas ao acaso.

5 EXPERIMENTOS

Foram realizados diversos experimentos utilizando uma máquina física com sistema operacional Linux, equipada com um processador i7-7560U e 8 GB de memória RAM. A máquina foi conectada a uma rede local via Wi-Fi e foram utilizados contêineres para implementar a arquitetura de computação em borda.

Um *script* em *Python* foi desenvolvido para popular o conjunto de dados utilizado, adicionando informações para a simulação de requisições de dispositivos IoT, que eram enviadas para o contêiner responsável pela classificação (nó Gerente). Essas requisições continham informações sobre o tipo de serviços, os recursos necessários, a latência, distância dos nós, tempo decorrido da solicitação, tipo de conexão e taxa de transmissão da rede.

Nos experimentos, o fluxo abaixo é seguido, variando o número de requisições de tarefas para os nós de borda:

1. Os dados das requisições/tarefas são enviadas para o nó classificador.
2. Esses dados servem de entrada para o classificador calcular as prioridades a cada uma das requisições.
3. Ocorre um monitoramento dos recursos dos nós de borda, enviados para o nó gerente que contem o classificador. Dessa forma, a tabela de prioridades é ajustada de acordo com os recursos disponíveis.
4. A alocação das requisições é feita com base na fila de prioridades, dos recursos que essas tarefas necessitam e dos recursos disponíveis nos nós de borda naquele momento, além da distância euclidiana dos nós em relação ao nó classificador.
5. Ocorre uma verificação em tempo de execução no estado dos containers, para caso haja falha (estado *offline*) as requisições possam ser remanejadas para outros nós de modo a não comprometer a execução das tarefas.

Para execução dos experimentos, a simulação é iniciada com base em todas as configurações definidas no arquivo de *Docker Compose*: contêineres, quantidade de nós de borda, o tipo de comunicação entre eles, dentre outras configurações.

Os experimentos realizados permitem avaliar os modelos de classificação apresentados (KNN, SVM e Regressão Logística) e selecionar aquele que obtiver os melhores resultados como o modelo de classificação de prioridades. Além disso, permitem avaliar a eficiência do gerenciamento de recursos usando o mecanismo proposto, alocando as requisições nos dispositivos de borda, com base nas regras de prioridades de tarefas estabelecidas, e avaliando o consumo

dos recursos por parte desses nós. Os resultados obtidos são discutidos na Seção 6 (Resultados e Discussões).

5.1 *Design* Experimental

Foram realizados seis experimentos (A, B, C, D, E e F) com diferentes cenários. O experimento A consiste na avaliação dos modelos de classificação (KNN, SVM e Regressão Logística) apresentados para o nó classificador de prioridade, com base nas métricas descritas na Seção 4.4 (Matriz de Confusão, Acurácia, F1-score, Precisão, *Recall*). Foi selecionado o modelo que obteve os melhores resultados de classificação.

No Experimento B foram realizadas alocações de requisições para dispositivos de borda com base nas regras predefinidas, utilizando um classificador para determinar as prioridades. Foram utilizados três nós de borda, variando entre 500 e 1000 requisições. Em seguida, foi avaliado o consumo de CPU pelos nós de borda e foi analisado o comportamento do mecanismo de gerenciamento de recursos em comparação com uma abordagem em que as tarefas são atribuídas selecionando o nó mais próximo do classificador (baseado na distância)Foi monitorado e analisado o consumo de recursos pelos nós de borda em ambos os ambientes.

No Experimento C foram empregados seis nós de borda, com a quantidade de requisições recebidas variando entre 500 e 1000. Realizou-se uma comparação em um ambiente em que a arquitetura proposta é adotada e em outro ambiente que não incorporou o mecanismo proposto neste estudo. A alocação das requisições era realizada exclusivamente com base no critério de latência, selecionando o nó mais próximo ao classificador.

Os experimentos D e E seguiram os mesmos cenários dos experimentos B (3 nós e borda e requisições variando entre 500 e 1000) e C (6 nós e borda e requisições variando entre 500 e 1000), respectivamente. O objetivo destes experimentos é avaliar o consumo de memória com quantidade diferentes de nós de borda em cada experimento.

No Experimento F, uma falha é forçada para simular o desligamento de um nó de borda, visando demonstrar o processo de atuação do mecanismo de falhas proposto diante dessa situação. Dessa forma, o objetivo desse experimento é avaliar como o sistema se comporta frente a uma falha nos nós, analisando o consumo de recursos nos nós de borda. Essa avaliação permite entender como o sistema reage, se é capaz de se adaptar e redistribuir as tarefas de forma eficiente, garantindo a continuidade do serviço mesmo com a indisponibilidade temporária de um ou mais nós de borda.

Nos experimentos realizados, foram implementadas estratégias para balancear o uso dos recursos. Quando um recurso monitorado por um nó de borda atinge 85% de utilização, o nó de borda deixa de receber novas requisições até que o recurso seja liberado. Além disso, é feita uma verificação das condições dos nós de borda.

Abaixo segue um resumo dos objetivos dos experimentos realizados:

(i) Avaliar os modelos de classificação implementados.

(ii) Comparar a proposta com a abordagem tradicional de escolher o nó mais próximo.

(iii) Avaliar a proposta em relação ao uso adaptativo de recursos, considerando diferentes quantidades de requisições.

(iv) Verificar o comportamento da proposta em caso de falha em algum nó de borda.

Cada experimento abordou esses objetivos específicos em diferentes configurações para obter uma visão abrangente do desempenho e eficácia da proposta.

Os objetivos de cada experimentos, cenários e métricas são descritos na Tabela 2.

Tabela 2 – Resumo dos Experimentos Realizados

Experimento	Cenário	Objetivos	Avaliação
A	—	Avaliar os modelos de classificação	Métricas de validação dos modelos de classificação
B	I - 3 nós: 500 requisições II - 3 nós: 500 requisições III - 3 nós: 1000 requisições IV - 3 nós: 1000 requisições	Comparar o mecanismo proposto com a alocação considerando apenas distância *(I e III usam o mecanismo)*	Consumo de recursos dos nós (CPU)
C	I - 6 nós: 500 requisições II - 6 nós: 500 requisições III - 6 nós: 1000 requisições IV - 6 nós: 1000 requisições	Comparar o mecanismo proposto com a alocação considerando apenas distância *(I e III usam o mecanismo)*	Consumo de recursos dos nós (CPU)
D	I - 3 nós: 500 requisições II - 3 nós: 500 requisições III - 3 nós: 1000 requisições IV - 3 nós: 1000 requisições	Comparar o mecanismo proposto com a alocação considerando apenas distância *(I e III usam o mecanismo)*	Consumo de recursos dos nós (Memória)
E	I - 6 nós: 500 requisições II - 6 nós: 500 requisições III - 6 nós: 1000 requisições IV - 6 nós: 1000 requisições	Comparar o mecanismo proposto com a alocação considerando apenas distância *(I e III usam o mecanismo)*	Consumo de recursos dos nós (Memória)
F	6 nós: 1000 requisições	Avaliar o mecanismo de falha	Realocação das requisições

Fonte: elaborada pelo autor.

6 RESULTADOS E DISCUSSÕES

Neste capítulo são apresentados os resultados obtidos nos experimentos realizados. Primeiramente é realizada uma análise sobre os resultados obtidos por cada modelo de classificação (KNN, SVM e Regressão Logística), justificando a escolha do modelo mais apropriado para execução do experimento com a alocação das requisições nos nós de borda. São avaliados os resultados obtidos com as métricas apresentadas para validação de algoritmos de classificação. Em seguida, o modelo classificatório com os melhores resultados é implementado e é conduzida uma análise do consumo de CPU e Memória RAM durante a execução do experimento, considerando diferentes quantidades de nós de borda (3 e 6) e de requisições (500 e 1000). Realizou-se uma comparação entre um ambiente com a proposta dessa dissertação implementada da alocação utilizando a abordagem proposta e um ambiente com alocação por distância, (sem utilizar o mecanismo proposto), na qual todo o processo de alocação e escolha do nó se dá pela menor distância, nesse caso, as requisições são enviadas para os nós de borda mais próximos.

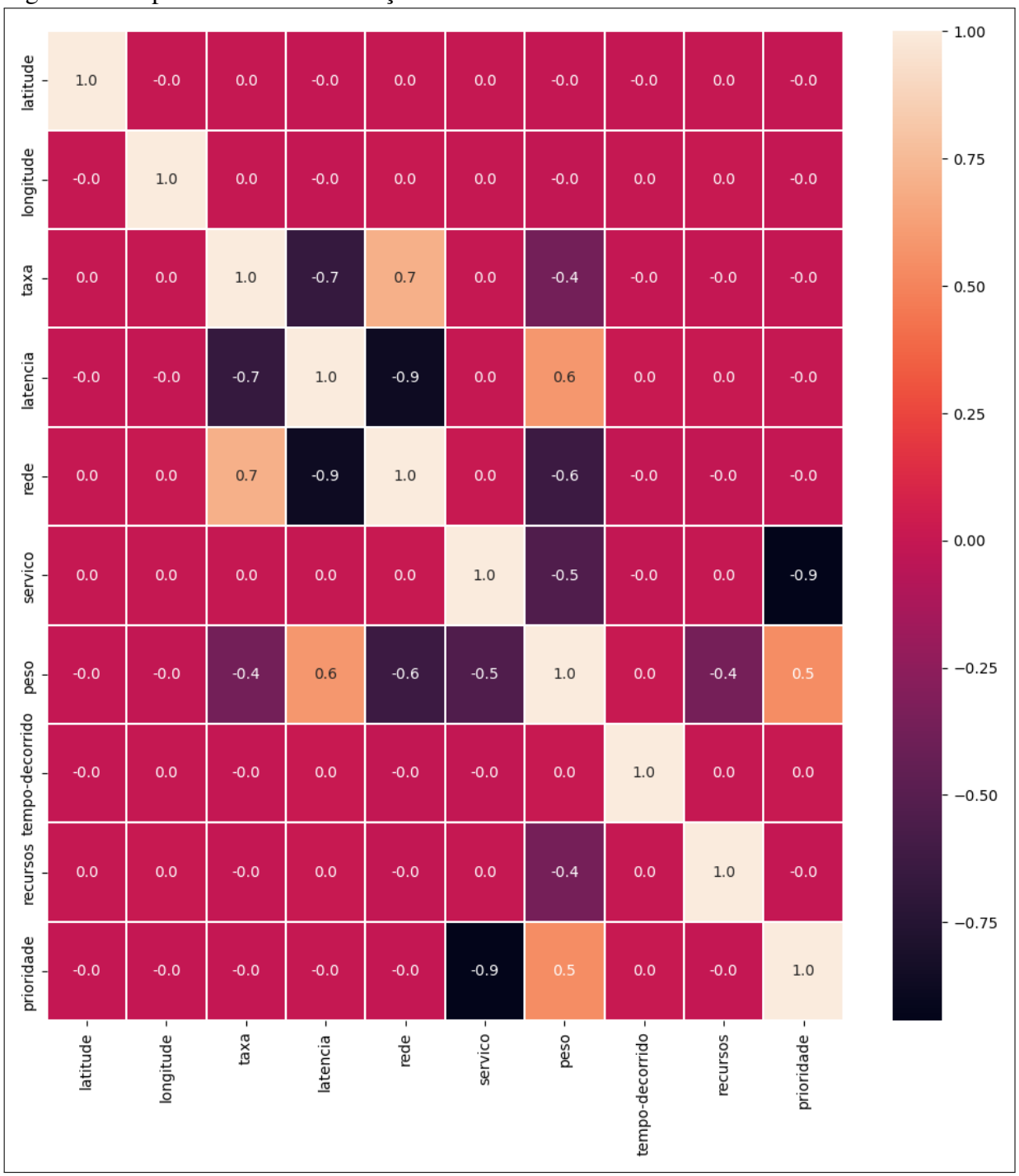
6.1 Resultados do Experimento A

Esta seção tem como objetivo apresentar os resultados dos experimentos realizados para validar o classificador implementado, bem como a definição de prioridade e alocação de recursos dos nós de borda usando as tabelas de prioridade geradas pelo classificador.

Os *atributos* utilizadas provenientes das requisições R_i, tem influência nos resultados do modelo de classificação, sendo elas latitude, longitude, rede, taxa de transmissão, recursos, prioridade, pesos. Primeiramente, é realizado um estudo sobre a relação entre as variáveis. A Figura 5 apresenta o gráfico de correlação das variáveis. Um gráfico de correlação variável é uma representação visual que mostra a relação entre diferentes variáveis em um conjunto de dados. Ele pode ser usado para identificar se existe uma relação linear entre as variáveis, bem como para avaliar a força e a direção dessa relação. Um gráfico de correlação variável pode ajudar a identificar relacionamentos entre variáveis e fornecer informações sobre sua dependência ou independência dentro de um conjunto de dados.

Com base na Figura 5, podemos observar que existem poucas correlações entre as variáveis, sendo observadas relações fortes apenas nos casos de rede, taxa e latência. Uma conexão de rede melhor parece contribuir para uma taxa de transferência maior e uma latência menor. Além disso, é possível notar uma forte correlação entre as variáveis serviço e prioridade.

Figura 5 – Experimento A - Correlação entre as variáveis



Fonte: elaborada pelo autor.

Isso sugere que, quando o serviço muda, a prioridade também se altera.

O gráfico também é útil para detectar *outliers*, identificando pontos que se distanciam significativamente da tendência principal do conjunto de dados. *Outliers* podem representar casos atípicos ou excepcionais que podem requerer uma análise mais aprofundada. No entanto, é importante observar que essas conclusões são baseadas na análise visual do gráfico de correlação. Para uma análise mais precisa e estatisticamente sólida, pode-se calcular um coeficiente de

correlação, por exemplo, o coeficiente de correlação de *Pearson*, para quantificar a intensidade das relações entre as variáveis e determinar a significância estatística dessas correlações.

No experimento, as requisições R_i de dispositivos utilizando o conjunto de dados mencionado, contendo as *features* previamente apresentadas, são inseridas no classificador. O conjunto de dados foi dividido em conjuntos de treinamento, teste e validação, com 70% dos dados sendo utilizados para treinar o classificador e os outros 30% para teste. A validação foi realizada utilizando um conjunto de 16829 requisições, que corresponde ao tamanho total do conjunto de dados. As regras estabelecidas anteriormente (4) foram aplicadas ao classificador para que ele aprendesse a classificar corretamente a prioridade de cada tipo de serviço, com base na latência, tipo de rede, localização e quantidade de recursos necessários. A seguir, são apresentados os resultados para cada modelo de classificação implementado.

6.1.1 Matrizes de Confusão

Para validar os resultados e para fins de comparação, as matrizes de confusão para cada modelo foram geradas, conforme mostrado na Figura 6.

A seguir, segue uma análise detalhada da matriz de cada modelo e das informaçções que se pode extrair desses resultados.

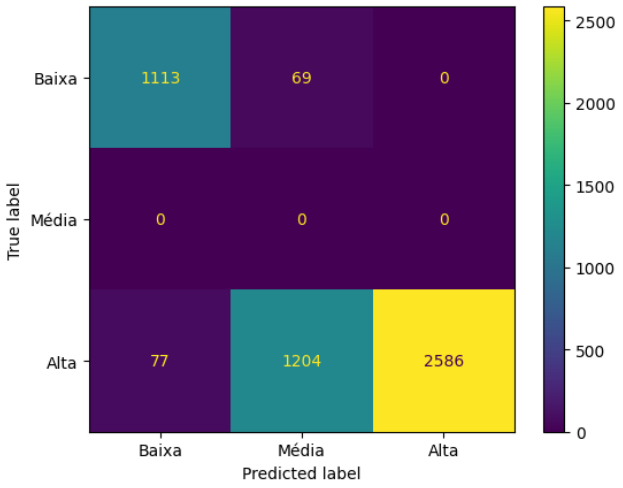
6.1.2 SVM - Support Vector Machine

O SVM apresentou resultados inferiores aos outros dois modelos de classificação (Regressão Logística e KNN).

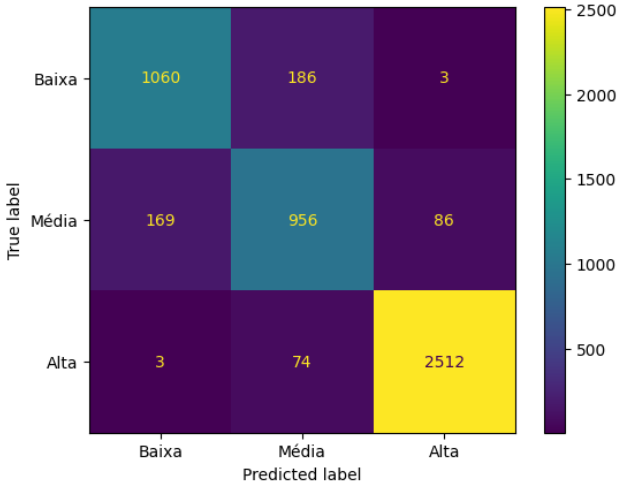
A análise realizada sobre o desempenho do modelo de classificação usando SVM revelou algumas informações importantes. Inicialmente, a matriz de confusão permitiu a visualização da distribuição de acertos e erros para cada classe. Observou-se que o modelo teve dificuldades em classificar corretamente as classes com prioridade definida como "Média", não tendo alcançado nenhuma classificação precisa para essa categoria. Por outro lado, as classes com prioridades "Alta"e "Baixa"foram mais acertadamente classificadas, embora tenha ocorrido um número significativo de erros para a classe "Baixa".

A análise das métricas de avaliação também trouxe *insights* relevantes. A acurácia de aproximadamente 73% evidenciou que o modelo acertou cerca da metade das previsões feitas, mas cometeu erros em aproximadamente 44% dos casos. Isso indica que o modelo não está alcançando um nível de precisão desejável em suas previsões.

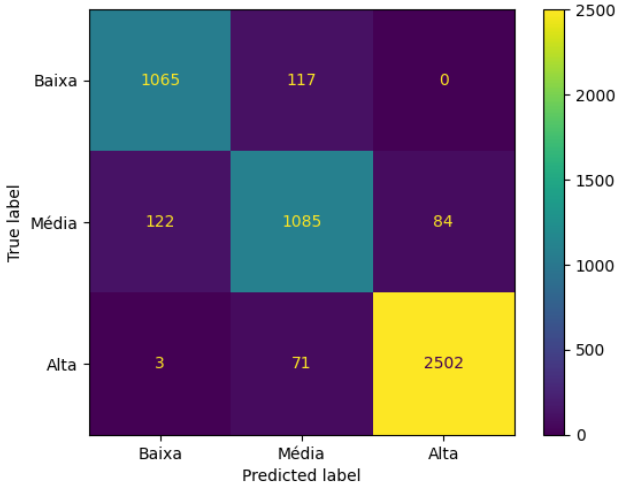
Figura 6 – Experimento A - Comparação das matrizes de confusão dos modelos



a) Matriz de Confusão - SVM



b) Matriz deconfusão - Regressão Logística



c) Matriz de Confusão - KNN

Os valores obtidos para precisão indicam que das vezes em que o modelo fez uma previsão positiva, ele estava correto em cerca de 63% dos casos, um valor relativamente baixo, já para um recall de 53% indica que o modelo não está sendo tão eficaz. Além disso, o valor médio do F1-score atingindo 56% mostrou que o modelo conseguiu encontrar um equilíbrio moderado entre precisão e recall, sendo capaz de identificar corretamente exemplos positivos e limitar os falsos positivos, mas ainda longe de resultados satisfatórios para o problema em questão.

Pode-se concluir que o modelo de classificação de prioridade usando SVM tem alguns pontos fortes, mas também apresenta desafios significativos. Embora o modelo tenha alcançado um bom equilíbrio entre precisão e revocação, suas previsões ainda não são suficientemente precisas, pois seu desempenho não é uniforme em todas as classes. A precisão sugere que o modelo está tomando decisões positivas com uma certa confiança, mas o recall indica que ele está perdendo uma proporção significativa de exemplos positivos.

Em comparação com outros modelos estudados, como KNN e Regressão Logística, o modelo SVM se mostrou menos eficaz. A classificação imprecisa levou a valores mais baixos das métricas de avaliação e afetou negativamente o desempenho geral do modelo.

6.1.3 Regressão Logística

A Regressão Logística apresentou resultados melhores que o SVM, com uma acurácia de aproximadamente 89%.

Os resultados apresentados oferecem uma perspectiva favorável acerca do desempenho do sistema de classificação, sugerindo que ele gera previsões com níveis aceitáveis de precisão e confiabilidade. Ao analisar a Figura 6, torna-se evidente que o modelo de Regressão Logística demonstrou uma superioridade considerável em termos de precisão quando comparado ao método SVM. O modelo de Regressão Logística efetuou a categorização das prioridades de maneira notavelmente mais precisa, o que é evidenciado pela distribuição mais equilibrada entre acertos e erros para cada classe na matriz de confusão.

A partir da análise da matriz de confusão, é perceptível que o modelo de Regressão Logística alcançou um número menor de acertos em termos absolutos nas categorias de prioridade Alta e Baixa, porém se destacou consideravelmente na classificação da prioridade Média, superando a performance do SVM.

Ao analisar a matriz de confusão e considerar os valores de Verdadeiros Positivos (VP), Verdadeiros Negativos (VN), Falsos Positivos (FP) e Falsos Negativos (FN), podemos calcular a taxa de acurácia, a qual atingiu 89%. Isso sugere que o modelo está desempenhando bem sua função de classificar as categorias de forma correta e minimizar os erros.

Quanto à precisão do modelo, que mostra o número de previsões positivas corretas, registrou cerca de 84%. Isso aponta que o modelo emite decisões positivas com um grau razoável de confiança e comete erros apenas em um pequeno número de casos. Com um F1-score médio de 78%, o modelo está alcançando um equilíbrio satisfatório entre precisão e recall.

O recall, que foi de 84%, indica que o modelo é capaz de identificar a maioria dos exemplos positivos e minimiza os casos em que deixa passar exemplos que deveria ter identificado. Em síntese, os resultados indicam que o modelo está operando de maneira consistente, com acertos frequentes e erros controlados.

Os resultados obtidos indicam um desempenho sólido do modelo de classificação de prioridade utilizando Regressão Logística. Esses resultados são promissores e o modelo poderia

ser considerado como o classificador principal de toda a arquitetura. No entanto, é notável que o modelo KNN obteve um desempenho ainda mais impressionante, com uma acurácia que ultrapassou os 90%. Esses resultados são apresentados a seguir.

6.1.4 KNN - K-Nearest Neighbors

O KNN apresentou os melhores resultados em relação ao outros dois modelos de classificação (SVM e Regressão Logística), com uma acurácia de aproximadamente 92%.

A matriz de confusão permite visualizar o número de acertos e erros no modelo para cada classe. Nesse caso, os resultados fornecem uma visão bastante positiva do desempenho dos modelo de classificação. O KNN obteve 92% de acurácia isso sugere que o modelo está realizando previsões altamente precisas e é capaz de classificar as prioridades de maneira eficaz.

Com uma precisão de 90% indica que o modelo está tomando decisões positivas com uma confiança considerável e está minimizando os erros obtidos, aliado a isso o um recall médio de 89% significa que o modelo conseguiu identificar corretamente cerca de 89% das instâncias positivas presentes no conjunto de dados. Essas instâncias positivas referem-se a amostras ou instâncias dentro do conjunto de dados que pertencem à classe considerada "positiva"para o problema de classificação, nesse caso são as classes que a requisição realmente pertence, Baixa, Média ou Alta.

O F1-score médio de 87% relatam que o modelo atinge um equilíbrio sólido entre precisão e recall. Este valor reforça um desempenho consistente e bem equilibrado do modelo.

Dessa forma, o modelo de classificação que se mostrou mais eficiente foi o KNN, apesar de ser somente um pouco superior à Regressão Logística. Um fator interessante que pode influenciar isso e alterar o desempenho de um modelo são seus hiperparâmetros. Os modelos de classificação podem ser "ajustados"baseado em hiperparâmetros para melhorar os resultados, porém não foi objeto de estudo do presente trabalho.

O KNN e a Regressão Logística são algoritmos de aprendizado de máquina com propriedades únicas, cada um com suas próprias vantagens, sendo mais apropriados para diferentes tipos de problemas. Abaixo, destacam-se algumas vantagens do KNN em comparação com a regressão logística, que podem contribuir para a obtenção de resultados mais favoráveis, embora não garanta necessariamente resultados superiores:

– O KNN é um método não paramétrico, o que significa que não faz nenhuma suposição específica sobre a distribuição dos dados. Ele depende dos vizinhos mais próximos para

tomar decisões, tornando-o mais flexível com relação aos dados de entrada. A regressão logística, por outro lado, assume uma distribuição específica dos dados, geralmente uma distribuição binomial ou multinomial.

– KNN pode lidar com relacionamentos não lineares entre recursos e saída. Ele pode capturar decisões complexas que não podem ser facilmente modeladas por regressão logística, que é um modelo linear. O KNN não faz suposições sobre a forma funcional dos dados e pode se adaptar a padrões mais complexos.
– KNN pode ser mais robusto para *outliers* em comparação com a regressão logística. Como o KNN é baseado em vizinhos mais próximos, a presença de *outliers* pode ter menos impacto no resultado final. Em contraste, a regressão logística é afetada por pontos de dados individuais e pode ser sensível a valores discrepantes.
– KNN pode lidar melhor com conjuntos de dados desbalanceados em que as classes de destino têm proporções diferentes. Como a classificação é baseada nos vizinhos mais próximos, o KNN pode se adaptar a diferentes proporções de classe. A regressão logística pode ter problemas se houver um desequilíbrio de classe significativo, pois pode tender a favorecer a classe majoritária.

A Tabela 3 mostra o comparativo entre todas os modelos de classificação de prioridades utilizados em nosso experimento, utilizando valores médio de Acurácia, Precisão, Recall e F1-score e os erros.

Tabela 3 – Métricas de avaliação dos modelos

Modelos de classificação	Acurácia	Precisão	F1-score	Recall
SVM	0.73	0.63	0.56	0.53
Regressão Logística	0.89	0.84	0.78	0.84
kNN	**0.92**	**0.90**	**0.87**	**0.89**

Fonte: elaborada pelo autor.

O desempenho desses modelos de classificação podem variar dependendo dos dados e da natureza do problema. Em alguns casos, a regressão logística pode ter um desempenho melhor em termos de precisão (usando métricas de avaliação mais apropriadas), enquanto em outros casos, o KNN pode ser mais adequado para capturar padrões complexos nos dados.

É importante escolher as métricas de avaliação corretas para cada tipo de problema (classificação ou regressão) e entender as características e limitações de cada algoritmo para fazer uma escolha adequada em relação ao desempenho esperado.

De forma geral, a análise dos resultados aponta para o modelo de classificação que

se destacou: o KNN. Este demonstrou a capacidade de gerar previsões precisas e coerentes, equilibrando eficazmente precisão e recall. A baixa diferença entre as previsões e os valores reais, como refletido pelo MSE e MAE, confirma a alta qualidade das previsões geradas pelos modelos.

6.2 Resultados do Experimento B - (CPU)

Após a definição do modelo de classificação de prioridades, o KNN, para auxiliar na gerencia dos recursos, controlando o uso desses recursos de acordo com as prioridades de cada requisição R_i, dá-se prosseguimento na avaliação do consumo de recursos no ambiente de borda.

No Experimento B, esta análise tem como objetivo avaliar o impacto do mecanismo proposto na utilização da CPU em comparação com uma abordagem tradicional. Na abordagem tradicional, o nó de borda mais próximo é selecionado para executar uma tarefa, sem levar em consideração outros fatores que afetam o uso dos recursos limitados da borda.

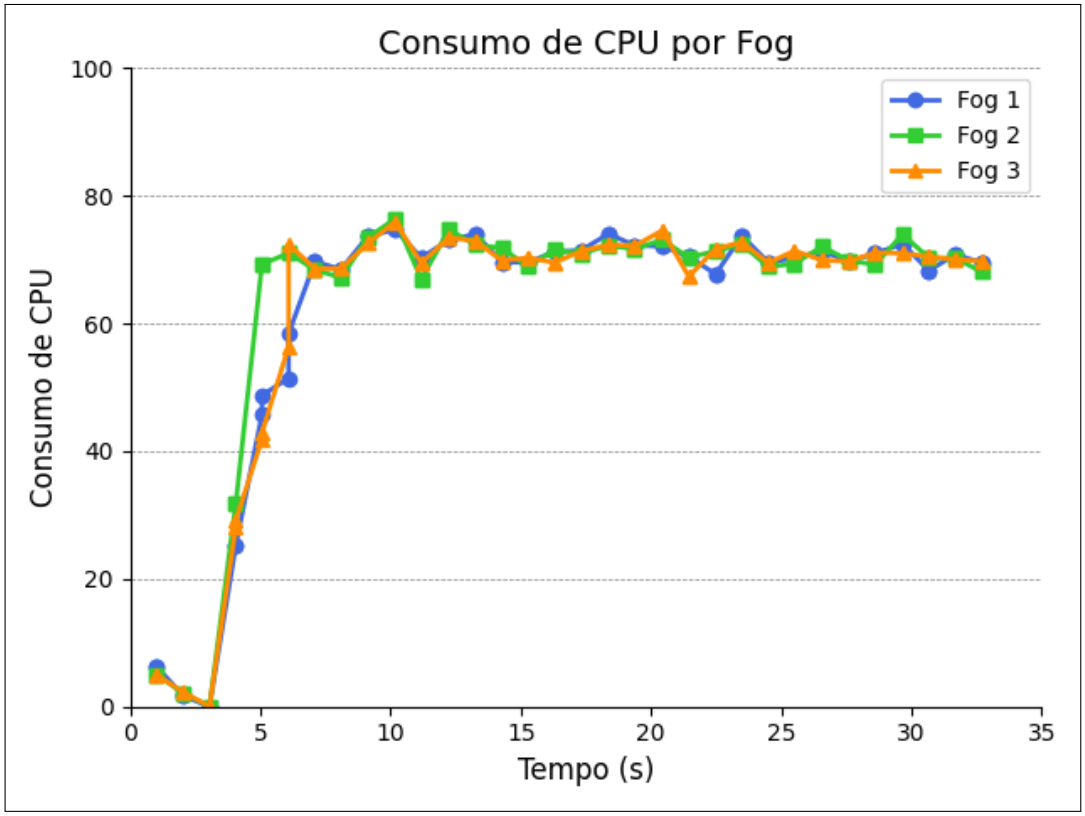
Para realizar essa comparação, foram conduzidos experimentos em dois cenários diferentes. No Cenário I do Experimento B (Figura 7), foram criados 3 nós de borda (*fog*) e um total de 500 requisições foram processadas usando o mecanismo proposto neste trabalho. No Cenário II do Experimento B (Figura 8), a abordagem tradicional foi utilizada, considerando apenas os nós de borda mais próximos para o processamento das tarefas.

Nas Figuras 7 e 8, é apresentado o consumo de CPU durante a execução do experimento com três nós de borda e 500 requisições. Nota-se que, com a abordagem proposta neste trabalho, o uso de CPU se mantém relativamente balanceado, não ultrapassando o valor de 80% de utilização dos recursos dos nós. O limiar estabelecido para esse cenário é de 85%; portanto, se um nó monitorado atinge ou ultrapassa 85% de uso de seus recursos, ele não recebe novas requisições temporariamente.

No gráfico da Figura 8 (sem o mecanismo proposto), as requisições são direcionadas para o nó de borda mais próximo, resultando em um consumo irregular com vários picos de utilização, mantendo uma média de consumo entre 85% e 90% dos recursos.

É importante ressaltar que o limiar de 85% foi estabelecido em todos os cenários, com arquitetura proposta, como uma espécie de barreira para receber novas requisições. No entanto, se um nó estiver com 79% de utilização de recursos, por exemplo, uma nova demanda pode fazer com que ele ultrapasse esse limite estabelecido. Esse limite foi adotado para evitar que os nós fiquem sobrecarregados e possam sofrer falhas. Mesmo assim, pode-se observar picos

Figura 7 – Experimento B - Cenário I - 500 requisições - Alocação com o Mecanismo Proposto



Fonte: elaborada pelo autor.

de utilização de 100% dos recursos na alocação por distância, o que pode levar à sobrecarga e consequentemente à falha do nó.

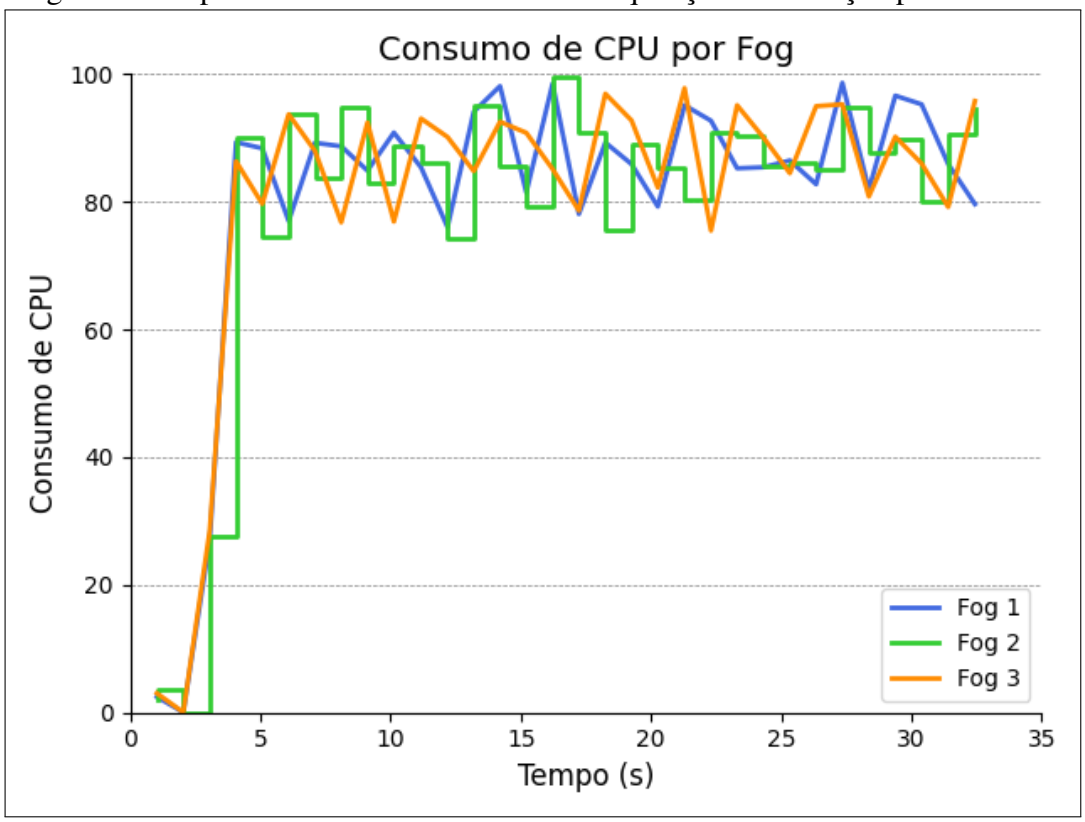
Nos cenários III e IV do Experimento B, o número de requisições foi duplicado (1000 requisições) com o objetivo de avaliar novamente o comportamento do mecanismo proposto. As Figuras 9 e 10 mostram o consumo da CPU para os cenários III e IV, com 3 nós de borda e 1000 requisições.

Novamente, ao analisar os gráficos, é possível observar como o mecanismo reage ao aumento do volume de requisições. Com o mecanismo em funcionamento, o uso de CPU continua a ser gerenciado de forma mais equilibrada, com picos no máximo de 86% a 87% de utilização. Como há uma sobrecarga maior de requisições, é esperado que alguns nós ultrapassem momentaneamente o limite de 85% de utilização do recurso.

Por outro lado, no cenário com alocação por distância de requisições, nota-se um aumento significativo de picos de utilização de CPU conforme o número de requisições aumenta, tanto picos superiores e picos inferiores, significando um uso desbalanceado dos recursos.

Durante esse experimento, o consumo de CPU foi monitorado em todos os cenários. Isso permitiu observar as diferenças na distribuição e utilização dos recursos de processamento

Figura 8 – Experimento B - Cenário II - 500 requisições - Alocação por Distância



Fonte: elaborada pelo autor.

em cada caso. Os resultados obtidos permitem análise sobre a eficácia do mecanismo proposto em relação à abordagem tradicional, especialmente em relação ao uso da CPU nos nós de borda.

6.3 Resultados do Experimento C - (CPU)

No Experimento C realizou-se o aumento do número de nós *fog* para 6 e manteve a variação de 500 e 1000 requisições. Com 500 requisições no Cenário I (Figura 11), o uso do mecanismo proposto resultou em uma utilização máxima de CPU de em torno de 41% da capacidade total e uma média em torno de 39%, demonstrando sua efetividade na distribuição equitativa de tarefas entre todos os nós.

Por outro lado, no experimento sem o mecanismo, no Cenário II com 500 requisições (Figura 12), pode-se observar picos de utilização de CPU próximos a 58% e uma média em torno de 41%.

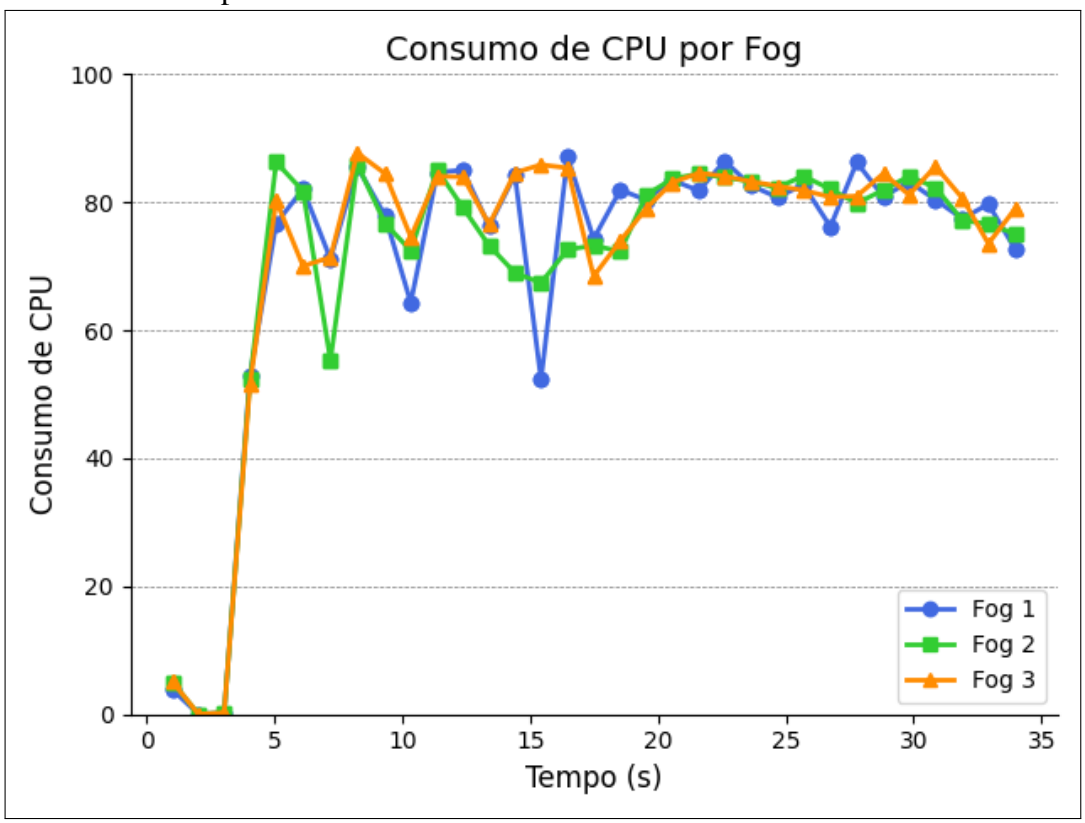
Esses resultados evidenciam que o mecanismo proposto apresenta vantagens interessantes na gestão dos recursos de CPU, mantendo a carga uniformemente distribuída entre os nós de borda. Isso resulta em uma utilização mais eficiente dos recursos disponíveis, reduzindo o risco de sobrecarga e possibilitando um desempenho mais estável do sistema como um todo.

Nos Cenários III e IV (Figuras 13 e 14), aumentou-se novamente o número de requisições para 1000, e pode ser observado, que com o mecanismo proposto (Figura 13), uma elevação mais controlada do uso dos recursos nos momentos de intervalo em t=0s a t=8s, mantendo em média de aproximadamente 42% de utilização da CPU e com máxima de 58% de utilização.

Já na Figura 14 com alocação por distância, observam-se picos maiores de uso da CPU, ultrapassando 60% de utilização. Nota-se também que no intervalo t=5s a t=7s, o consumo sobe exponencialmente rápido nos nós. Em média, o consumo fica em torno de 55%, o que não compromete o comportamento dos nós.

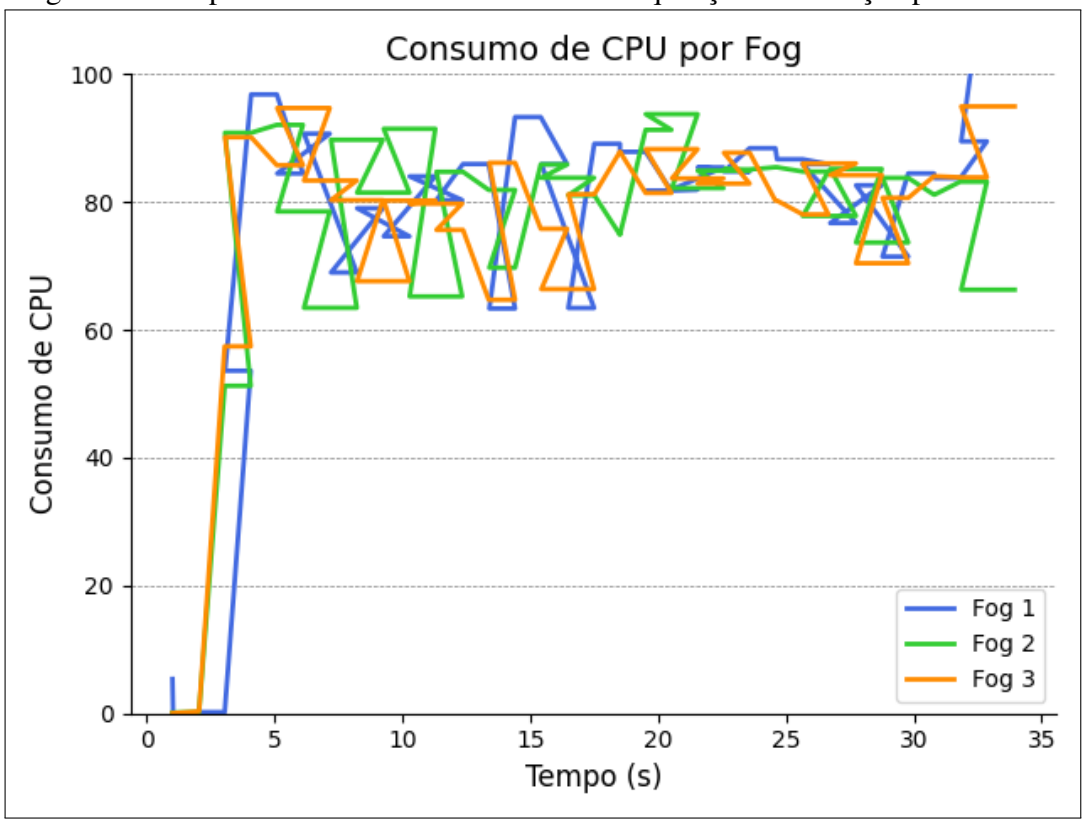
Observa-se que o consumo dos recursos nos Experimentos C não ultrapassa, em nenhum momento, 80% da capacidade dos recursos disponíveis. Esse resultado é atribuído ao aumento do número de nós de borda, pois quanto mais nós são disponíveis, mais recursos estão disponíveis para distribuir a carga. No entanto, é importante observar que o processo de aumentar o número de nós de borda também tem um custo computacional maior, o que leva a um *trade-off* em decisões de projeto. A decisão de aumentar o número de nós deve ser ponderada

Figura 9 – Experimento B - Cenário III - 1000 requisições - Alocação com o Mecanismo Proposto



Fonte: elaborada pelo autor.

Figura 10 – Experimento B - Cenário IV - 1000 requisições - Alocação por Distância



Fonte: elaborada pelo autor.

tendo em conta os recursos disponíveis, os custos associados e a capacidade de gestão de toda a infraestrutura.

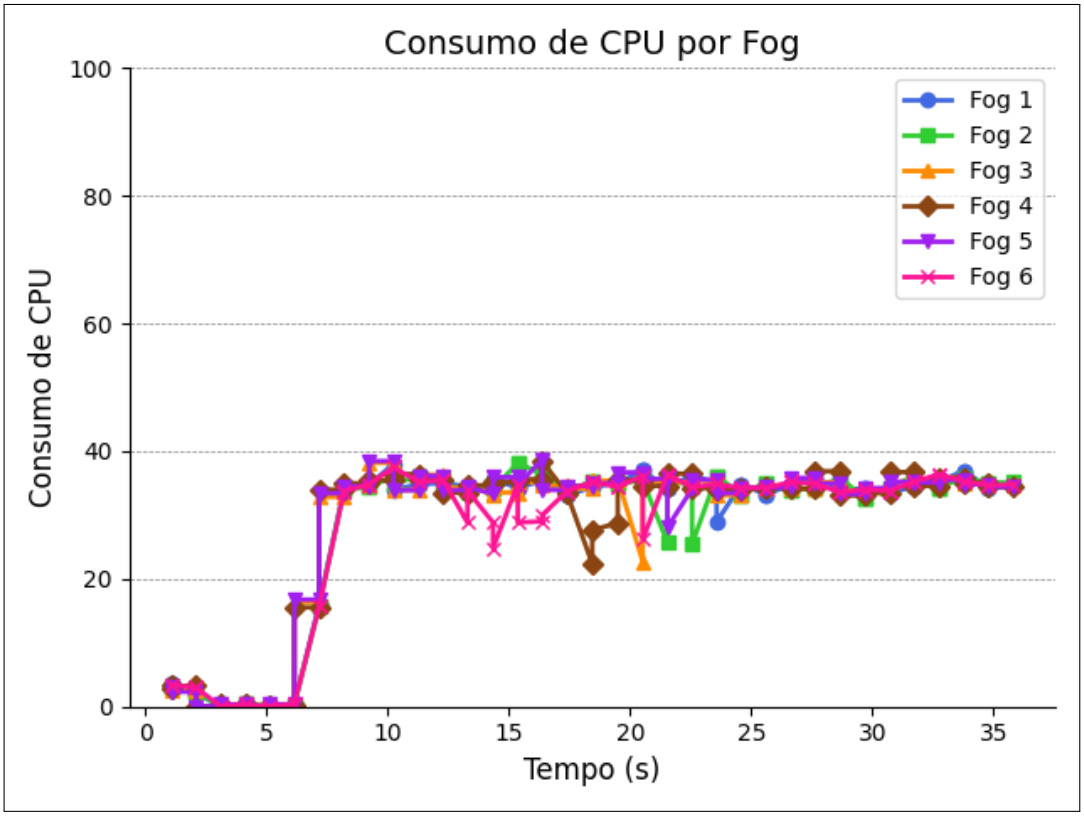
Ao aplicarmos a técnica estatística do teste t de Student, tanto para o Experimento B quanto para o Experimento C, chegamos ao resultado que indica "haver uma diferença significativa entre as médias". Esse achado denota a presença de evidência estatística suficiente para rejeitar a hipótese nula. A hipótese nula (H_0) sustenta que não existe diferença significativa entre as médias dos grupos que estão sendo comparados.

Consequentemente, a conclusão advinda deste resultado é que existe, de fato, uma diferença significativa. Isso nos leva a inferir que as médias dos grupos diferem a tal ponto que essa discrepância não pode ser atribuída meramente ao acaso. Isso, por sua vez, implica que existe um efeito real decorrente da utilização das duas abordagens que estamos comparando.

6.4 Resultados do Experimento D - (Memória)

Neste experimento foi realizada uma análise detalhada do consumo de Memória RAM durante a execução do experimento, considerando diferentes quantidades de requisições. Os dois ambientes distintos foram novamente comparados: um ambiente com a implementação

Figura 11 – Experimento C - Cenário I - 500 requisições - Alocação com o Mecanismo Proposto



Fonte: elaborada pelo autor.

do mecanismo proposto e outro ambiente sem esse mecanismo, onde as requisições foram enviadas para o nó *fog* mais próximo sem considerar o mecanismo de gerenciamento.

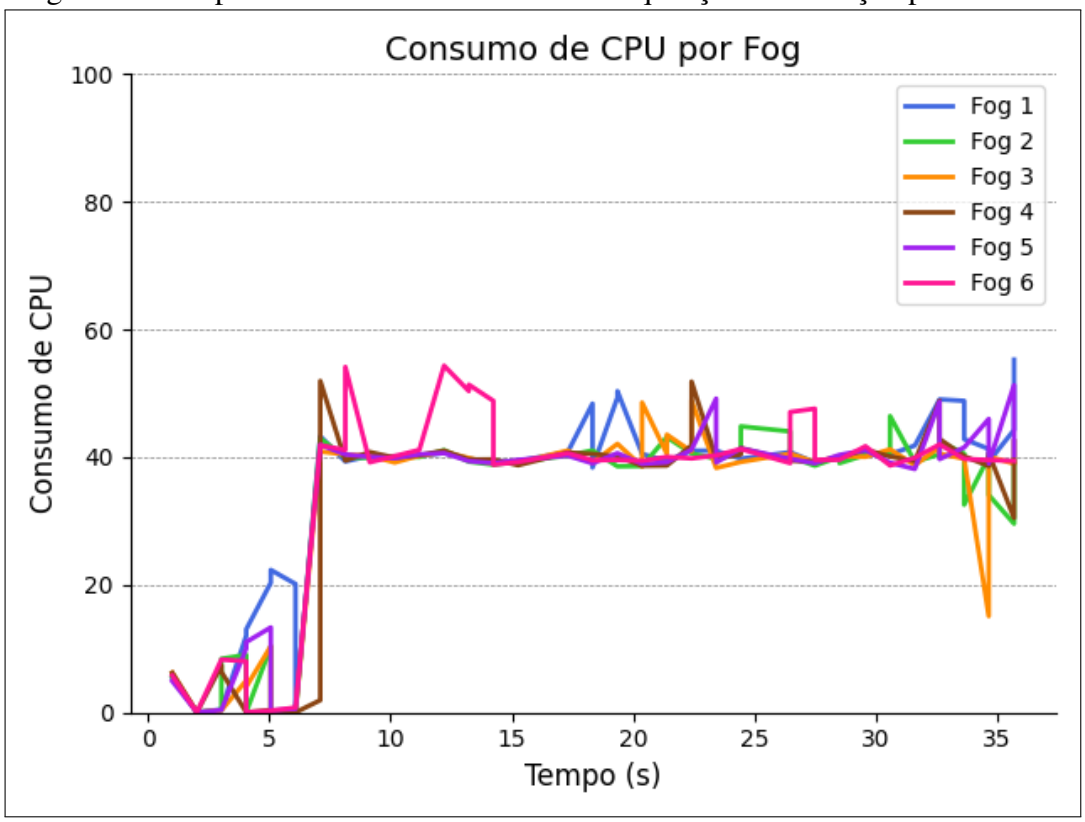
Uma avaliação do consumo de RAM é necessária para entender como o mecanismo proposto afeta o uso do recurso de memória do sistema em comparação com a abordagem tradicional de despacho de solicitações. Essa análise é valiosa para determinar a eficácia do mecanismo no gerenciamento do uso de memória e garantir a alocação adequada de recursos, especialmente em cenários de alta demanda e sob diferentes cargas de requisições.

Com base nos resultados obtidos, é possível entender melhor o comportamento do sistema em relação ao consumo de memória, avaliar a eficácia do mecanismo proposto e tomar decisões informadas para otimizar o desempenho e a estabilidade do ambiente de borda.

Na Figura 15 (Cenário I), ao analisar a utilização do mecanismo proposto neste trabalho, com 3 nós de borda e 500 requisições, é possível observar que o uso da memória se mantém relativamente baixo, não ultrapassando a barreira dos 20% e com uma média de utilização em torno de 16% da capacidade total. Isso demonstra um controle eficiente do uso desse recurso, evidenciando o bom desempenho do mecanismo de alocação.

Por outro lado, na Figura 16, ao analisar o Cenário II, ainda com 3 nós de borda e

Figura 12 – Experimento C - Cenário II - 500 requisições - Alocação por Distância



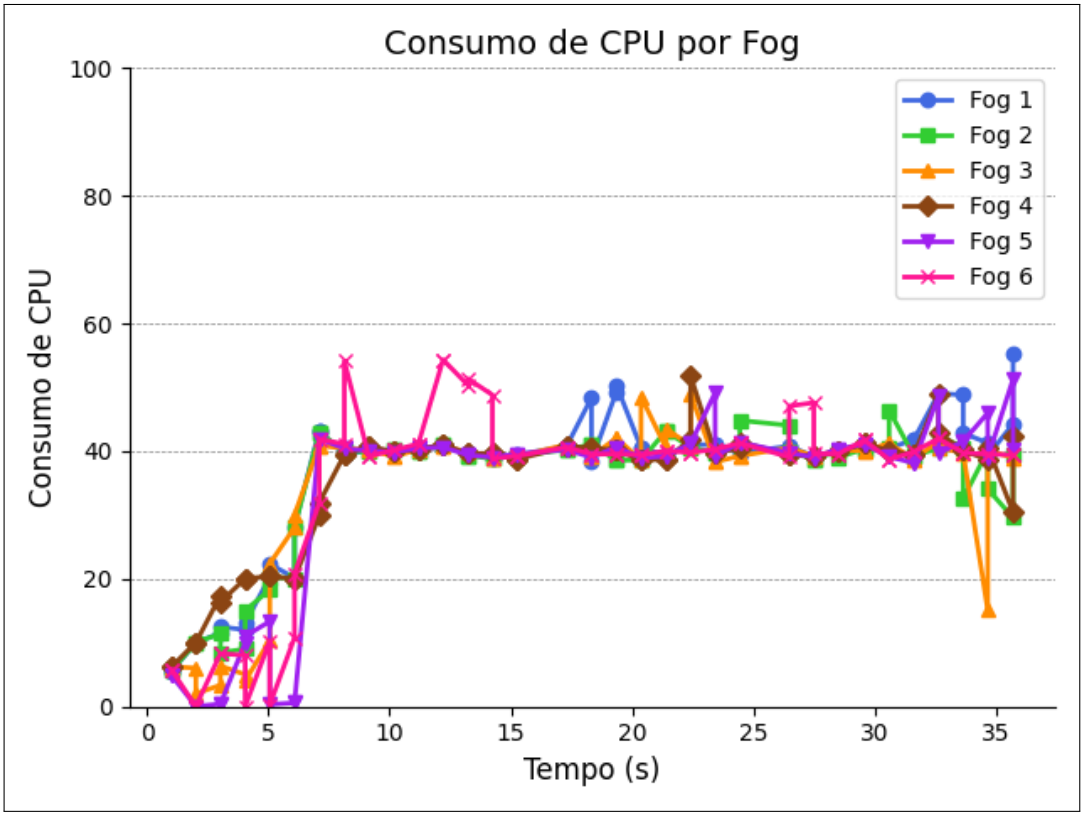
Fonte: elaborada pelo autor.

500 requisições, mas sem a utilização do mecanismo de alocação proposto, pode-se observar que o consumo de Memória RAM tem picos de até 40% no nó de borda 1. Isso ocorre devido ao fato desse nó ser o mais próximo no experimento, resultando em uma alocação maior de requisições para ele. No entanto, devido ao número baixo de requisições, o uso dos recurso de Memória não foi tão afetado. Apesar disso, a arquitetura e o mecanismo de alocação proposto nesta dissertação mostraram-se mais eficientes e controlados na utilização da memória.

Na Figura 17 (Cenário III), novamente o número de requisições é aumentado para 1000, e desta vez o consumo de Memória foi maior. Com o mecanismo proposto, observa-se picos de mais de 60% de utilização de memória e uma média em torno de 38% a 40%. Apesar do aumento, o consumo de memória ainda é menor e mais regulado em comparação com os resultados obtidos sem a utilização do mecanismo, conforme mostrado na Figura 18 (Cenário IV). Neste caso, o uso médio ultrapassa os 40%, e picos chegam a ultrapassar a barreira dos 80%.

Mais uma vez, é importante observar que, assim como ocorre com os recursos da CPU, se um nó for sobrecarregado por memória devido à alocação descontrolada de solicitações, ele pode parar de funcionar e prejudicar o sistema. Assim, os resultados indicam que o mecanismo proposto é eficaz no gerenciamento do consumo de Memória, mantendo-o em níveis seguros e

Figura 13 – Experimento C - Cenário III - 1000 requisições - Alocação com o Mecanismo Proposto



Fonte: elaborada pelo autor.

evitando a sobrecarga de algum nó em particular.

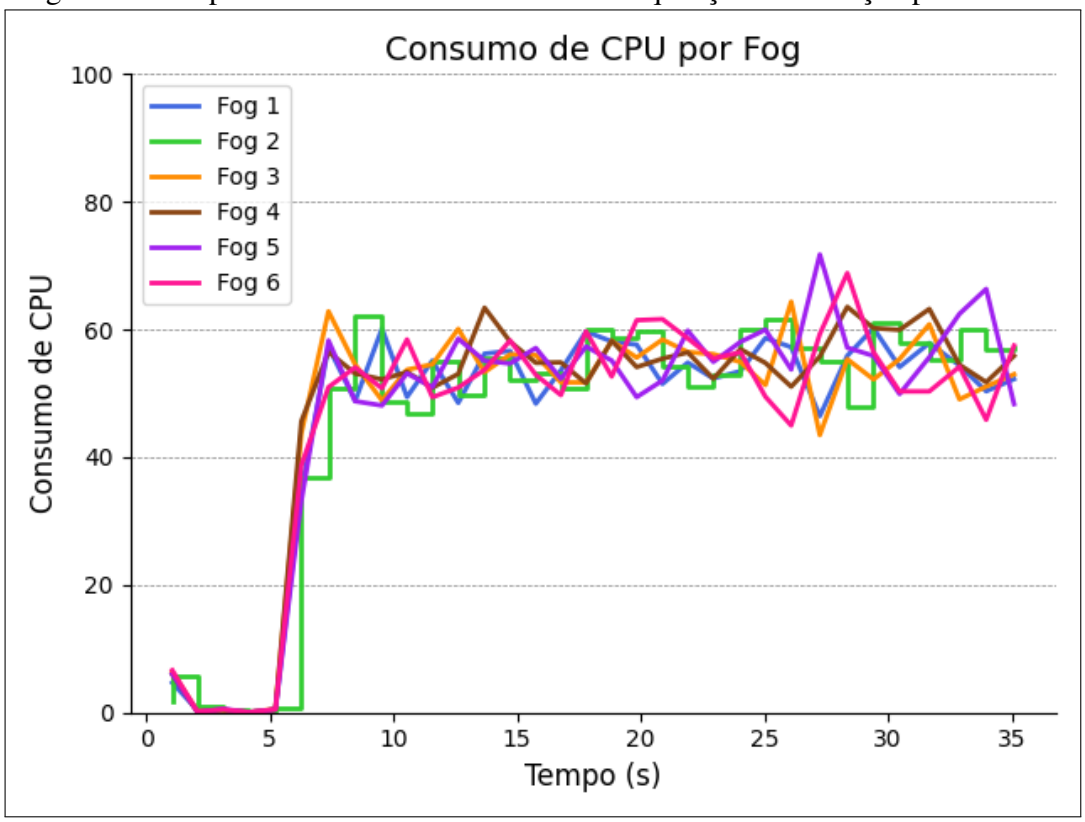
6.5 Resultados do Experimento E - (Memória)

No Experimento E foi realizado um procedimento semelhante ao Experimento D, mas com o dobro do número de de nós de borda (6 nós). No Cenário I (Figura 19), devido ao baixo número de requisições e um número relativamente elevado de nós de borda, observa-se um consumo reduzido deste recurso, com picos de utilização de no máximo 30%. Por outro lado, no Cenário II (Figura 20), sem o mecanismo proposto, o consumo foi um pouco maior, alcançando picos de no máximo 42%.

Neste cenário, com um baixo número de requisições, o sistema demonstrou um funcionamento equilibrado. . Observou-se uma média de redução de cerca de 8% no consumo de memória, em comparação com a ausência da arquitetura proposta. Embora possa parecer uma melhoria modesta, em ambientes com recursos restritos, qualquer ganho já representa um indicador positivo

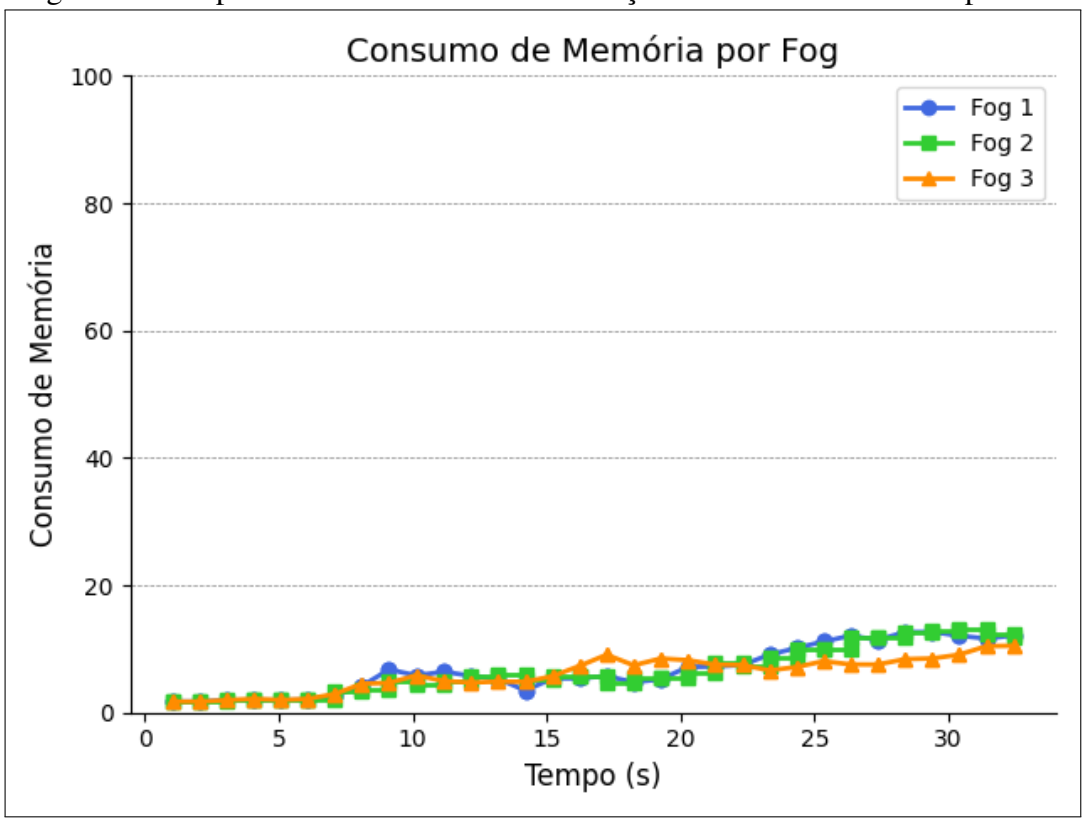
Nas etapas III e IV do Experimento E, quando o número de solicitações aumentou

Figura 14 – Experimento C - Cenário IV - 1000 requisições - Alocação por Distância



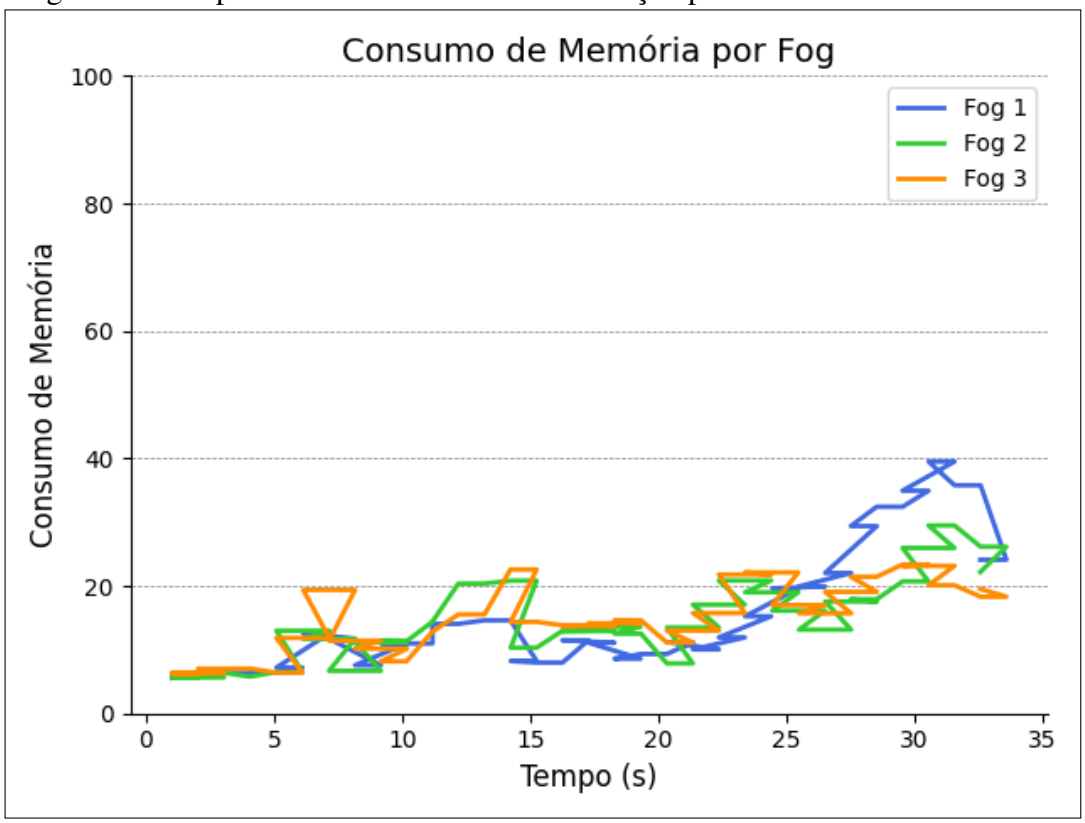
Fonte: elaborada pelo autor.

Figura 15 – Experimento D - Cenário I - Alocação com o Mecanismo Proposto



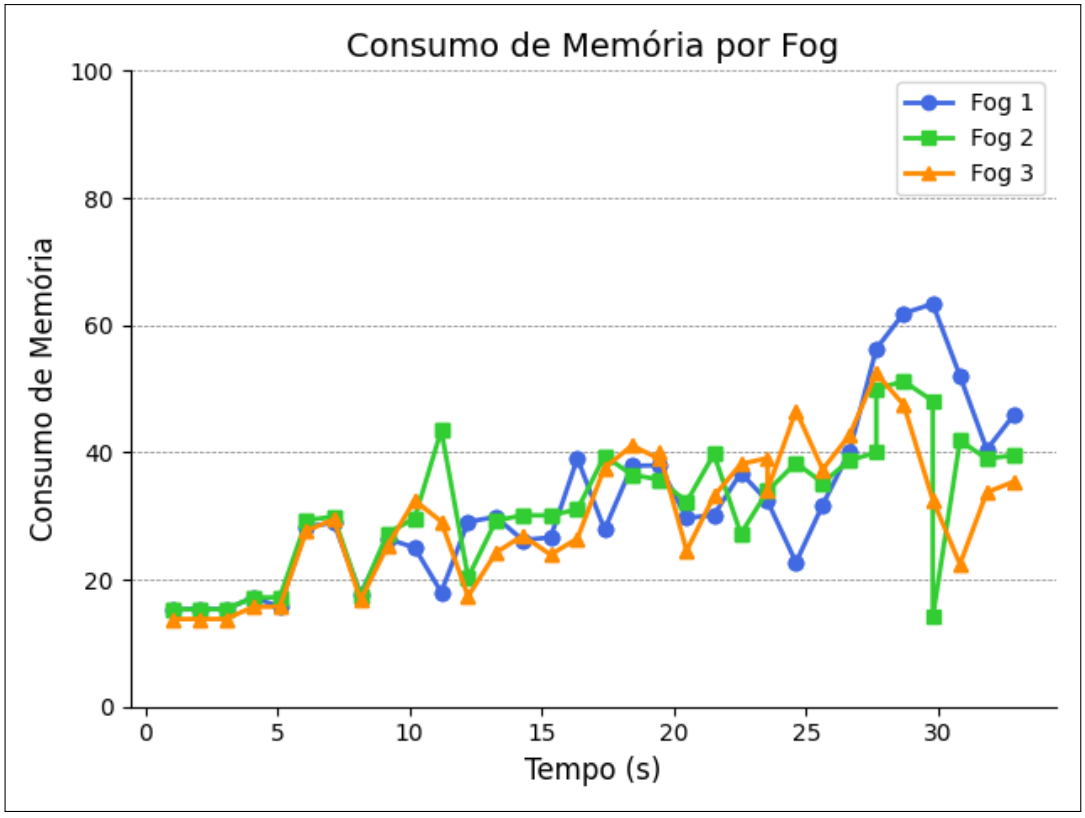
Fonte: elaborada pelo autor.

Figura 16 – Experimento D - Cenário II - Alocação por Distância



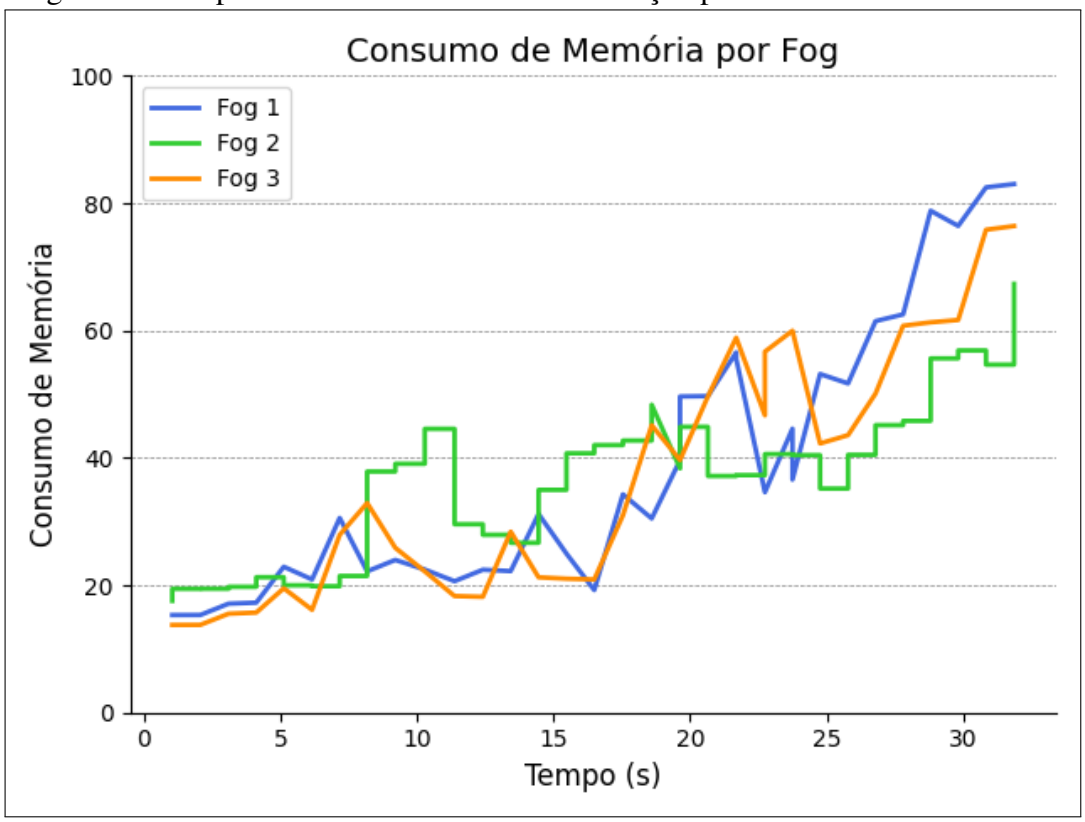
Fonte: elaborada pelo autor.

Figura 17 – Experimento D - Cenário III - Alocação com o Mecanismo Proposto



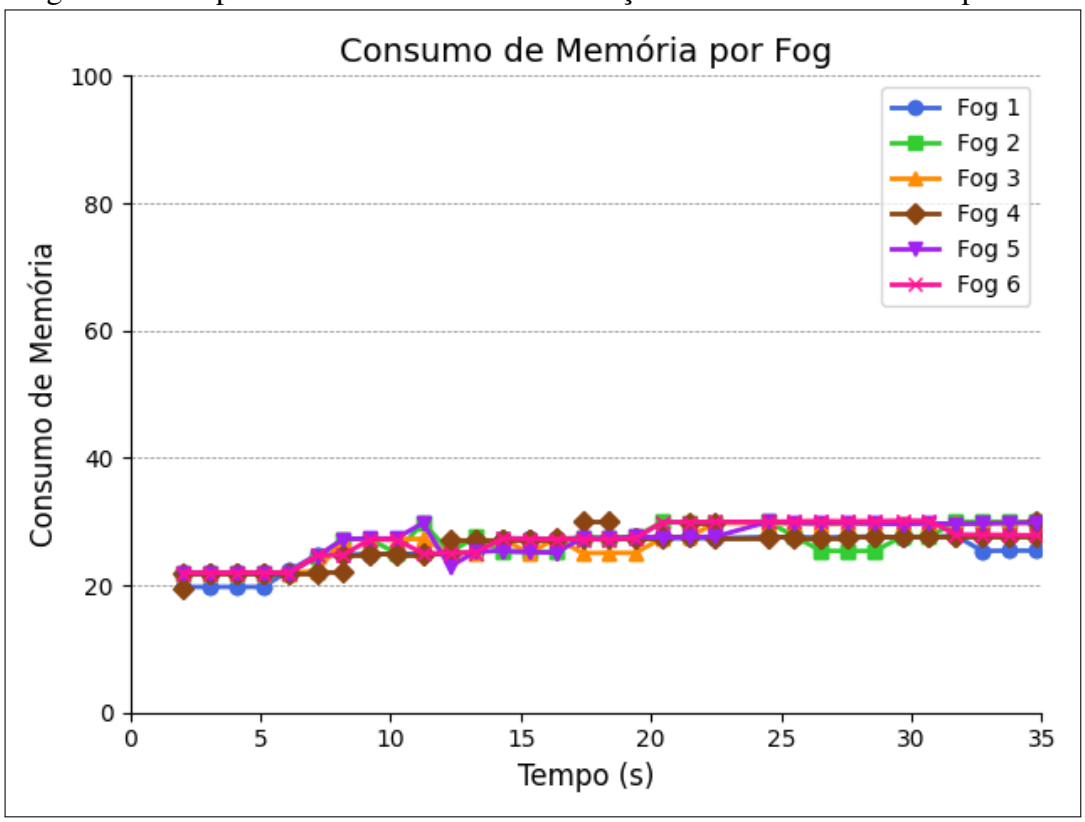
Fonte: elaborada pelo autor.

Figura 18 – Experimento D - Cenário VI - Alocação por Distância



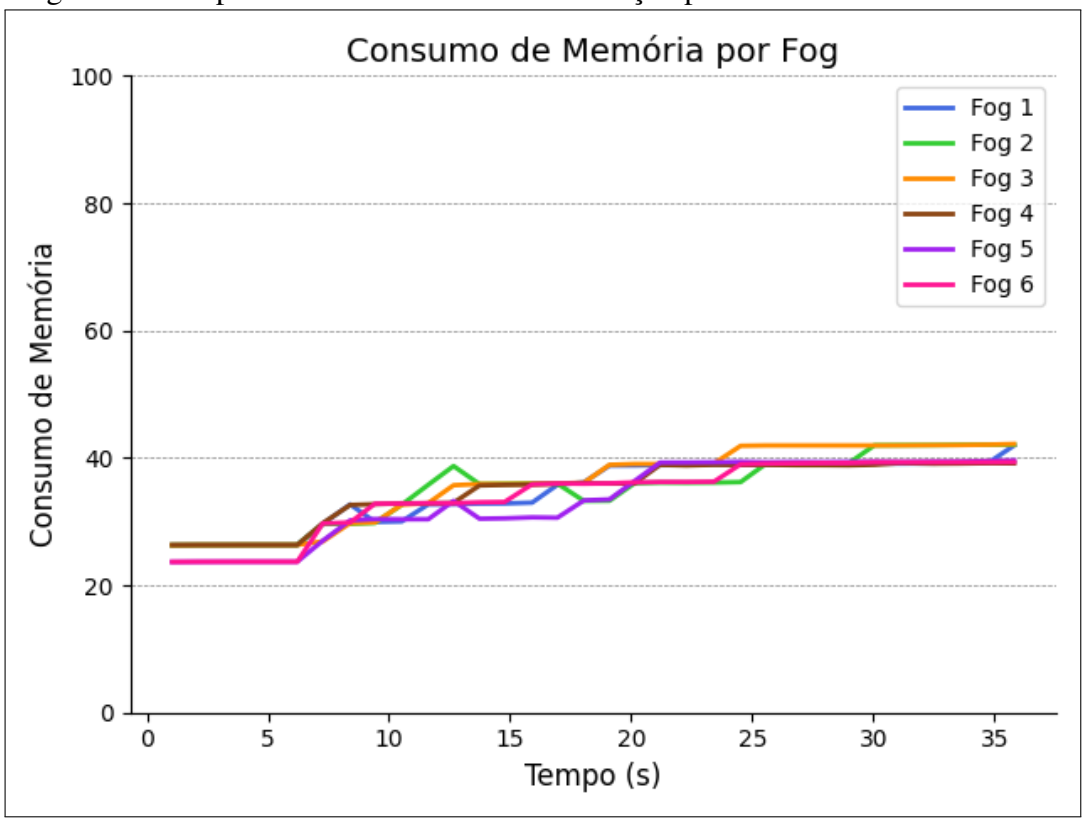
Fonte: elaborada pelo autor.

Figura 19 – Experimento E - Cenário I - Alocação com o Mecanismo Proposto



Fonte: elaborada pelo autor.

Figura 20 – Experimento E - Cenário II - Alocação por Distância



Fonte: elaborada pelo autor.

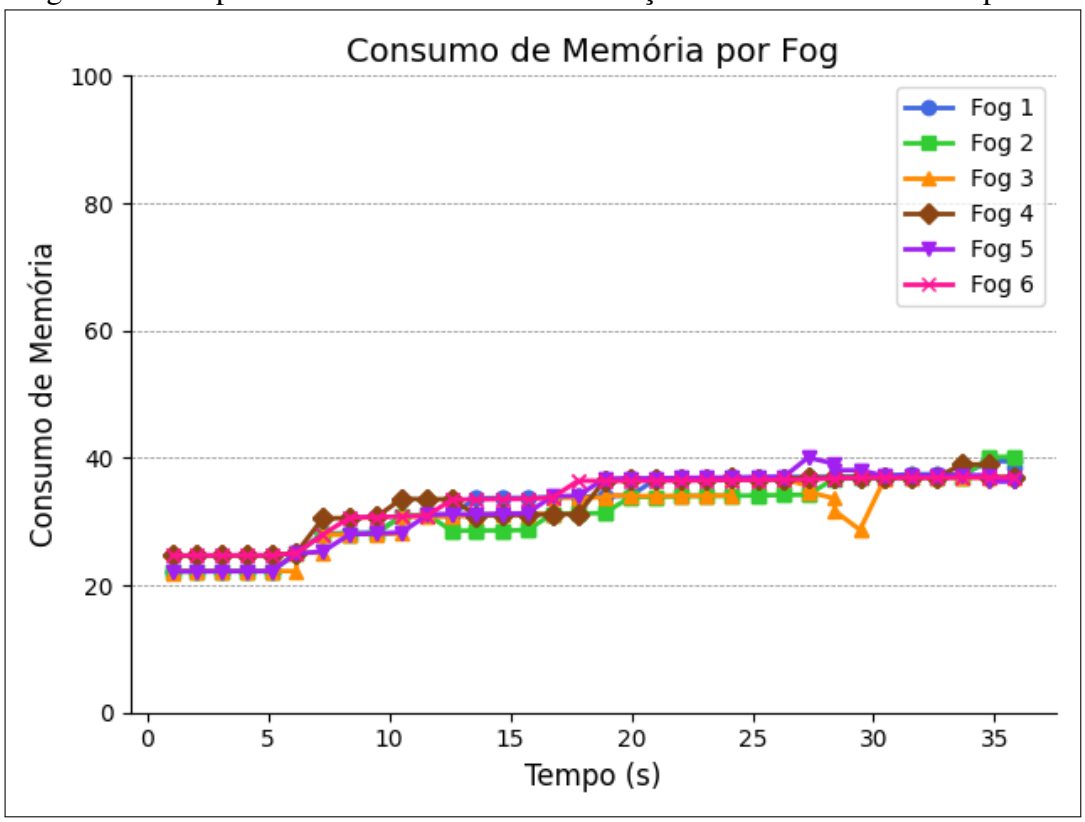
para 1000, foi possível observar um padrão semelhante ao observado nas etapas I e II. O uso do mecanismo proposto resultou em um ligeiro aumento no consumo de memória RAM (conforme ilustrado na Figura 21), com picos de uso atingindo no máximo 41%. Enquanto isso, no Cenário IV, em que o mecanismo proposto não foi utilizado (conforme Figura 22), o consumo de memória também aumentou de forma discreta, ultrapassando ligeiramente os 60%. Em média, o consumo de memória com o mecanismo mínimo oscilou entre 20% e 40%, enquanto a alocação baseada em distância resultou em um consumo variando entre 30% e 60%.

Apesar do acréscimo no consumo de memória à medida que o número de requisições aumenta, é notável que o mecanismo proposto mantém seu papel de controlar o uso desse recurso de maneira eficaz e balanceada. Em ambos os cenários, o consumo de memória não atinge um patamar crítico que possa comprometer o funcionamento estável do sistema.

Assim, nos cenários III e IV, fica evidente que o mecanismo proposto continua a oferecer vantagens em termos de gerenciamento e otimização do consumo de memória, garantindo o bom funcionamento e desempenho do sistema mesmo em situações com requisitos mais exigentes.

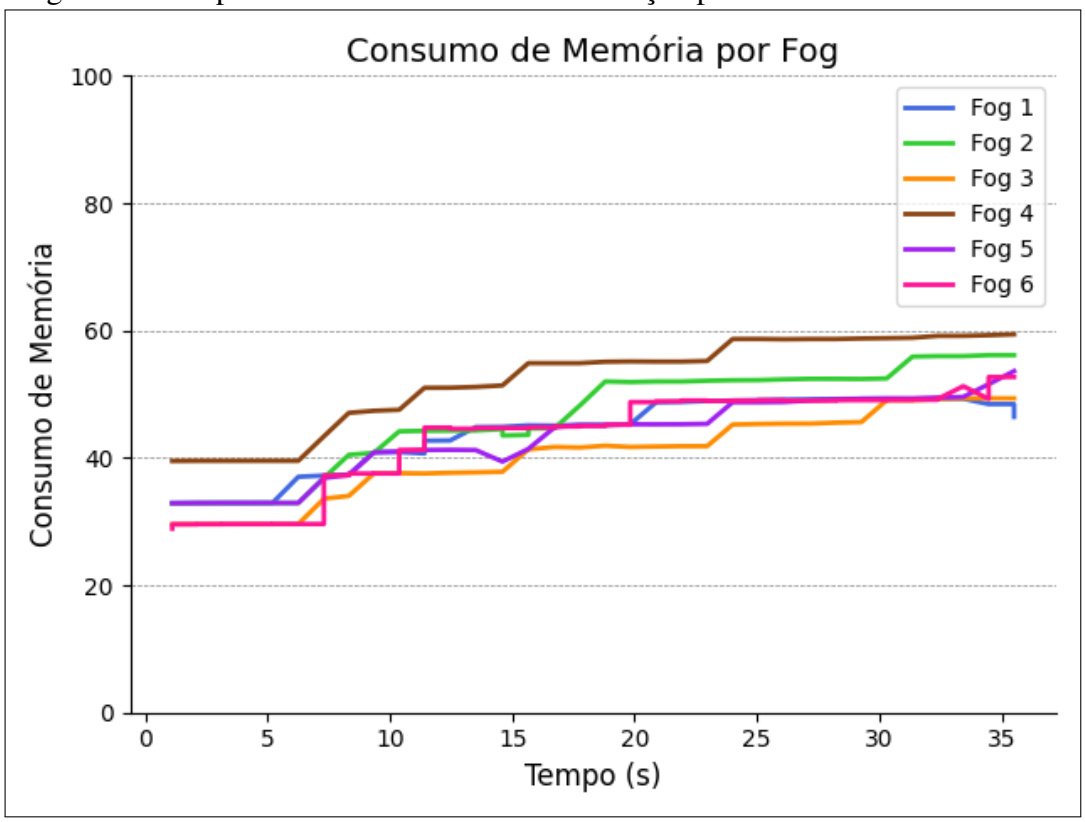
É importante observar que a vantagem do mecanismo proposto torna-se mais aparente

Figura 21 – Experimento E - Cenário III - Alocação com o Mecanismo Proposto



Fonte: elaborada pelo autor.

Figura 22 – Experimento E - Cenário IV - Alocação por Distância



Fonte: elaborada pelo autor.

à medida que a carga de requisições aumenta e a quantidade de nós de borda é menor. Em cenários de maior demanda, o mecanismo de alocação de recursos proposto neste trabalho pode se tornar essencial para garantir uma distribuição mais equilibrada de recursos e evitar a sobrecarga de nós específicos.

Esse o uso controlado da Memória é essencial para garantir o funcionamento estável e confiável de um sistema de computação em borda, pois recursos sobrecarregados podem levar a falhas e indisponibilidade de serviços. A abordagem proposta neste trabalho provou ser benéfica no gerenciamento do consumo de memória e contribui para um desempenho de infraestrutura mais equilibrado neste tipo de ambientes.

Do modo análogo à análise do consumo de CPU, aplicamos a metodologia estatística do teste t de Student aos Experimentos D e E, obtendo conclusões semelhantes. Em ambos os casos, constatamos a possibilidade de rejeitar a hipótese nula.

Assim sendo, as constatações convergem para a existência de evidências estatísticas que sustentam a presença de uma diferença substancial. Isso se aplica à escolha de adotar ou não a abordagem apresentada neste estudo.

6.6 Resultados do Experimento F - Mecanismo de Falha

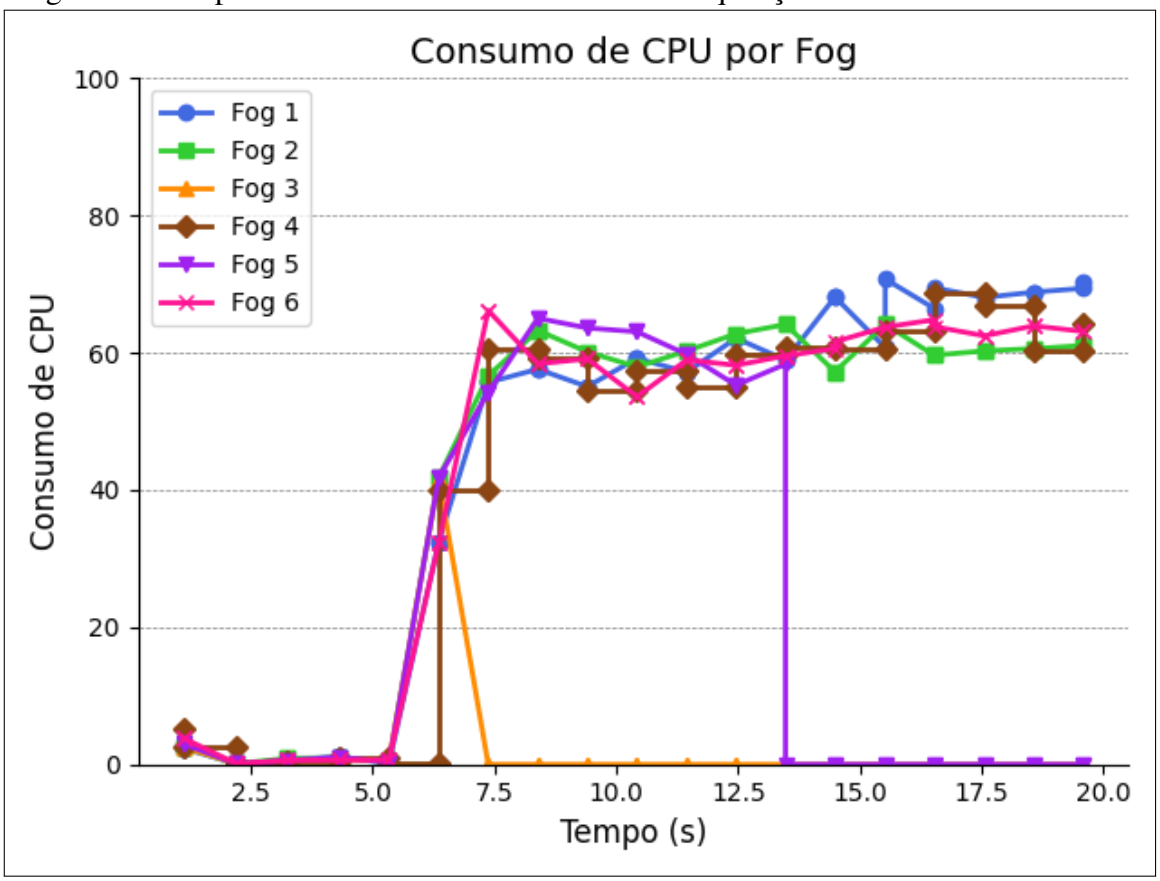
No Experimento F (Figura 23) utilizou-se o cenário III do Experimento E, com 1000 requisições e 6 nós de borda. Neste experimento, foram selecionados aleatoriamente dois nós de névoa para forçar uma falha (nós 3 e 5).

Primeiro, o Nó 3 foi escolhido para ser desligado abruptamente, simulando um desligamento por escassez de recurso ou uma falha do dispositivo. Observa-se que após o desligamento (em t = 7,0s), os outros nós passaram a consumir mais recursos, podendo ser observado no Nó 4. Isso ocorre porque todas as requisições que estavam alocadas nos nós de borda 3 ficam em um estado de salvamento, aguardando a resolução da solicitação. Uma *Thread* funciona como um *checkpoint*, pois nesse caso, como um nó caiu (Nó 3), todas as requisições em processamento foram perdidas. No entanto, devido a esse mecanismo de *checkpoint*, essas requisições são realocadas nos outros nós disponíveis que possuem capacidade para processá-las. Esse processo ocorre devido ao monitoramento contínuo dos recursos utilizados.

Após a realocação das requisições, o consumo dos nós que as receberam tende a aumentar temporariamente até que tudo se estabilize novamente. Esse processo representa um balanceamento do uso dos nós previamente conectados ao nó desligado. Essa capacidade de

redistribuição das requisições confirma a capacidade do sistema de se recuperar de falhas, uma vez que foi identificada a falha na rede e os outros nós absorveram as requisições perdidas. Esse processo é repetido mais uma vez em t = 14,0s, com o desligamento do Nó 5.

Figura 23 – Experimento F - 6 nós de borda e 1000 requisições



Fonte: elaborada pelo autor.

É importante ressaltar que os nós desligados foram escolhidos aleatoriamente, uma vez que não é possível prever qual nó irá falhar. Isso demonstra a capacidade do sistema de lidar com falhas de forma adaptativa, realocando as tarefas nos nós disponíveis para garantir a continuidade do serviço.

6.6.1 Discussões Finais

Comparando os resultados obtidos nos dois ambientes (com mecanismo proposto e com o aleatório), é possível determinar se a alocação com o mecanismo proposto é eficaz na melhoria da utilização da CPU e Memória e, posteriormente, no desempenho geral do sistema. Esta análise é essencial para verificar e avaliar a eficácia do mecanismo de alocação de recursos proposto neste trabalho e fornece informações importantes sobre sua utilidade em ambientes

de computação de borda, desde a definição de prioridade até a execução de requisições. A compreensão desses resultados pode contribui para a tomada de decisões informadas sobre o gerenciamento de recursos em sistemas de borda de modo a alcançar um melhor equilíbrio entre desempenho, custo computacional e consumo dos recursos.

Comparando a arquitetura apresentada neste trabalho com outros métodos de definição de prioridade, abordados no trabalho (BUI *et al.*, 2021), observa-se que os autores utilizam critérios diferentes para definir prioridades. Eles empregam uma estratégia mais complexa, levando em consideração seis fatores para classificar as requisições de recursos, o que requer uma análise mais elaborada das solicitações e, consequentemente, pode aumentar o tempo total de processamento. Os resultados desse trabalho demonstram que os autores conseguiram definir suas prioridades das solicitações, no entanto, notou-se um consumo excessivo dos recursos e os mesmos destacam que um ponto fraco na abordagem foi identificado na distribuição de prioridade. Embora a utilização dos recursos seja menor em comparação com outros algoritmos de definição de prioridade utilizados por eles, a utilização chega próximo de 100%, sendo reduzida apenas após um determinado período.

Na arquitetura proposta nesta dissertação, além de levar diversos fatores em consideração para a definição de prioridade, como em (BUI *et al.*, 2021), também houve preocupação em controlar o uso dos recursos nos nós de borda. Seria ineficiente ter uma boa classificação de prioridades para requisições sem um controle adequado do uso dos recursos para a resolução dessas solicitações, pois isso poderia levar à indisponibilidade dos nós. A utilização de recursos é um aspecto importante para o gerenciamento eficiente dos recursos. Nos resultados obtidos neste trabalho, observou-se que em nenhum caso testado houve uma sobrecarga excessiva de utilização dos recursos das bordas, mesmo quando há menos nós disponíveis e mais requisições.

Outro fator importante a ser comparado é o fato de que os autores em (BUI *et al.*, 2021) estabelecem a utilização de um fator de preempção no algoritmo para tratar situações de emergência. Essa pode ser uma estratégia válida, mas se não for gerenciada adequadamente, pode causar gargalos na rede. Isso ocorre porque os recursos já foram alocados e a interrupção dessa tarefa resultará na perda desses recursos. Se uma tarefa está em execução, é porque foi classificada como prioritária em relação às outras. Portanto, é essencial considerar cuidadosamente o impacto da preempção nas operações da rede.

Um dos principais critérios que diferencia o trabalho proposto dos demais é o seu mecanismo de controle de falha, o qual desempenha um papel crucial devido à constante

exposição dos nós de borda a possíveis falhas, tais como erros de conexão, operacionalidade instável e indisponibilidade. Essa característica é de significativa importância, visto que os nós de borda têm um papel fundamental no processamento e encaminhamento de dados e requisições. Esse mecanismo é uma forma controlar e de gerenciar os recursos na borda.

7 CONCLUSÕES E TRABALHOS FUTUROS

A gestão de recursos é uma técnica fundamental em sistemas que envolvem computação em borda, devido às limitações dos recursos disponíveis. Nesta dissertação é apresentado um mecanismo para alocar eficientemente os recursos em ambientes de computação em borda, com o objetivo principal de garantir uma alocação de modo a priorizar as tarefas de acordo com sua importância, os recursos necessários e a disponibilidade em tempo real nos nós de borda.

Para atingir esse objetivo, foi desenvolvido um classificador de prioridades para as requisições, baseado em técnicas de aprendizado de máquina. Esse classificador é treinado com base em dados históricos e características relevantes das tarefas, como sua criticidade, latência, tipo de conexão, localização, tempo decorrido e carga de recursos requeridos. Utilizando esses dados, o modelo de classificação de prioridades, em conjunto com o monitoramento dos recursos em tempo de execução, fornece uma distribuição de prioridades para as requisições dos dispositivos IoT. Com isso, é possível alocar os recursos de forma eficiente e reduzir o consumo desnecessário.

O classificador mais eficiente em nossas avaliações foi o KNN, com uma acurácia de 92%, superando os modelos de classificação SVM e Regressão Logística. Obtivemos uma precisão de 0.90, F1-score de 0.97 e recall de 0.89. Dessa forma, o KNN foi escolhido como o classificador no experimento de alocação de requisições e controle de recursos.

No mecanismo de alocação de recursos foram otimizados a distribuição dos recursos disponíveis nos nós de borda, considerando a prioridade atribuída às tarefas e a quantidade e disponibilidade dos recursos necessários. O uso desse mecanismo resulta em uma alocação eficiente dos recursos, reduzindo seu consumo. Por exemplo, observa-se que o uso da CPU se tornou mais estável, variando em média entre 75% a 80% para 3 nós de borda e em 40% para 6 nós de borda, quando comparado à abordagem com alocação aleatória, evitando picos de consumo que poderiam levar à indisponibilidade do serviço.

Em relação à Memória, também foram observadas melhorias significativas, especialmente em cenários com mais requisições e menos nós de borda. O mecanismo demonstrou eficiência ao reduzir o consumo de Memória, mantendo-o estável, com média abaixo do consumo da abordagem sem o mecanismo proposto. Esses resultados indicam que o mecanismo proposto é capaz de lidar de forma adequada com uma carga maior de requisições, garantindo uma alocação mais eficiente dos recursos de memória e evitando sobrecargas em nós específicos.

Além disso, o mecanismo de controle de falha proposto mostrou-se eficiente em

detectar erros de nós de borda e realocar as tarefas para garantir o atendimento de todas as requisições. Dessa forma, assegura-se que o serviço continue funcionando mesmo em situações de falha.

No contexto da Computação em borda, em que a latência e o tempo de resposta são críticos, a alocação de recursos com base na prioridade das tarefas é fundamental para atender às necessidades das aplicações e serviços. A abordagem proposta neste trabalho, combinando um classificador de prioridades e um mecanismo de alocação de recursos inteligente, oferece uma solução simples e eficaz para otimizar a alocação de recursos em ambientes de Computação em borda.

7.1 Trabalhos Futuros

Como trabalhos futuros, há diversas possibilidades de aprimoramento da proposta apresentada. Algumas delas incluem:

1. Avaliação de escalabilidade: É importante realizar experimentos e avaliar o desempenho da proposta em termos de escalabilidade. Isso envolve testar o sistema com um maior volume de requisições e verificar como ele se comporta em termos de tempo de resposta, utilização de recursos e capacidade de lidar com cargas de trabalho mais intensas. Essa avaliação permitirá identificar possíveis gargalos e otimizar a arquitetura para garantir um desempenho consistente mesmo em cenários de alta demanda.
2. Utilização de dados de aplicações reais: Para validar ainda mais a proposta, é interessante coletar dados de aplicações reais e testar o sistema com esses dados. Além disso, pode-se utilizar uma base de dados voltado no cenário de *Mobile Edge Computing*. Isso permitirá avaliar como a alocação de recursos com base na prioridade das tarefas se comporta em situações reais e verificar se os resultados obtidos são consistentes com as expectativas.
3. Ajuste de hiperparâmetros dos classificadores: Para melhorar a capacidade de generalização dos classificadores de prioridades, é recomendado ajustar seus hiperparâmetros. Isso ajudará o modelo a evitar problemas como o superajuste (*overfitting*) ou o subajuste (*underfitting*), permitindo que ele aprenda padrões relevantes nos dados e seja capaz de generalizar para diferentes situações reais. Utilizar também *crossvalidation* (validação cruzada) para validar os modelos escolhidos.
4. Maximização da eficiência dos pesos de prioridade: Aperfeiçoar a estratégia de adição de pesos nas prioridades é uma área de interesse. Isso envolve ajustar os pesos de forma a

torná-los mais assertivos, levando em consideração diferentes quantidades de recursos e tipos adicionais de serviços. Essa maximização da eficiência dos pesos ajudará a melhorar a alocação de recursos e a priorização das tarefas, garantindo um desempenho mais otimizado do sistema.

5. Tornar a escolha do nó independente da posição do nó, visando mobilidade, uma vez que lidar com a posição geográfica dos nós pode ser uma tarefa complexa. Como alternativa, pode ser introduzido o Received Signal Strength Indication (Indicador de Intensidade do Sinal Recebido) (RSSI), que é um tipo de métrica que analisa a qualidade e potência do sinal de conexão recebido por um determinado aparelho. Podemos utilizar esse método para determinar as requisições dos dispositivos finais mais próximos.
6. Realizar ajustes no mecanismo de tratamento de falhas com o objetivo de prevenir o consumo excessivo de memória ao armazenar os estados dos nós e as requisições, como uma maneira de contornar possíveis falhas. Uma abordagem viável pode ser a utilização de algoritmos de otimização e a alocação de recursos subutilizados ou mesmo utilização da Nuvem.
7. Para enriquecer a análise de priorização de tarefas, realizaremos a aplicação de outras técnicas, tais como Escalonamento por Lote, Escalonamento *Round Robin* e Escalonamento por Antecipação (Preemptivo), e, em seguida, realizar uma comparação com a abordagem atualmente utilizada.

REFERÊNCIAS

ARENA, F.; PAU, G. When edge computing meets iot systems: Analysis of case studies. **China Communications**, [*S.l.*], v. 17, n. 10, p. 50–63, 2020. Disponível em: https://ieeexplore.ieee.org/abstract/document/9248516. Acesso em: 03 abr. 2023.

AWAD, M.; KHANNA, R. **Efficient Learning Machines**: Theories, concepts, and applications for engineers and system designers. [*S. l.*]: Apress Berkeley, CA, 2015.

BACHIEGA, J.; COSTA, B.; CARVALHO, L. R.; ROSA, M. J. F.; ARAUJO, A. Computational resource allocation in fog computing: A comprehensive survey. **ACM Comput. Surv.**, New York, NY, USA, v. 55, n. 14s, jul. 2023. Disponível em: https://doi.org/10.1145/3586181. Acesso em: 21 abr. 2023.

BAYıLMış, C.; EBLEME, M. A.; ÇAVUşOğLU Ünal; KüçüK, K.; SEVIN, A. A survey on communication protocols and performance evaluations for internet of things. **Digital Communications and Networks**, [*S.l.*], v. 8, n. 6, p. 1094–1104, 2022. Disponível em: https://www.sciencedirect.com/science/article/pii/S2352864822000347. Acesso em: 21 mar. 2023.

BELLAVISTA, P.; BERROCAL, J.; CORRADI, A.; DAS, S.; FOSCHINI, L.; ZANNI, A. A survey on fog computing for the internet of things. **Pervasive and Mobile Computing**, [*S.l.*], v. 52, p. 71–99, 2019. Disponível em: https://www.sciencedirect.com/science/article/abs/pii/S1574119218301111. Acesso em: 05 abr. 2023.

BENCHIKH, L.; LOUAIL, L. Task scheduling approaches for fog computing. *In*: 2021 30TH WIRELESS AND OPTICAL COMMUNICATIONS CONFERENCE (WOCC), 2021, Taipei. **IEEE**. [*S.l.*]: IEEE, 2021. p. 38–42. Disponível em: https://ieeexplore.ieee.org/document/9603112. Acesso em: 13 mai. 2023.

BHUSHAN, S.; MAT, M. Priority-queue based dynamic scaling for efficient resource allocation in fog computing. *In*: 2021 IEEE INTERNATIONAL CONFERENCE ON SERVICE OPERATIONS AND LOGISTICS, AND INFORMATICS (SOLI), 2021, Singapore. **IEEE**. [*S.l.*]: IEEE, 2021. p. 1–6. Disponível em: https://ieeexplore.ieee.org/document/9672442. Acesso em: 15 mai. 2023.

BONOMI, F.; MILITO, R.; ZHU, J.; ADDEPALLI, S. Fog computing and its role in the internet of things. *In*: PROCEEDINGS OF THE FIRST EDITION OF THE MCC WORKSHOP ON MOBILE CLOUD COMPUTING, 2012, Helsinki. **Association for Computing Machinery**. New York: Association for Computing Machinery, 2012. p. 13–16. Disponível em: https://doi.org/10.1145/2342509.2342513. Acesso em: 02 mai. 2023.

BUI, T. B.; SAKR, A.; CASTRILLóN, J.; SCHUSTER, R. Six-factors score-based match-making based on priority and preemption for resource allocation in edge computing. *In*: 2021 IEEE INTERNATIONAL CONFERENCE ON EDGE COMPUTING (EDGE), 2021, Chicago. **IEEE**. [*S.l.*]: IEEE, 2021. p. 44–50. Disponível em: https://ieeexplore.ieee.org/document/9712013. Acesso em: 21 mai. 2023.

CHENG, Z.; LI, P.; WANG, J.; GUO, S. Just-in-time code offloading for wearable computing. **IEEE Transactions on Emerging Topics in Computing**, [*S.l.*], v. 3, n. 1, p. 74–83, 2015. Disponível em: https://ieeexplore.ieee.org/document/7004835. Acesso em: 16 mai. 2023.

COSTA, A.; ROCHA, A.; DELICATO, F.; SOUZA, J. Balanceamento de carga na borda da rede usando blockchain das coisas. *In*: ANAIS DO XII SIMPóSIO BRASILEIRO DE COMPUTAçãO UBíQUA E PERVASIVA, 2020, Cuiabá. **SBC**. Porto Alegre: Sociedade Brasileira de Computação, 2020. p. 91–100. Disponível em: https://sol.sbc.org.br/index.php/sbcup/article/view/11215. Acesso em: 14 mar. 2023.

DLAMINI, S.; VENTURA, N. Resource management in fog computing: Review. *In*: 2019 INTERNATIONAL CONFERENCE ON ADVANCES IN BIG DATA, COMPUTING AND DATA COMMUNICATION SYSTEMS (ICABCD), 2019, Winterton. **IEEE**. [*S.l.*]: IEEE, 2019. p. 1–7. Disponível em: https://ieeexplore.ieee.org/document/8851016. Acesso em: 13 mar. 2023.

DOCKER, I. **Develop faster. Run anywhere**. 2022. Disponível em: https://www.docker.com/. Acesso em: 16 jul. 2023.

FACELI, K.; LORENA, A. C.; GAMA, J.; ALMEIDA, T. A. de; CARVALHO, A. C. P. de L. F. de. **Inteligência artificial**: uma abordagem de aprendizado de máquina. Rio de Janeiro: LTC, 2021.

GOMES, F. P. **Curso de estatística experimental**. 15. ed. [*S. l.*]: Editora FEALQss, 2009.

HAJAM, S. S.; SOFI, S. A. Iot-fog architectures in smart city applications: A survey. **China Communications**, [*S.l.*], v. 18, n. 11, p. 117–140, 2021. Disponível em: https://ieeexplore.ieee.org/document/9631186. Acesso em: 17 abr. 2023.

HONG, C.-H.; VARGHESE, B. Resource management in fog/edge computing: A survey on architectures, infrastructure, and algorithms. **ACM Comput. Surv.**, New York, NY, USA, v. 52, n. 5, sept. 2019. Disponível em: https://doi.org/10.1145/3326066. Acesso em: 16 mar. 2023.

JABBAR, M. A.; DEEKSHATULU, B.; CHANDRA, P. Classification of heart disease using k-nearest neighbor and genetic algorithm. **Procedia Technology**, [*S.l.*], v. 10, p. 85–94, 2013. Disponível em: https://www.sciencedirect.com/science/article/pii/S2212017313004945. Acesso em: 02 abr. 2023.

JR, D. W. H.; LEMESHOW, S.; STURDIVANT, R. X. **Applied Logistic Regression**. [*S. l.*]: John Wiley & Sons, 2013.

KANSAL, P.; KUMAR, M.; VERMA, O. Classification of resource management approaches in fog/edge paradigm and future research prospects: a systematic review. **The Journal of Supercomputing**, [*S.l.*], v. 78, p. 13145–13204, jul. 2022. Disponível em: https://doi.org/10.1007/s11227-022-04338-1. Acesso em: 25 jun. 2023.

KRAMER, O. Dimensionality reduction by unsupervised k-nearest neighbor regression. *In*: 2011 10TH INTERNATIONAL CONFERENCE ON MACHINE LEARNING AND APPLICATIONS AND WORKSHOPS, 2011, Honolulu. **IEEE**. [*S.l.*]: IEEE, 2011. p. 275–278. Disponível em: https://ieeexplore.ieee.org/document/6146983. Acesso em: 03 abr. 2023.

LAHMAR, I. B.; BOUKADI, K. Resource allocation in fog computing: A systematic mapping study. *In*: 2020 FIFTH INTERNATIONAL CONFERENCE ON FOG AND MOBILE EDGE COMPUTING (FMEC), 2020, Paris. **IEEE**. [*S.l.*]: IEEE, 2020. p. 86–93. Disponível em: https://ieeexplore.ieee.org/document/9144705. Acesso em: 13 mar. 2023.

LIU, X.; QIN, Z.; GAO, Y. Resource allocation for edge computing in iot networks via reinforcement learning. *In*: ICC 2019 - 2019 IEEE INTERNATIONAL CONFERENCE ON COMMUNICATIONS (ICC), 2019, Shanghai. **IEEE**. [*S.l.*]: IEEE, 2019. p. 1–6. Disponível em: https://ieeexplore.ieee.org/document/8761385. Acesso em: 08 abr. 2023.

MADEJ, A.; WANG, N.; ATHANASOPOULOS, N.; RANJAN, R.; VARGHESE, B. Priority-based fair scheduling in edge computing. *In*: 2020 IEEE 4TH INTERNATIONAL CONFERENCE ON FOG AND EDGE COMPUTING (ICFEC), 2020, Melbourne. **IEEE**. [*S.l.*]: IEEE, 2020. p. 39–48. Disponível em: https://ieeexplore.ieee.org/document/9139184. Acesso em: 09 abr. 2023.

MANSOURI, Y.; BABAR, M. A. A review of edge computing: Features and resource virtualization. **Journal of Parallel and Distributed Computing**, [*S.l.*], v. 150, p. 155–183, 2021. Disponível em: https://www.sciencedirect.com/science/article/pii/S0743731520304317. Acesso em: 02 mar. 2023.

MARIANO, D. Métricas de avaliação em machine learning: acurácia, sensibilidade, precisão, especificidade e f-score. **Revista Brasileira de Bioinformática**, [*S.l.*], 2021. Disponível em: https://bioinfo.com.br/metricas-de-avaliacao-em-machine-learning-acuracia-sensibilidade-precisao-especificidade-e-f-score/. Acesso em: 18 mar. 2023.

MARTINEZ, I.; HAFID, A. S.; JARRAY, A. Design, resource management, and evaluation of fog computing systems: A survey. **IEEE Internet of Things Journal**, [*S.l.*], v. 8, n. 4, p. 2494–2516, 2021. Disponível em: https://ieeexplore.ieee.org/document/9194714. Acesso em: 17 mar. 2023.

MARTINS, J. S. B. Fatiamento de rede com alocação de recursos baseada nos algoritmos q-learning e sarsa. **JSMNet Networking and Technical Reviews**, [*S.l.*], oct. 2021. Disponível em: https://doi.org/10.5281/zenodo.5544309. Acesso em: 21 mar. 2023.

MQTT.ORG. **MQTT**: The standard for iot messaging. 2023. Disponível em: https://mqtt.org/. Acesso em: 08 mai. 2023.

PASSOS, R. P.; SILIO, L.; LIMA, B.; OLIVEIRA, J. R.; PEREIRA, A.; MARTINS, G.; CAMARGO, L.; FILENI, C.; RODRIGUES, M. F.; JUNIOR, G. V. Aplicação do classificador k-nearest neighbors (knn) na área da saúde: relação cintura-quadril e pressão arterial. **Centro de Pesquisas Avançadas em Qualidade de Vida**, [*S.l.*], v. 13, n. 2, p. 1–7, jan. 2021. Disponível em: http://www.doi.org/10.36692/v13n2-18. Acesso em: 02 jun. 2023.

POTDAR, A.; NARAYAN, D.; KENGOND, S.; MULLA, M. Performance evaluation of docker container and virtual machine. **Procedia Computer Science**, [*S.l.*], v. 171, p. 1419–1428, jan. 2020. Disponível em: https://www.sciencedirect.com/science/article/pii/S1877050920311315?via%3Dihub. Acesso em: 19 mai. 2023.

RAJPUT, K.; OZA, B. A. A comparative study of classification techniques in data mining. **International Journal of Creative Research Thoughts**, [*S.l.*], v. 5, p. 154–163, 2017. Disponível em: https://ijcrt.org/viewfull.php?&p_id=IJCRT1703023. Acesso em: 21 abr. 2023.

ROCHA, K.; JúNIOR, A. Anova medidas repetidas e seus pressupostos: Análise passo a passo de um experimento. **Revista Eletrônica Perspectivas da Ciência e Tecnologia**, [*S.l.*], v. 10, p. 29, ago. 2018. Disponível em: https://revistascientificas.ifrj.edu.br/index.php/revistapct/article/view/955. Acesso em: 04 abr. 2023.

RUSMAN, J.; TAHIR, Z.; SALAM, A. E. U. Fog computing concept implementation in work error detection system of the industrial machine using support vector machine (svm). *In*: 2019 INTERNATIONAL SEMINAR ON RESEARCH OF INFORMATION TECHNOLOGY AND INTELLIGENT SYSTEMS (ISRITI), 2019, Yogyakarta. **IEEE**. [*S.l.*]: IEEE, 2019. p. 160–164. Disponível em: https://ieeexplore.ieee.org/document/9034597. Acesso em: 06 abr. 2023.

SAMANTA, A.; TANG, J. Dyme: Dynamic microservice scheduling in edge computing enabled iot. **IEEE Internet of Things Journal**, [*S.l.*], v. 7, n. 7, p. 6164–6174, 2020. Disponível em: https://ieeexplore.ieee.org/document/9042873. Acesso em: 02 mai. 2023.

SARAH, A.; NENCIONI, G.; KHAN, M. M. I. Resource allocation in multi-access edge computing for 5g-and-beyond networks. **Computer Networks**, [*S.l.*], v. 227, p. 1–21, 2023. Disponível em: https://www.sciencedirect.com/science/article/pii/S1389128623001652. Acesso em: 01 mai. 2023.

SASAKI, Y.; YOKOTANI, T. Performance evaluation of mqtt as a communication protocol for iot and prototyping. **Advances in Technology Innovation**, [*S.l.*], v. 4, n. 1, p. 21–29, jan. 2019. Disponível em: https://ojs.imeti.org/index.php/AITI/article/view/2522. Acesso em: 12 mar. 2023.

SCIKIT-LEARN. **Scikit-learn machine learning in python**. 2023. Disponível em: https://scikit-learn.org/stable/. Acesso em: 26 abr. 2023.

SHARIF, Z.; JUNG, L. T.; AYAZ, M. Priority-based resource allocation scheme for mobile edge computing. *In*: 2022 2ND INTERNATIONAL CONFERENCE ON COMPUTING AND INFORMATION TECHNOLOGY (ICCIT), 2022, Tabuk. **IEEE**. [*S.l.*]: IEEE, 2022. p. 138–143. Disponível em: https://ieeexplore.ieee.org/document/9711641. Acesso em: 09 abr. 2023.

SHARIF, Z.; JUNG, L. T.; RAZZAK, I.; ALAZAB, M. Adaptive and priority-based resource allocation for efficient resources utilization in mobile-edge computing. **IEEE Internet of Things Journal**, [*S.l.*], v. 10, n. 4, p. 3079–3093, 2023. Disponível em: https://ieeexplore.ieee.org/document/9534772. Acesso em: 19 mai. 2023.

SILVA, K. Teixeira da; MIGUEL, G.; CAVALCANTI, G.; MARINHO, M.; MADEIRO, F. Algoritmos de aprendizagem supervisionada com conjuntos de dados desbalanceados para classificação de requisitos não-funcionais. *In*: ANAIS DO 15 CONGRESSO BRASILEIRO DE INTELIGÊNCIA COMPUTACIONAL, 2021, Joinville. **SBIC**. [*S.l.*]: SBIC, 2021. p. 1–7. Disponível em: https://sbic.org.br/eventos/cbic_2021/cbic2021-125/. Acesso em: 26 abr. 2023.

SMOLA, A. J.; SCHöLKOPF, B. A tutorial on support vector regression. **Statistics and Computing**, [*S.l.*], v. 14, n. 3, p. 199–222, 2004. Disponível em: https://link.springer.com/article/10.1023/B:STCO.0000035301.49549.88. Acesso em: 07 abr. 2023.

SUYAL, M.; GOYAL, P. A two-phase classifier model for predicting the drug satisfaction of the patients based on their sentiments. *In*: BALAS, V. E.; SINHA, G. R.; AGARWAL, B.; SHARMA, T. K.; DADHEECH, P.; MAHRISHI, M. (Ed.). **Emerging Technologies in Computer Engineering: Cognitive Computing and Intelligent IoT**. Cham: Springer International Publishing, 2022. p. 79–89. Disponível em: https://link.springer.com/chapter/10.1007/978-3-031-07012-9_7. Acesso em: 01 abr. 2023.

TRAN-DANG, H.; KIM, D.-S. An information framework for internet of things services in physical internet. **IEEE Access**, [*S.l.*], v. 6, p. 43967–43977, 2018. Disponível em: https://ieeexplore.ieee.org/document/8429894. Acesso em: 19 abr. 2023.

TRAN-DANG, H.; KIM, D.-S. Task priority-based resource allocation algorithm for task offloading in fog-enabled iot systems. *In*: 2021 INTERNATIONAL CONFERENCE ON INFORMATION NETWORKING (ICOIN), 2021, Jeju Island. **IEEE**. [*S.l.*]: IEEE, 2021. p. 674–679. Disponível em: https://ieeexplore.ieee.org/document/9333992. Acesso em: 21 mar. 2023.

VIEIRA, M. N. **Medições e avaliações comparativas de desempenho e energia de algoritmos de machine learning para mitigar ameaças de disponibilidade em ambiente iot.** 2021. 125 f. Dissertação (Mestrado Profissional em Tecnologia da Informação) – Instituto Federal de Educação da Paraíba, João Pessoa, 2021. Disponível em: https://repositorio.ifpb.edu.br/handle/177683/1736. Acesso em: 18 mar. 2023.

WANG, K.; TAN, Y.; SHAO, Z.; CI, S.; YANG, Y. Learning-based task offloading for delay-sensitive applications in dynamic fog networks. **IEEE Transactions on Vehicular Technology**, [*S.l.*], v. 68, n. 11, 2019. Disponível em: https://ieeexplore.ieee.org/document/8848447. Acesso em: 28 abr. 2023.

WANG, Z.; LV, T.; CHANG, Z. Computation offloading and resource allocation based on distributed deep learning and software defined mobile edge computing. **Computer Networks**, [*S.l.*], v. 205, jan. 2022. Disponível em: https://www.sciencedirect.com/science/article/pii/S1389128621005843. Acesso em: 13 abr. 2023.

YIN, C.; LI, T.; QU, X.; YUAN, S. An optimization method for resource allocation in fog computing. *In*: 2020 INTERNATIONAL CONFERENCES ON INTERNET OF THINGS (ITHINGS) AND IEEE GREEN COMPUTING AND COMMUNICATIONS (GREENCOM) AND IEEE CYBER, PHYSICAL AND SOCIAL COMPUTING (CPSCOM) AND IEEE SMART DATA (SMARTDATA) AND IEEE CONGRESS ON CYBERMATICS (CYBERMATICS), 2020, Rhodes. **IEEE**. [*S.l.*]: IEEE, 2020. p. 821–828. Disponível em: https://ieeexplore.ieee.org/document/9291558. Acesso em: 22 mar. 2023.

ZHOU, C.; GONG, C.; HUI, H.; LIN, F.; ZENG, G. A task-resource joint management model with intelligent control for mission-aware dispersed computing. **China Communications**, [*S.l.*], v. 18, n. 10, p. 214–232, 2021. Disponível em: https://ieeexplore.ieee.org/document/9597648. Acesso em: 06 mai. 2023.