UNIVERSIDADE FEDERAL DE SANTA CATARINA
CAMPUS FLORIANÓPOLIS
PROGRAMA DE PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

Henrique Antonio Buzin Vargas

Uma Arquitetura Modular para Ambientes Inter-Névoa

Florianópolis
2025

Henrique Antonio Buzin Vargas

Uma Arquitetura Modular para Ambientes Inter-Névoa

Dissertação submetida ao Programa de Pós-Graduação em Ciência da Computação da Universidade Federal de Santa Catarina para a obtenção do título de mestre em Ciência da Computação.
Orientador: Prof.ª Dr.ª Patricia Della Méa Plentz

Florianópolis
2025

Ficha de identificação da obra

A ficha de identificação é elaborada pelo próprio autor.

Orientações em:

http://portalbu.ufsc.br/ficha

Henrique Antonio Buzin Vargas

Uma Arquitetura Modular para Ambientes Inter-Névoa

O presente trabalho em nível de mestrado foi avaliado e aprovado por banca examinadora composta pelos seguintes membros:

Prof. Dr. Odorico Machado Mendizabal
Universidade Federal de Santa Catarina

Prof.(a) xxxx, Dr(a).
Instituição xxxx

Prof.ª Dr.ª Atslands Rego da Rocha
Universidade Federal do Ceará

Certificamos que esta é a **versão original e final** do trabalho de conclusão que foi julgado adequado para obtenção do título de mestre em Ciência da Computação.

Coordenação do Programa de
Pós-Graduação

Prof.ª Dr.ª Patricia Della Méa Plentz
Orientador

Florianópolis, 2025.

Dedico este trabalho a todos os meus professores,
sendo, meus pais, os maiores entre eles.

AGRADECIMENTOS

Neste momento, sinto uma gratidão imensa por cada pessoa que fez parte desta jornada. Agradeço de coração aos meus pais e à minha namorada, cujos amores incondicionais e apoios constantes foram a base sólida sobre a qual construí meus sonhos e conquistas. Sem vocês, nada disso teria sido possível.

À UFSC (Universidade Federal de Santa Catarina) e a todos os seus membros, expresso meu profundo agradecimento. Cada professor, funcionário e colega desempenhou um papel crucial no meu crescimento acadêmico. Através do ensino de qualidade, das valiosas ferramentas e dos recursos disponibilizados, pude adquirir o conhecimento necessário para realizar este trabalho.

Em especial, minha orientadora Patrícia Della Méa Plentz, você foi uma inspiração e guia ao longo dessa jornada. Sua dedicação, sabedoria e orientação foram fundamentais para a construção desta qualificação. Agradeço por compartilhar seu conhecimento, suas perspectivas e por me motivar a alcançar meu melhor desempenho. Sua generosidade em investir tempo e energia em minha formação é um presente inestimável.

Um agradecimento especial à LexisNexis, em especial a Mauro Marques e Alysson Oliveira. Sua colaboração e apoio foram cruciais para o desenvolvimento deste trabalho. Suas expertise e capacidade analítica, com seu suporte técnico e encorajamento constante, enriqueceram imensamente minha pesquisa. Sou profundamente grato(a) a ambos por sua dedicação e por acreditarem no potencial deste projeto.

A todos vocês, meus pais, minha namorada, minha orientadora e a equipe da LexisNexis, sou profundamente grato(a). Seus ensinamentos, apoio e encorajamento foram a força motriz que me impulsionou a superar desafios e alcançar meus objetivos. Cada palavra de incentivo, cada gesto de apoio e cada momento de aprendizado deixaram uma marca indelével em minha vida acadêmica e pessoal.

Expresso minha sincera admiração e gratidão por cada contribuição que fizeram em minha jornada. Vocês são verdadeiros pilares em minha vida, e sou grato(a) por terem acreditado em mim e por terem compartilhado esse caminho comigo.

”A educação é a arma mais poderosa que
você pode usar para mudar o mundo.”
(Nelson Mandela)

RESUMO

Este trabalho apresentou o desenvolvimento e a avaliação de uma arquitetura modular para ambientes de computação em névoa, projetada para suportar comunicação direta entre diferentes domínios e integrar dispositivos de borda, nós de névoa e a nuvem, tomando como referência o modelo do OpenFog Consortium. A solução foi construída com base nos pilares de escalabilidade, abertura, autonomia, programabilidade e hierarquia, considerando o contexto de cidades inteligentes. A proposta incorporou abstração de protocolos para permitir interoperabilidade entre diferentes tecnologias de comunicação, além de um mecanismo de balanceamento dinâmico de carga e agregação de dados para otimizar o uso de recursos e reduzir o tráfego de dados para a nuvem. A metodologia envolveu a simulação de dois dias de operação contínua, com dispositivos distribuídos em dois domínios de névoa e geração periódica de dados, analisando métricas de latência, consumo de memória, throughput, taxa de entrega e fator de agregação. Os resultados indicaram latência ponta a ponta média de 15,6 ms, taxa de entrega de 100% e redução de aproximadamente 99,97% nas requisições à nuvem, mantendo desempenho compatível com aplicações sensíveis ao tempo. A comparação com trabalhos como o EXEGESIS evidenciou diferenças no gerenciamento de serviços e na flexibilidade de adaptação a novos cenários, destacando vantagens na separação entre a camada de aplicação e a de protocolos. Entre as perspectivas futuras estão a implementação de autenticação e criptografia ponta a ponta, estratégias de recuperação automática e validação da arquitetura em domínios como saúde conectada e transporte inteligente.

Palavras-chave: computação em névoa. interoperabilidade. balanceamento de carga.

ABSTRACT

This research presented the development and evaluation of a modular architecture for fog computing environments, designed to support direct communication between different domains and to integrate edge devices, fog nodes, and the cloud, taking the OpenFog Consortium reference model as a guideline. The solution was built on the pillars of scalability, openness, autonomy, programmability, and hierarchy, within the context of smart cities. The proposal incorporated protocol abstraction to enable interoperability among different communication technologies, along with a dynamic load balancing mechanism and data aggregation to optimize resource usage and reduce cloud traffic. The methodology involved simulating two days of continuous operation, with devices distributed across two fog domains and generating periodic data, analyzing metrics such as latency, memory consumption, throughput, delivery rate, and aggregation factor. The results indicated an average end-to-end latency of 15.6 ms, a delivery rate of 100%, and approximately 99.97% reduction in cloud requests, maintaining performance compatible with time-sensitive applications. Comparison with works such as EXEGESIS highlighted differences in service management and adaptability to new scenarios, with advantages in the separation between the application and protocol layers. Future perspectives include implementing end-to-end authentication and encryption, automatic recovery strategies, and validating the architecture in domains such as connected healthcare and intelligent transportation.

Keywords: fog computing. interoperability. load balancing.

LISTA DE FIGURAS

LISTA DE QUADROS

LISTA DE TABELAS

SUMÁRIO

1 INTRODUÇÃO

Estamos vivenciando um crescimento exponencial no número de dispositivos conectados à internet, impulsionado principalmente pela expansão da Internet das Coisas (IoT). Essa categoria abrange uma ampla gama de equipamentos, como smartphones, relógios inteligentes, sensores ambientais, eletrodomésticos inteligentes e veículos autônomos, entre outros. Esses dispositivos diferem significativamente entre si em termos de capacidade computacional, armazenamento local, autonomia energética e protocolos de comunicação utilizados, o que introduz desafios consideráveis de interoperabilidade e gerenciamento em larga escala.

Paralelamente, cresce a exigência por processamento de dados em tempo quase real, especialmente em aplicações de missão crítica. Veículos autônomos, por exemplo, precisam realizar inferências e tomar decisões instantâneas com base em grandes volumes de dados sensoriais (Markakis et al., 2017). Da mesma forma, em cenários de saúde digital, dispositivos vestíveis como *smartwatches* monitoram sinais vitais continuamente e podem enviar alertas automáticos a instituições médicas, possibilitando respostas emergenciais, como o envio imediato de ambulâncias em situações críticas (Cassel et al., 2024).

Diante desse cenário, a arquitetura de computação em névoa (*fog computing*) surgiu como uma solução eficaz para suprir a necessidade de processamento e armazenamento mais próximos da borda da rede. Essa abordagem reduz a latência e melhora a eficiência do sistema, sendo especialmente relevante em aplicações que exigem respostas rápidas e precisas. A computação em névoa ainda pode atuar como intermediária entre os dispositivos e a nuvem, realizando um pré-processamento local dos dados antes de enviá-los para a nuvem, o que facilita a integração, reduz o tráfego e diminui a carga sobre os servidores centrais.

Na Figura 1, temos ilustrado uma arquitetura básica de névoa, os dispositivos enviam dados para os equipamentos da camada de névoa, onde ocorre o processamento local. Os dados processados podem então ser devolvidos para os dispositivos ou encaminhados à nuvem, dependendo do contexto da aplicação.

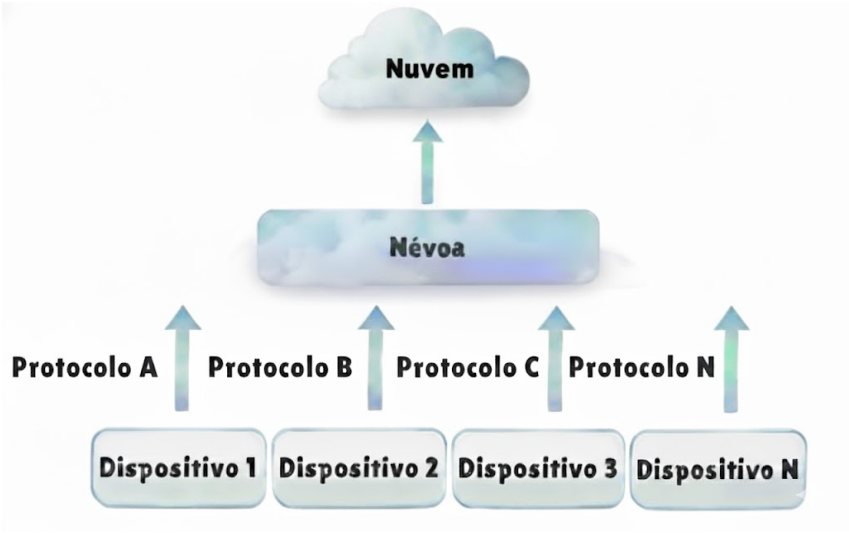
No entanto, a diversidade de dispositivos e protocolos de comunicação ainda representa um obstáculo à interoperabilidade plena. A ausência de um padrão universalmente aceito dificulta a comunicação eficiente entre equipamentos heterogêneos.

Com o objetivo de mitigar as limitações de interoperabilidade identificadas, este trabalho apresenta uma arquitetura modular de névoa capaz de estabelecer comunicação entre domínios distintos e gerenciar serviços de forma adaptável e distribuída.

1.1 OBJETIVOS

Nas seções abaixo estão descritos o objetivo geral e os objetivos específicos.

Figura 1 – Arquitetura básica de comunicação na computação em névoa.



Fonte: Do autor.

1.1.1 Objetivo Geral

Desenvolver uma arquitetura modular de computação em névoa voltada à comunicação entre diferentes névoas, com foco na integração de novos dispositivos, e na facilitação da interoperabilidade entre protocolos distintos, considerando aplicações cujos requisitos possam ser contemplados por essa abordagem.

1.1.2 Objetivos Específicos

Para atender ao objetivo geral descrito nesta seção, apresentam-se os seguintes objetivos específicos:

- Desenvolver uma arquitetura modular em camadas para a computação em névoa, definindo as funções e responsabilidades de cada camada, com foco na integração de novos dispositivos.
- Projetar e implementar um sistema de comunicação entre névoas, visando facilitar a interoperabilidade e reduzir a latência entre dispositivos que utilizam diferentes padrões e tecnologias, atendendo a aplicações cujos requisitos sejam compatíveis com a arquitetura.
- Desenvolver e integrar mecanismos de conversão de protocolos, permitindo a comunicação entre dispositivos com diferentes padrões e tecnologias.
- Avaliar a eficácia da arquitetura modular proposta por meio da medição de tempo de resposta, latência e quantidade de pacotes com e sem falhas, em simulações que reproduzam requisitos específicos das aplicações.

- Documentar os resultados obtidos e propor diretrizes, fornecendo orientações e melhores práticas para futuras implementações e pesquisas que atendam a requisitos similares.

2 FUNDAMENTAÇÃO TEÓRICA

Primeiramente, para que se possa compreender o contexto deste trabalho, é necessária a revisão de alguns conceitos fundamentais. Desta forma, iniciamos com a conceituação do que é computação distribuída e em névoa e avançamos nos conceitos pertinentes a este trabalho.

2.1 COMPUTAÇÃO EM NUVEM

A computação em nuvem é uma tecnologia que permite o acesso remoto a recursos computacionais, como servidores, armazenamento de dados, redes e software, através da internet. Essa abordagem oferece uma maneira eficiente de fornecer serviços de TI sob demanda, proporcionando flexibilidade, escalabilidade e redução de custos. A computação em nuvem envolve a utilização de data centers distribuídos para fornecer esses recursos, permitindo que os usuários acessem e utilizem capacidades computacionais de acordo com suas necessidades, sem a necessidade de investir em hardware próprio (Tanenbaum; Bos, 2015).

Os modelos de serviço da computação em nuvem são geralmente classificados em três categorias principais: Infraestrutura como Serviço (IaaS), Plataforma como Serviço (PaaS) e Software como Serviço (SaaS). No modelo IaaS, os provedores de nuvem oferecem recursos virtualizados, como servidores e armazenamento, que os clientes podem configurar e gerenciar conforme suas necessidades. Esse modelo é ideal para empresas que necessitam de controle sobre a infraestrutura e desejam a flexibilidade de escalar seus recursos conforme a demanda. Exemplos de IaaS incluem Amazon Web Services (AWS) e Microsoft Azure.

No modelo PaaS, a nuvem fornece uma plataforma que permite aos desenvolvedores criar, testar e implantar aplicações sem se preocupar com a gestão da infraestrutura subjacente. Tanenbaum e Bos destacam que o PaaS simplifica o processo de desenvolvimento ao fornecer um ambiente integrado com ferramentas e serviços necessários para a construção de aplicações. Exemplos de PaaS incluem Google App Engine e Heroku.

O modelo SaaS oferece aplicações de software através da internet, eliminando a necessidade de instalação e manutenção de software localmente nos dispositivos dos usuários. As aplicações SaaS são acessíveis via navegador web, proporcionando uma experiência de uso simplificada e atualizações automáticas. Exemplos populares de SaaS incluem Google Workspace, Microsoft Office 365 e Salesforce.

Tanenbaum e Bos enfatizam que a computação em nuvem também proporciona vantagens significativas em termos de continuidade de negócios e recuperação de desastres. Ao armazenar dados e executar aplicações em data centers distribuídos geograficamente, a nuvem garante que as operações possam ser retomadas rapidamente

em caso de falhas locais, aumentando a resiliência e a segurança das operações.

2.2 COMPUTAÇÃO EM NÉVOA

A computação em névoa, também conhecida como fog computing, surge como uma extensão do paradigma de computação em nuvem, oferecendo uma solução para as limitações de latência e largura de banda enfrentadas na nuvem centralizada. Diferente da computação em nuvem, onde os dados são processados em data centers distantes, a computação em névoa distribui recursos computacionais, de armazenamento e serviços ao longo da borda da rede, mais próximos dos dispositivos finais. Esta abordagem hierárquica envolve dispositivos finais, nós de névoa e a nuvem centralizada, permitindo a distribuição eficiente de tarefas e o processamento local de dados, essencial para aplicações que exigem alta responsividade e baixa latência (Rahmani et al., 2018).

Entre as principais vantagens da computação em névoa, destaca-se a redução significativa da latência, crucial para aplicações em tempo real, como veículos autônomos e sistemas de saúde conectados. Além disso, a economia de largura de banda é alcançada através do pré-processamento e filtragem de dados localmente, diminuindo a quantidade de dados transmitidos para a nuvem. A segurança e privacidade dos dados são aprimoradas ao processar informações sensíveis localmente, reduzindo os riscos durante a transmissão. A alta disponibilidade e resiliência dos serviços também são beneficiadas pela distribuição de recursos em múltiplos nós de névoa, aumentando a robustez do sistema contra falhas.

No contexto da Internet das Coisas (IoT), a computação em névoa revela-se particularmente relevante, suportando aplicações como cidades inteligentes, saúde conectada e a Indústria 4.0. Em cidades inteligentes, a névoa possibilita o monitoramento e controle eficientes de tráfego, energia e segurança pública. Na área da saúde, permite o monitoramento remoto de pacientes e análise de dados médicos em tempo real. Na indústria, a névoa facilita a automação, manutenção preditiva e otimização de processos, promovendo a eficiência operacional e a inovação. Esses exemplos demonstram como a computação em névoa pode transformar diversos setores, proporcionando benefícios significativos em termos de desempenho e confiabilidade.

Apesar das vantagens, a computação em névoa enfrenta desafios como a heterogeneidade dos dispositivos, a complexidade na gestão de recursos distribuídos e a necessidade de padrões interoperáveis.

2.3 PROTOCOLOS

Os protocolos de comunicação são fundamentais para o funcionamento das redes de computadores e para a troca eficiente e segura de informações entre dispo-

sitivos. Eles definem um conjunto de regras e normas que permitem a comunicação entre diferentes sistemas, sejam eles locais ou distribuídos globalmente (Tanenbaum; Wetherall, 2011). Na computação em névoa, os protocolos são usados para garantir a comunicação eficiente e segura entre dispositivos e servidores locais, permitindo a troca de informações de forma robusta e confiável.

HTTP (Hypertext Transfer Protocol) é um protocolo de requisição-resposta usado principalmente para a troca de informações na web. Ele opera sobre a camada de transporte TCP/IP e permite a comunicação entre clientes (navegadores web) e servidores. Os métodos mais comuns incluem GET (para recuperar dados) e POST (para enviar dados). O HTTP é sem estado, o que significa que cada requisição é independente, simplificando o design do servidor, mas pode necessitar de gerenciamento adicional para manter sessões de usuário (Gourley et al., 2002).

WebSocket é um protocolo de comunicação que proporciona um canal bidirecional e full-duplex sobre uma única conexão TCP. Diferente do HTTP, que segue um modelo de requisição-resposta, o WebSocket permite que dados sejam enviados e recebidos de ambos os lados a qualquer momento após a conexão inicial. Isso o torna ideal para aplicações que necessitam de atualizações em tempo real, como chats, jogos online e sistemas de trading financeiro, oferecendo menor latência e maior eficiência (Wang et al., 2013).

SFTP (Secure File Transfer Protocol) é um protocolo para transferência segura de arquivos que utiliza criptografia SSH para proteger a confidencialidade e integridade dos dados durante a transferência. Diferente do FTP tradicional, que transmite dados em texto claro, o SFTP encripta tanto os comandos quanto os dados, proporcionando uma camada adicional de segurança. É amplamente utilizado em ambientes que requerem troca segura de arquivos, como empresas e instituições acadêmicas, onde a proteção dos dados é crucial (Barrett; Silverman; Byrnes, 2005).

CoAP (Constrained Application Protocol) é um protocolo projetado para dispositivos restritos e redes de baixa capacidade, típico no contexto da Internet das Coisas (IoT). Baseado no modelo de requisição-resposta do HTTP, CoAP é otimizado para operar sobre UDP, resultando em menor sobrecarga de comunicação e melhor desempenho em redes de baixa capacidade. CoAP suporta mecanismos de confiabilidade e é interoperável com HTTP, facilitando a integração entre dispositivos IoT e a web tradicional (Bormann; Castellani; Shelby, 2012).

2.4 GRAPHQL

GraphQL é uma linguagem de consulta e manipulação de dados para APIs que foi desenvolvida pelo Facebook em 2012 e lançada como um projeto de código aberto em 2015. Em contraste com o tradicional modelo REST (Representational State Transfer), GraphQL permite que os clientes solicitem exatamente os dados de que

precisam, sem a necessidade de receber informações excedentes. Essa capacidade de seleção pode reduzir o volume de dados transferidos pela rede, resultando em uma maior eficiência tanto na comunicação quanto no desempenho geral das aplicações (Silveira, 2019).

Uma das principais vantagens do GraphQL é sua flexibilidade na recuperação de dados. Enquanto a arquitetura REST requer múltiplas requisições para diferentes endpoints para compilar dados relacionados, o GraphQL permite que todas essas informações sejam obtidas através de uma única consulta. Isso é possível porque o cliente especifica exatamente quais campos e relacionamentos deseja recuperar, e o servidor responde com os dados estruturados conforme solicitado. Esse modelo de consulta única não apenas simplifica o processo de desenvolvimento, mas também minimiza o tempo de resposta e a carga na rede.

Além disso, o GraphQL facilita a evolução das APIs. No modelo REST, a adição de novos campos ou a alteração da estrutura de dados pode exigir a criação de novas versões de endpoints para garantir a compatibilidade retroativa. Com GraphQL, as APIs podem evoluir de maneira mais fluida. Novos campos podem ser adicionados às respostas sem impactar os clientes existentes, pois cada cliente continua a solicitar apenas os dados específicos de que precisa.

Adicionalmente, a abordagem de GraphQL também melhora a experiência de desenvolvimento ao fornecer uma documentação mais clara e precisa. A própria estrutura da linguagem e as ferramentas associadas, como o GraphiQL, permitem que os desenvolvedores explorem e testem as APIs de maneira interativa. Isso contrasta com a documentação tradicional de APIs REST, que pode ser menos intuitiva e exigir mais esforço para ser mantida e compreendida.

2.5 HPCC SYSTEMS

O HPCC Systems (*High Performance Computing Cluster*) é uma plataforma de processamento massivamente paralelo, de código aberto, desenvolvida pela divisão HPCC Systems da LexisNexis Risk Solutions. Seu objetivo é lidar com grandes volumes de dados, desde a ingestão até a entrega de produtos de dados, com tempos de resposta reduzidos. Criado antes da popularização do Hadoop, não se baseia no paradigma *map-reduce*, possuindo uma arquitetura própria (Taylor, 2022).

A plataforma é composta por diversos servidores de infraestrutura responsáveis por funções como compilação de código, gerenciamento distribuído de dados e execução de trabalhos (*workunits*). O processamento é realizado por dois tipos de clusters: Thor e ROXIE.

O Thor é voltado para operações intensivas em grandes conjuntos de dados, realizando tarefas como transformação, limpeza, integração e análise em larga escala. É utilizado como ferramenta de *back office*, transformando dados brutos em produtos

finais e executando análises complexas.

O ROXIE (*Rapid Online Xml Inquiry Engine*) é destinado a consultas rápidas e atendimento a solicitações de usuários finais, buscando oferecer respostas em tempos muito reduzidos, frequentemente inferiores a um segundo. É, portanto, o componente voltado ao acesso e entrega de dados processados.

Ambos os clusters utilizam a linguagem ECL (*Enterprise Control Language*), de natureza declarativa. Nessa abordagem, o programador especifica o que deve ser feito, enquanto o sistema determina a forma mais eficiente de execução. Isso contribui para simplificar o desenvolvimento e reduzir a possibilidade de erros.

A arquitetura do HPCC Systems foi concebida para permitir escalabilidade progressiva e operação integrada, incorporando desde a ingestão e transformação de dados até a disponibilização dos resultados, sem depender de ferramentas externas para compor o ambiente de produção. Por ser de código aberto, pode ser implantado tanto em ambientes físicos com hardware convencional quanto em configurações baseadas em contêineres, adaptando-se às necessidades de diferentes cenários de processamento de dados.

3 REVISÃO BIBLIOGRÁFICA

Para a realização deste trabalho, foram conduzidas pesquisas em bases de dados de artigos científicos, utilizando as seguintes palavras-chave: *'fog'* e *'fog' AND 'architecture'*. Foi aplicado um filtro temporal abrangendo os últimos cinco anos, isto é, de 2020 a 2025. A Tabela 1 apresenta a quantidade de publicações encontradas em cada base de dados para os respectivos termos de busca.

Tabela 1 – Quantidade de artigos por palavra-chave e base de dados

Palavra-chave	**Banco de Dados**	**Quantidade de Artigos**
'fog'	Scopus IEEE	34.309 8.524
'fog' AND 'architecture'	Scopus IEEE	12.822 2.502

Fonte: Do autor.

Com base nos dados apresentados, observa-se que a base Scopus possui o maior número de artigos indexados, destacando-se como a principal fonte de publicações sobre o tema. Por outro lado, a base IEEE Xplore apresentou uma quantidade inferior de resultados, embora ainda represente um repositório relevante para estudos na área de computação em névoa.

Além disso, foram aplicados os seguintes critérios de inclusão para seleção dos estudos analisados nesta pesquisa:

- Artigos de pesquisa completos;
- Artigos redigidos em língua inglesa;
- Publicações classificadas no sistema CAPES Qualis[1] entre os estratos A1 e B2;
- Trabalhos revisados por pares e publicados em conferências ou periódicos científicos reconhecidos.

3.1 TRABALHOS CORRELATOS

Foram lidos títulos e resumo de cento e vinte e cinco artigos, dos quais vinte e cinco foram lidos completamente, e desses foram selecionados quatro que apresentam propostas semelhantes à proposta deste trabalho e por isso serão detalhados nas subseções seguintes.

[1] A Coordenação de Aperfeiçoamento de Pessoal de Nível Superior (CAPES) é uma agência do governo brasileiro responsável pela avaliação e fomento da pós-graduação.

3.1.1 Microservice instances selection and load balancing in fog computing using deep reinforcement learning approach

Os autores propõem uma abordagem para a seleção de instâncias de microsserviços e para o balanceamento de carga em ambientes de computação em névoa, empregando técnicas de Deep Reinforcement Learning. O estudo parte do princípio de que aplicações da Internet das Coisas podem ser estruturadas de forma modular, utilizando arquiteturas baseadas em microsserviços implantados próximos à borda da rede. Essa configuração, denominada computação fog nativa, busca superar as limitações de latência e de custo de comunicação dos modelos puramente em nuvem ao permitir o processamento distribuído de tarefas entre dispositivos, nós de névoa e servidores centrais (Boudieb; Ayed; Harous, 2024).

No modelo proposto, cada aplicação é composta por um conjunto de microsserviços interdependentes organizados em planos de serviço. O desafio central é selecionar esses planos de forma dinâmica e eficiente em ambientes muito variáveis, nos quais a carga de trabalho e as condições de rede mudam constantemente. Para isso, os autores apresentam o método Microservice Instances Selection Policy, que utiliza aprendizado por reforço profundo para otimizar a alocação de microsserviços, reduzindo atrasos e distribuindo a carga de forma equilibrada entre os nós disponíveis. O algoritmo incorpora mascaramento de ações, mapeamento adaptativo de ações e um esquema de experience replay ajustado para melhorar a eficiência e a estabilidade do treinamento.

Os resultados experimentais indicam que a política proposta reduz a taxa média de falhas em até sessenta e cinco por cento e melhora o balanceamento de carga em cerca de quarenta e cinco por cento em relação a algoritmos convencionais. O trabalho reforça a importância da colaboração entre névoa e nuvem para garantir escalabilidade, desempenho e resiliência em aplicações sensíveis ao tempo.

3.1.2 Enhancing modular application placement in a hierarchical fog computing: A latency and communication cost-sensitive approach

Neste trabalho é proposto uma estratégia de posicionamento de aplicações modulares em uma arquitetura hierárquica de computação em névoa, denominada Least Impact X, abreviada como LI X. O estudo aborda desafios de alocação de recursos em cenários com vários níveis de névoa, buscando reduzir o tempo de resposta e o custo de comunicação em aplicações sensíveis à latência. A motivação é evitar o esgotamento de recursos nas camadas mais próximas da borda quando requisitos específicos de uma aplicação são priorizados sem considerar o efeito global na hierarquia (Oliveira; Santos; Mendes, 2024).

A proposta distribui os módulos entre os níveis de névoa de forma a minimizar o

impacto sobre recursos críticos próximos à borda e, ao mesmo tempo, reduzir o tempo de resposta e o consumo de largura de banda. O algoritmo considera métricas de qualidade de serviço para equilibrar a utilização de recursos e evitar ociosidade em níveis superiores. Em simulações no iFogSim, a abordagem supera alternativas da literatura ao apresentar menor latência média e menor custo de comunicação, além de uso mais eficiente de recursos computacionais.

3.1.3 A Scalable Fog Computing Solution for Industrial Predictive Maintenance and Customization

É apresentado uma solução de computação em névoa para manutenção preditiva industrial e para customização de sistemas de Internet das Coisas industriais. A arquitetura é modular e flexível e combina sensores Nicla Sense ME, um concentrador em Raspberry Pi e um modelo LSTM para análise preditiva em tempo quase real. O diferencial é um ambiente sandbox que permite integrar modelos de aprendizado de máquina de forma isolada, preservando a segurança e a estabilidade do núcleo da plataforma. A solução utiliza Docker para encapsular módulos e escalar para dezenas de nós sensores sem degradação perceptível de desempenho, adotando ainda práticas de segurança como autenticação multifatorial, criptografia e detecção de anomalias (D'Agostino; Longo; Palmieri, 2025).

Nos experimentos, o sistema apresentou erro quadrático médio baixo e alta acurácia em laboratório e manteve desempenho consistente em ambiente industrial. O erro quadrático médio, também referido como RMSE, mede a diferença média entre valores previstos e valores observados e quanto menor melhor. A acurácia indica a proporção de previsões corretas em relação ao total. Os resultados reforçam a importância de integrar técnicas de inteligência artificial em ambientes industriais com recursos restritos para reduzir paradas, otimizar uso de recursos e permitir personalização.

3.1.4 Testbed analysis of multi fog architecture for interoperable Internet of Things

Os autores investigam uma arquitetura multinévoa com interoperabilidade semântica para aplicações em tempo real, implementada e avaliada em ambiente experimental. O objetivo é permitir que nós de névoa troquem informações e serviços de forma direta e eficaz, reduzindo a dependência de processamento em nuvem. A proposta desloca para a névoa funções usuais do modelo semântico, como filtragem, anotação e mapeamento, o que permite realizar parte do processamento localmente e diminui o trânsito de dados para servidores distantes (Rahman; Hussain, 2024).

No protótipo, dispositivos Raspberry Pi 2 Model B executam semântica local, armazenamento e serviços. Quando um nó está sobrecarregado, ele avalia se vale a pena repassar a tarefa para outro nó vizinho com base em tempo e energia estima-

dos. Caso nenhum candidato seja melhor, a tarefa é encaminhada para a nuvem. A arquitetura utiliza duas camadas, chamadas L2 Fog e L1 Fog. A primeira atua como gateway para sensores e atuadores usando 6LoWPAN e a segunda concentra serviços e banco de dados com Apache e MariaDB. Há ainda sincronização hierárquica baseada em identificadores e um mecanismo de controle de fila que define quando iniciar a transferência de tarefas.

Os experimentos mostram que o tempo médio de ida e volta de uma mensagem, conhecido como Round Trip Time ou RTT, foi reduzido em cerca de trinta e sete por cento em comparação ao uso direto da nuvem. O RTT é o intervalo total que uma solicitação leva para ir do dispositivo ao destino e retornar com a resposta e é um indicador direto de responsividade. No estudo, valores médios passaram de aproximadamente duzentos e oitenta milissegundos na nuvem para aproximadamente cento e três milissegundos com a arquitetura multinévoa. Também houve redução de cerca de doze por cento no tempo de execução de tarefas quando a transferência entre nós de névoa estava ativa. O tempo de execução é o período entre o envio de uma tarefa e a conclusão do processamento. Os autores ainda registram diminuição média de dez por cento no consumo de energia dos nós em relação ao funcionamento sem transferência entre nós. O consumo de energia é a quantidade de energia elétrica utilizada pelos dispositivos para realizar as tarefas e é crucial em cenários com recursos limitados.

Esses ganhos são explicados pelo processamento local de parte da semântica, pela divisão de carga entre nós próximos e pelo envio à nuvem apenas quando há real vantagem.

3.2 COMPARATIVO COM TRABALHOS CORRELATOS

Conforme apresentado nas subseções anteriores, os trabalhos analisados exploram diferentes perspectivas da computação em névoa, cada um contribuindo de maneira específica para o avanço da área. De modo geral, observa-se que as propostas anteriores tratam de aspectos pontuais, como otimização de desempenho, posicionamento de módulos e modularidade aplicada a contextos específicos, enquanto o presente trabalho reúne e amplia esses conceitos, com ênfase na interoperabilidade e comunicação entre domínios distintos de névoa.

O trabalho de Boudieb et al. (Boudieb; Ayed; Harous, 2024) propõe o uso de aprendizado por reforço profundo para selecionar instâncias de microsserviços e realizar o balanceamento de carga de forma adaptativa. A abordagem demonstra eficiência na redução de falhas e na melhoria da distribuição de tarefas, mas permanece restrita a um único domínio de névoa, sem prever cooperação entre diferentes nós. Por sua vez, Oliveira et al. (Oliveira; Santos; Mendes, 2024) apresentam uma estratégia hierárquica para o posicionamento de aplicações modulares, buscando reduzir a latência e

o custo de comunicação. Embora eficiente na alocação de módulos, a proposta não contempla mecanismos de interoperabilidade entre diferentes camadas ou arquiteturas de névoa.

A solução de D'Agostino, Longo e Palmieri (D'Agostino; Longo; Palmieri, 2025), voltada à manutenção preditiva industrial, introduz modularidade com o uso de contêineres e integração de aprendizado de máquina em tempo quase real. Essa proposta apresenta alta escalabilidade e segurança, mas limita-se ao contexto industrial e não aborda a colaboração entre domínios de névoa. Já Rahman e Hussain (Rahman; Hussain, 2024) propõem uma arquitetura multinévoa com interoperabilidade semântica, permitindo que nós de névoa colaborem com base em métricas como tempo de resposta e consumo de energia. Apesar dos avanços em comunicação distribuída, o modelo ainda depende de mecanismos fixos de decisão e não incorpora uma estrutura modular interna aos nós.

A arquitetura proposta neste trabalho diferencia-se por integrar múltiplas camadas modulares dentro de cada nó de névoa, combinando flexibilidade estrutural, interoperabilidade e cooperação entre domínios. Cada camada possui responsabilidades bem definidas: a camada de protocolos realiza a comunicação com diferentes dispositivos e formatos de dados; a camada de processamento gerencia o fluxo interno, realiza o armazenamento temporário e coordena as tarefas; e a camada de serviço executa aplicações configuráveis que processam e preparam os dados para as etapas seguintes. Essa organização favorece a manutenção, a escalabilidade e a integração eficiente de novos serviços, permitindo que partes específicas do sistema sejam adaptadas sem comprometer o funcionamento geral.

Além da modularidade interna, outro diferencial importante é a comunicação entre domínios de névoa, que permite o balanceamento cooperativo de carga e o redirecionamento de solicitações entre diferentes regiões conforme a disponibilidade de recursos. O nó primário atua como ponto de entrada e registro dos dispositivos conectados, garantindo distribuição equilibrada e continuidade de operação.

Com base nesses conceitos, o próximo capítulo apresenta em detalhes a arquitetura modular proposta, descrevendo suas camadas, fluxos de comunicação e mecanismos de integração entre névoas, evidenciando como essa estrutura supera as limitações identificadas nos trabalhos correlatos.

4 PROPOSTA

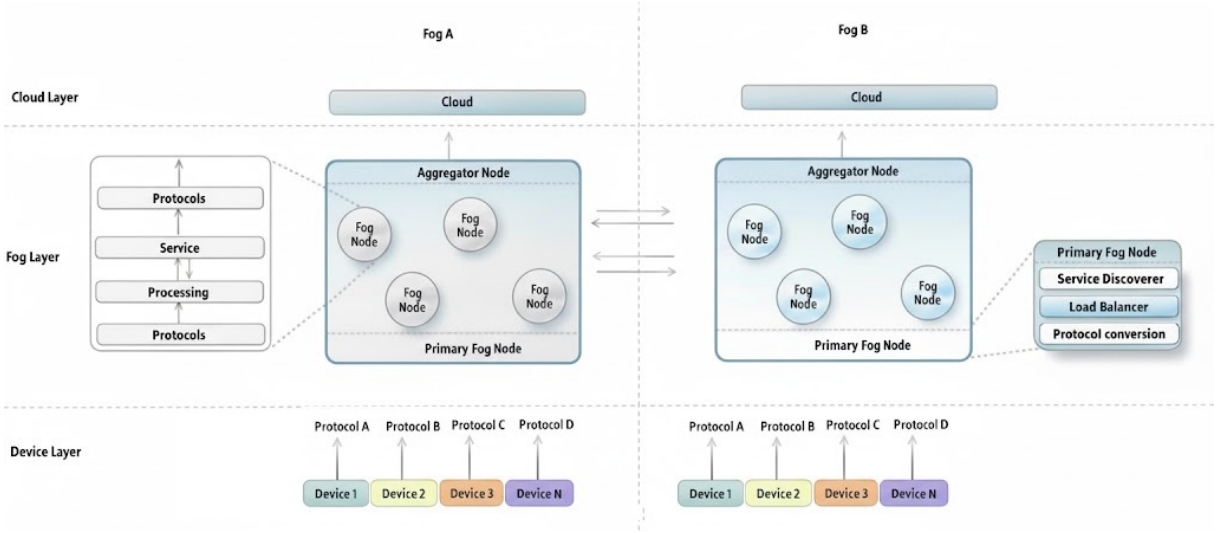
Este capítulo descreve a arquitetura modular de computação em névoa desenvolvida ao longo deste trabalho. O conteúdo está organizado para apresentar, inicialmente, a visão geral e o papel de cada tipo de nó. Em seguida, são detalhados os componentes, a organização interna dos nós de névoa, os mecanismos de comunicação entre domínios e o fluxo de dados previsto no sistema.

4.1 VISÃO GERAL

A arquitetura proposta é composta por três tipos de nós, cada um com responsabilidades distintas no fluxo de dados e na coordenação do processamento distribuído. O nó primário atua como ponto de entrada de um domínio de névoa, gerenciando dispositivos, distribuindo carga e coordenando a comunicação com outros domínios. Os nós de névoa executam o processamento local e oferecem serviços configuráveis para tratamento dos dados. O nó agregador consolida e organiza as informações processadas antes do envio à nuvem.

A Figura 2 apresenta uma visão de alto nível, mostrando o caminho percorrido pelos dados desde os dispositivos na borda até a nuvem, incluindo as interações e responsabilidades de cada tipo de nó.

Figura 2 – Visão geral da arquitetura proposta.



Fonte: Do autor.

4.2 COMPONENTES

Esta seção apresenta, em detalhe, as funções e responsabilidades de cada tipo de nó que compõe a arquitetura: nó primário, nós de névoa e nó agregador.

4.2.1 Dispositivos

Os dispositivos de borda são responsáveis por coletar dados diretamente do ambiente físico, atuando como o ponto inicial do fluxo de informações dentro da arquitetura de névoa. Eles podem ser sensores, medidores inteligentes, câmeras ou outros equipamentos capazes de capturar dados em tempo real.

Esses dispositivos podem ser fabricados por diferentes empresas e utilizar diversos protocolos de comunicação, o que torna essencial a existência de mecanismos de padronização e compatibilidade nos níveis superiores do sistema. Essa diversidade permite maior flexibilidade na integração de tecnologias, mas também exige que os nós de névoa consigam interpretar corretamente as informações recebidas, independentemente do protocolo utilizado.

Além disso, os dispositivos mantêm comunicação constante com o nó de névoa para atualizar seu estado e garantir que os dados sejam enviados corretamente. Caso percam a conexão, podem armazenar temporariamente as informações até que a comunicação seja restabelecida.

Dessa forma, os dispositivos de borda formam a base do ecossistema de computação em névoa, aproximando a coleta de dados do ambiente físico e contribuindo para a eficiência, interoperabilidade e agilidade de todo o sistema.

4.2.2 Nó Primário

O nó primário de névoa é a porta de entrada da arquitetura, ela recebe dados dos dispositivos de borda e distribui a carga de trabalho entre os nós de névoa disponíveis, direcionando as requisições conforme a especialização de cada um.

Quando um novo nó é iniciado, ele se registra no nó principal informando sua especialização, como água ou energia, e envia atualizações periódicas para indicar que continua ativo. Caso pare de enviar esses sinais, é removido automaticamente.

A distribuição da carga é balanceada, seguindo uma abordagem parecida com o Round Robin. Cada nó processa uma requisição por vez, mas sua especialização é respeitada.

Ao ser inicializado, o nó primário avisa os demais nós primários que está ativo e informa quantos nós especializados possui. Durante o funcionamento, ele monitora o número de dispositivos e requisições. Se houver desequilíbrio, verifica com outros nós primários qual tem mais capacidade disponível.

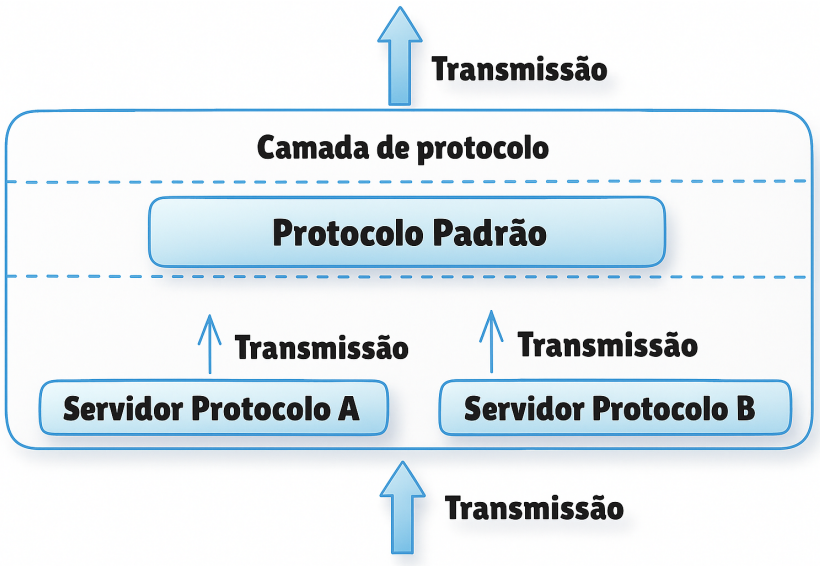
Por exemplo, se houver poucos nós especializados, o nó principal pode redirecionar requisições para outro domínio com mais recursos. Quando detecta sobrecarga, ele envia uma mensagem de broadcast para os outros nós primários pedindo informações atualizadas sobre uso. Com base nas respostas, identifica o domínio menos sobrecarregado e com menor latência, e redireciona as novas requisições para ele.

4.2.3 Nós de Névoa

Os nós de névoa realizam o processamento distribuído e foram organizados em uma estrutura interna baseada em camadas, de forma a facilitar a manutenção e permitir a adição de novas funções sem comprometer o restante do sistema.

A primeira camada, denominada protocolos, é responsável por receber as requisições provenientes do nó primário, realizar as verificações e validações necessárias e repassar as informações para a camada seguinte. Essa estrutura é ilustrada na Figura 3, onde diferentes servidores de protocolos alimentam um protocolo padrão utilizado internamente.

Figura 3 – Camada de protocolos de um nó de névoa.

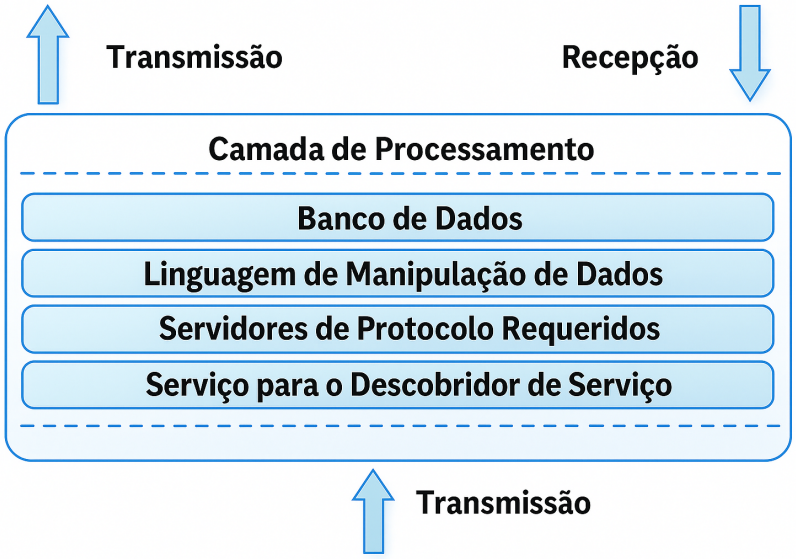


Fonte: Do autor.

A camada de processamento atua como núcleo de coordenação interna, gerenciando a comunicação entre as demais camadas e executando as necessidades específicas do nó de névoa. Essa camada também mantém um mecanismo para consultas internas de dados, permitindo que a camada de serviços acesse apenas as informações estritamente necessárias para seu funcionamento. Além disso, é por meio da camada de processamento que o nó de névoa realiza interações administrativas,

como o registro inicial no nó primário. A Figura 4 mostra os principais componentes dessa camada.

Figura 4 – Camada de processamento de um nó de névoa.



Fonte: Do autor.

Na camada de serviços é executada a aplicação que implementa a lógica de pré-processamento dos dados. Utilizando a linguagem de consulta disponibilizada pela camada de processamento, essa camada obtém os dados necessários, executa o tratamento definido pela aplicação e devolve o resultado à camada de processamento, que, por sua vez, ajusta a saída para envio pela camada de protocolos.

4.2.4 Nó Agregador

O nó agregador recebe dados pré-processados dos nós de névoa e os consolida antes de enviá-los para a nuvem. Sua função é reunir informações de vários nós, realizando conversões de formato ou protocolo quando necessário para manter a compatibilidade com o sistema em nuvem.

Ao receber os dados, o agregador combina as informações em um único arquivo estruturado, que pode incluir médias, totais ou dados agrupados, dependendo do contexto da aplicação.

Esse processo reduz o tráfego de rede e o número de transmissões para a nuvem, melhorando o desempenho, economizando largura de banda e aumentando a eficiência. Após a consolidação, os dados são enviados para a camada de nuvem para análises mais detalhadas.

4.2.5 Nuvem

A camada de nuvem atua como o ponto central de processamento e gerenciamento de dados, lidando com tarefas que exigem mais poder computacional do que os nós de névoa podem oferecer.

Com uma infraestrutura robusta, ela recebe os dados pré-processados do nó agregador e executa a etapa final de análise e consolidação. Também realiza operações complexas, como aprendizado de máquina, inteligência artificial e análises preditivas, que demandam alto desempenho e grandes volumes de dados.

Além disso, a nuvem garante armazenamento seguro e permanente, mantendo dados históricos que apoiam a tomada de decisões, auditorias e estudos avançados.

Assim, a camada de nuvem complementa a arquitetura de névoa ao oferecer capacidade de processamento e armazenamento em larga escala.

5 METODOLOGIA

Este capítulo descreve o conjunto de etapas e procedimentos adotados para o desenvolvimento da arquitetura modular de computação em névoa, desde a concepção até a validação experimental. O objetivo é apresentar, de forma sequencial e organizada, como o trabalho foi conduzido, abordando desde a análise inicial até a execução dos testes.

5.1 PLANEJAMENTO E ANÁLISE INICIAL

O trabalho foi iniciado com uma análise do estado da arte em computação em névoa, comunicação entre névoas e interoperabilidade de protocolos. Esta etapa permitiu identificar abordagens já utilizadas e lacunas existentes, servindo de base para a definição dos requisitos funcionais e estruturais da arquitetura.

5.2 DEFINIÇÃO DA ARQUITETURA

Com base nos requisitos levantados, foi elaborada a arquitetura modular, especificando os papéis de cada tipo de nó (primário, de névoa e agregador) e a organização interna dos componentes. Nesta fase, também foram definidos os mecanismos de comunicação e as interações entre nós de diferentes névoas.

5.3 IMPLEMENTAÇÃO DOS COMPONENTES

A implementação foi conduzida em etapas, de forma a permitir testes incrementais e ajustes contínuos:

1. Desenvolvimento do nó primário, com funções de registro, balanceamento e encaminhamento de requisições.
2. Implementação dos nós de névoa, estruturados em camadas para suportar diferentes protocolos, processamento interno e execução de serviços.
3. Construção do nó agregador, responsável pela consolidação e envio de dados à nuvem.

5.4 PROCEDIMENTOS DE TESTE

Após a implementação, serão realizados testes com foco em:

- **Interoperabilidade**: verificar a capacidade da arquitetura de lidar com diferentes protocolos de comunicação.

- **Balanceamento de carga**: avaliar o comportamento do nó primário na distribuição das requisições entre os nós de névoa, incluindo cenários de encaminhamento para outras névoas.
- **Agregação de dados**: analisar a consolidação realizada pelo nó agregador e seu impacto na redução de requisições à nuvem.

6 TESTES

Este capítulo descreve os procedimentos de teste adotados para avaliar a arquitetura proposta, abrangendo ambiente, cenários, métricas, coleta e análise dos dados.

6.1 CONTEXTUALIZAÇÃO DO PROBLEMA

Para avaliar a arquitetura proposta, foi adotado um cenário inspirado em aplicações de cidades inteligentes, com foco na gestão de recursos de água e energia. Nesse contexto, foram considerados medidores inteligentes que enviam dados de consumo para uma camada de névoa, onde ocorre o pré-processamento das informações antes de seu encaminhamento à nuvem.

Os dados utilizados para simulação foram obtidos a partir de conjuntos públicos, sendo um relacionado ao consumo residencial de energia elétrica (Hernández et al., s.d.) e outro referente a medições de consumo de água (Hellenic Data Service, s.d.).

Antes da utilização nos testes, ambos os conjuntos de dados passaram por um processo de pré-processamento que incluiu a remoção de informações não essenciais e a agregação das medições em intervalos horários. Esse tratamento visou manter apenas os elementos relevantes para simular a operação contínua dos dispositivos, aproximando o comportamento dos dados ao fluxo esperado em um ambiente real de monitoramento urbano.

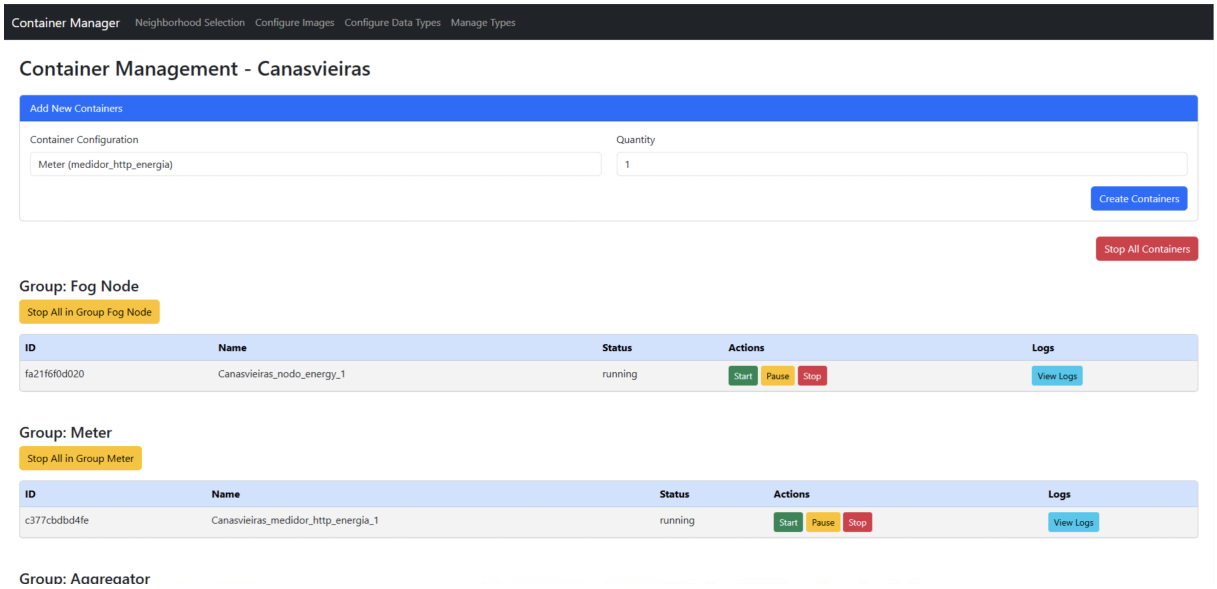
6.2 AMBIENTE DE TESTES

O ambiente de testes foi configurado para representar um cenário de cidade inteligente, no qual cada domínio de névoa corresponde a um bairro distinto. Cada névoa pode conter um nó primário, um nó agregador e um número variável de nós de névoa e dispositivos, de acordo com a demanda e o tipo de aplicação simulada.

A implantação e o gerenciamento desses componentes foram realizados por meio de uma interface web desenvolvida para este trabalho, que interage diretamente com o Docker para criar e controlar as instâncias. Essa interface permite selecionar o bairro (névoa) no qual os componentes serão implantados, definir o tipo de contêiner e a quantidade de instâncias, além de oferecer controle individual ou em grupo para iniciar, pausar, parar e visualizar logs.

A Figura 5 apresenta a tela de gerenciamento de contêineres para o bairro Canasvieiras. Nela, observa-se a criação de um nó de névoa e de um medidor de energia, ambos em execução. O painel agrupa as instâncias por tipo de componente, permitindo visualizar rapidamente seu estado e realizar ações sobre cada grupo ou instância individual.

Figura 5 – Interface para gerenciamento de contêineres no ambiente de testes.



Fonte: Do autor.

6.2.1 Tecnologias e Ambiente Físico

No ambiente de testes, todos os componentes da arquitetura foram implementados em *Python*, com exceção do medidor compatível com o protocolo *CoAP*, que foi desenvolvido em *Node.js* devido à disponibilidade de bibliotecas específicas e otimização do fluxo de comunicação nesse protocolo.

Os experimentos foram executados em uma única máquina física, utilizada para hospedar os contêineres Docker que compõem cada instância de nó e dispositivo. A Tabela 2 apresenta as especificações do equipamento utilizado.

Tabela 2 – Especificações da máquina utilizada nos testes.

Componente	**Especificação**
Processador	Intel Core i7-7700K @ 4.20GHz
Memória RAM	32 GB DDR4
Armazenamento	SSD 2 TB
Sistema Operacional	Windows 11
Docker	v4.38.0

Fonte: Do autor.

6.3 PROTOCOLOS

No ambiente de testes, o nó primário é responsável por receber dados utilizando diferentes protocolos, incluindo *CoAP*, *MQTT* e *HTTP*. Independentemente do

protocolo de origem, as mensagens são convertidas para *HTTP* antes de serem encaminhadas aos nós de névoa, de forma a padronizar o tráfego interno.

Dentro de um nó de névoa, a comunicação entre a camada de protocolos e a camada de processamento ocorre por meio de *WebSocket*, mantendo a conexão aberta para evitar o custo de abertura e fechamento contínuo de conexões. Essa abordagem reduz a sobrecarga e melhora o tempo de resposta.

A partir do nó de névoa, os dados já passam a ser organizados em estrutura *CSV*, formada a partir das informações recebidas em *JSON*. Essa etapa permite preparar os dados para compatibilidade com o formato aceito pela nuvem. A transmissão do nó de névoa para o nó agregador é realizada via *SFTP*.

No nó agregador, os arquivos *CSV* recebidos de diferentes nós de névoa são consolidados em um único arquivo antes de serem enviados para a camada de nuvem, também via *SFTP*, para a zona de entrada (*landing zone*) do *HPCC Systems*.

6.4 DISTRIBUIÇÃO ENTRE NÉVOAS E BALANCEAMENTO DE CARGA

Nos experimentos, os 120 dispositivos foram distribuídos entre dois domínios de névoa distintos, representando diferentes regiões no cenário de cidade inteligente. A distribuição inicial foi a seguinte:

- **Névoa A**: 70 dispositivos (12 medidores de energia via *HTTP*, 12 via *CoAP*, 12 via *MQTT*, 12 medidores de água via *HTTP*, 12 via *CoAP*, 10 via *MQTT*).
- **Névoa B**: 50 dispositivos (8 medidores de energia via *HTTP*, 8 via *CoAP*, 8 via *MQTT*, 8 medidores de água via *HTTP*, 8 via *CoAP*, 10 via *MQTT*).

Essa distribuição propositalmente desigual permitiu avaliar o comportamento do sistema sob condições de carga diferentes entre os domínios. Quando uma das névoas atingiu maior taxa de utilização, o mecanismo de balanceamento foi acionado: o nó primário da névoa mais carregada redirecionou parte das requisições para a outra, realizando a conversão para *HTTP* antes do envio, independentemente do protocolo de origem (*HTTP*, *CoAP* ou *MQTT*).

Nos testes, foram simulados dois dias de operação (48 horas), com cada dispositivo enviando leituras periódicas agregadas em janela horária de consumo.

No nó agregador, as medições de cada domínio de névoa foram consolidadas em um único arquivo por dia e transmitidas para o *HPCC Systems* ao final de cada dia, sem que nenhum contêiner apresentasse falhas durante a execução.

6.5 GRAPHQL

Nos nós de névoa, a comunicação entre a camada de serviços e a camada de processamento é realizada utilizando *GraphQL* como linguagem de consulta de

dados. Essa abordagem permite que a aplicação especifique exatamente quais campos e registros deseja receber, evitando transferências desnecessárias e reduzindo a sobrecarga de processamento.

O *GraphQL* possibilita ainda a consulta a múltiplas fontes de dados de forma unificada, agregando resultados distintos em uma única resposta. Nos testes realizados, o armazenamento local temporário nos nós de névoa foi implementado com *MongoDB*, escolhido por sua flexibilidade para lidar com dados em formato `JSON` e pela capacidade de inserir registros heterogêneos sem a exigência de um esquema fixo.

Com essa integração, a camada de serviços consegue solicitar à camada de processamento apenas as informações estritamente necessárias para o pré-processamento, recebendo uma resposta já filtrada e adaptada ao contexto da aplicação. Isso mantém a arquitetura modular, otimiza o fluxo de dados interno e garante maior agilidade na execução das tarefas.

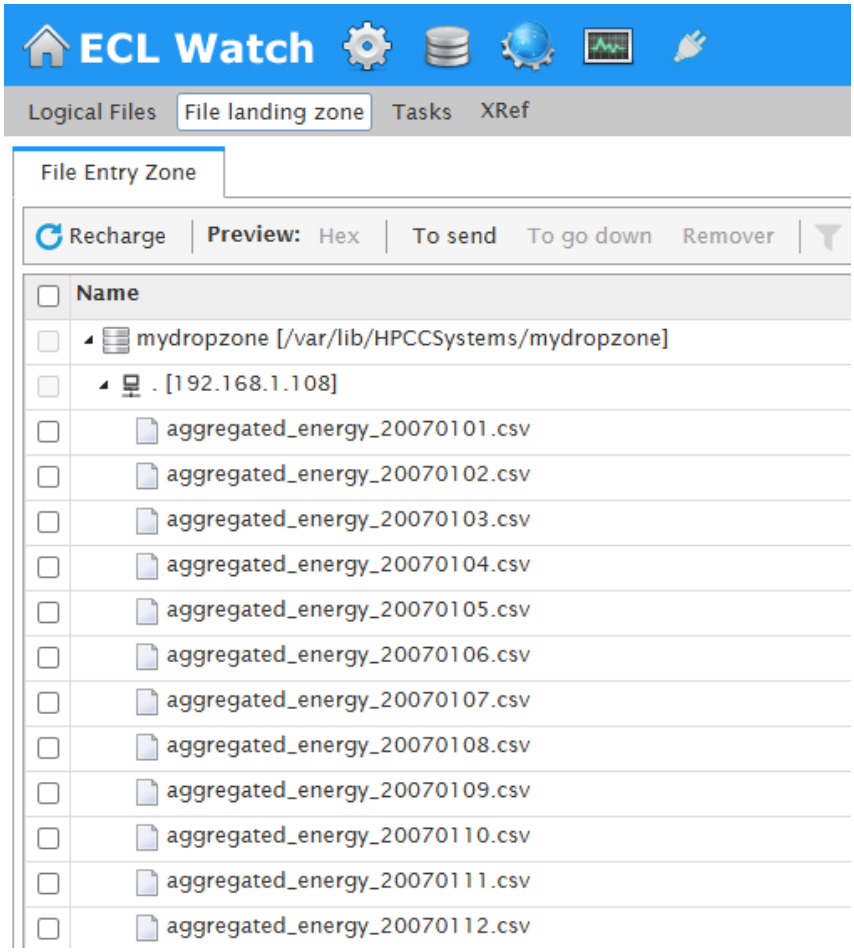
6.6 HPCC SYSTEMS NA CAMADA DE NUVEM

Na fase de testes, os dados consolidados pelos nós agregadores foram enviados para a camada de nuvem, implementada com o *HPCC Systems*. Os arquivos recebidos, no formato `.csv` suportado pela plataforma, resultam da união dos pré-processamentos realizados nos diferentes nós de névoa, reunidos pelo nó agregador antes da transmissão.

Esses arquivos foram disponibilizados na zona de entrada da plataforma (*landing zone*), onde ficaram prontos para processamento e consultas.

A Figura 6 apresenta o ambiente *ECL Watch*, mostrando os arquivos agregados de consumo de energia, já no formato consolidado por data, prontos para serem tratados ou analisados conforme as necessidades da aplicação.

Figura 6 – Arquivos consolidados na *landing zone* do HPCC Systems.



Fonte: Do autor.

7 RESULTADOS

Esta seção apresenta as métricas obtidas nos testes da arquitetura proposta. Para cada métrica, é mostrado o cálculo realizado com base nos valores coletados e, em seguida, um parágrafo explicando seu significado e comentando o valor obtido.

7.1 VALORES MEDIDOS

Os valores apresentados nas Tabelas 3 e 4 foram obtidos a partir de *timestamps* registrados nos pontos de entrada e saída de cada componente, bem como de medições de consumo de memória realizadas durante o envio de mensagens.

A Tabela 3 mostra as latências médias unidirecionais para cada segmento do fluxo, abrangendo desde a recepção inicial no *Load Balancer* até o envio final ao HPCC Systems, enquanto a Tabela 4 apresenta o consumo médio de memória por componente.

Observa-se que o maior tempo de transmissão ocorre no trajeto entre o Agregador e o HPCC Systems, indicando que esta etapa concentra a maior parte da latência no fluxo de dados. Em relação ao uso de memória, verifica-se que os nós de névoa demandam mais recursos, devido às tarefas de pré-processamento e armazenamento temporário, enquanto os componentes de controle, como o *Load Balancer* e o Agregador, apresentam consumo significativamente menor.

Tabela 3 – Latências médias unidirecionais medidas

Caminho	**Latência (ms)**
Medidor $\rightarrow$ Load Balancer	4,6
Load Balancer $\rightarrow$ Nó de Névoa	2,0
Nó de Névoa $\rightarrow$ Agregador	2,4
Agregador $\rightarrow$ HPCC Systems	6,6

Fonte: Do autor.

Tabela 4 – Consumo médio de memória por componente

Componente	**Memória (MiB)**
Load Balancer	37,37
Agregador	40,14
Nó de Névoa (Energia)	276,60
Simulador de Medidor HTTP	19,97

Fonte: Do autor.

7.2 LATÊNCIA PONTA A PONTA

$$L_t = L_{ml} + L_{ln} + L_{na} + L_{ah}$$

Onde:

- L_{ml} = latência do Medidor $\rightarrow$ Load Balancer = 4,6 ms
- L_{ln} = latência do Load Balancer $\rightarrow$ Nó de Névoa = 2,0 ms
- L_{na} = latência do Nó de Névoa $\rightarrow$ Agregador = 2,4 ms
- L_{ah} = latência do Agregador $\rightarrow$ HPCC Systems = 6,6 ms

$$L_t = 4{,}6 + 2{,}0 + 2{,}4 + 6{,}6 = 15{,}6 \text{ ms}$$

O cálculo considera a soma das latências médias unidirecionais medidas para cada segmento do fluxo de transmissão.

7.3 THROUGHPUT

$$\text{Throughput} = \frac{M_p}{T}$$

Onde:

- N_d = número de dispositivos = 120
- F_m = frequência de envio de mensagens (mensagens por hora) = 1
- T = duração total do experimento = 48 h
- M_p = total de mensagens processadas = 5,760

$$\text{Throughput} = \frac{5{,}760}{48} = 120 \text{ msgs/h}$$

O cálculo considera a quantidade de mensagens processadas dividida pelo tempo total do experimento, resultando no número médio de mensagens processadas por hora.

7.4 TAXA DE ENTREGA

$$M_e = N_d \times F_m \times T = 120 \times 1 \times 48 = 5{,}760$$

$$\text{Taxa de entrega} = \frac{M_r}{M_e} \times 100$$

Onde:

- M_r = quantidade total de mensagens recebidas = 5,760

- M_e = quantidade total de mensagens esperadas = 5,760

$$\text{Taxa de entrega} = \frac{5{,}760}{5{,}760} \times 100 = 100\%$$

O cálculo considera a razão entre a quantidade de mensagens recebidas e o total esperado no período do experimento.

7.5 FATOR DE AGREGAÇÃO

$$\rho = \frac{M_r}{M_n}$$

Onde:

- M_r = mensagens recebidas pelo agregador em dois dias = $2 \times 2880 = 5760$
- M_n = mensagens enviadas à nuvem pelo agregador no mesmo período = 2

$$\rho = \frac{5760}{2} = 2880$$

O cálculo considera a razão entre a quantidade de mensagens recebidas pelo agregador e o número de mensagens enviadas à nuvem em um mesmo intervalo de tempo.

7.6 REDUÇÃO DE REQUISIÇÕES À NUVEM

$$\text{Redução}(\%) = \frac{(N_d \times M_d \times 2) - M_a}{N_d \times M_d \times 2} \times 100$$

Onde:

- N_d = número de dispositivos = 120
- M_d = mensagens enviadas por dispositivo/dia = 24
- M_a = mensagens enviadas via agregador no total de dois dias = 2

$$\text{Redução}(\%) = \frac{(120 \times 24 \times 2) - 2}{120 \times 24 \times 2} \times 100 = \frac{5760 - 2}{5760} \times 100 \approx 99{,}97\%$$

O cálculo considera a quantidade total de mensagens enviadas diretamente pelos dispositivos no período de dois dias, comparada ao número de mensagens transmitidas via agregador no mesmo período.

8 CONCLUSÃO

Este trabalho apresentou o desenvolvimento e a avaliação de uma arquitetura modular para ambientes de computação em névoa, com suporte à comunicação direta entre domínios distintos e integração entre dispositivos de borda, nós de névoa e a nuvem, tomando como referência o modelo proposto pelo OpenFog Consortium (IEEE. . . , 2018). A solução foi projetada considerando os principais pilares dessa arquitetura no contexto de cidades inteligentes, buscando atender às diretrizes de escalabilidade, abertura, autonomia, programabilidade e hierarquia.

A escalabilidade foi contemplada na forma como a infraestrutura pode ser expandida sem reconfigurações complexas, permitindo a adição de novos nós ou dispositivos de maneira transparente. A abertura esteve presente no suporte a diferentes protocolos de comunicação, favorecendo a interoperabilidade entre sistemas heterogêneos. A autonomia foi explorada por meio do balanceamento de carga dinâmico e distribuído entre domínios, dispensando a necessidade de um ponto central de orquestração. A programabilidade se refletiu na flexibilidade para ajustar o comportamento do balanceador e dos nós de névoa de acordo com as necessidades do cenário de uso. Já a hierarquia esteve evidente na organização em camadas, interligando dispositivos de borda, névoa e nuvem de forma estruturada e modular.

Os resultados obtidos indicam que a combinação de abstração de protocolos, balanceamento dinâmico de carga e alto fator de agregação contribuiu para reduzir o volume de requisições à nuvem e melhorar a utilização dos recursos, mantendo tempos de resposta compatíveis com aplicações sensíveis à latência.

Na comparação com trabalhos como o EXEGESIS (**exegesis2021**), que permite a substituição de serviços em ambiente de execução por meio da reconstrução do contêiner, a arquitetura proposta requer a interrupção para a troca de serviços. No entanto, a separação entre a camada de aplicação e as demais possibilita interromper apenas a aplicação, realizar os ajustes necessários e retomá-la sem modificar a lógica de comunicação. Durante esse processo, a camada de protocolos permanece ativa, continuando a receber dados, processá-los e armazená-los temporariamente, o que contribui para reduzir perdas e facilitar a retomada do serviço. Essa característica pode simplificar a manutenção e a adaptação a novos requisitos, minimizando o impacto de mudanças, ainda que a substituição implique uma breve indisponibilidade. Além disso, o encaminhamento distribuído entre névoas permite o tráfego direto de dados entre domínios, evitando a necessidade de um ponto único de orquestração.

Como contribuição, o trabalho demonstrou a viabilidade de integrar múltiplos protocolos e distribuir cargas entre diferentes domínios de névoa sem comprometer o desempenho. Para trabalhos futuros, sugere-se a adoção de mecanismos de autenticação e criptografia ponta a ponta, a implementação de estratégias de recuperação

automática em caso de falhas e a aplicação da arquitetura em outros domínios, como saúde conectada e transporte inteligente, considerando diferentes perfis de tráfego e demanda.

REFERÊNCIAS

BARRETT, Daniel J.; SILVERMAN, Richard E.; BYRNES, Robert G. **SSH, The Secure Shell: The Definitive Guide**. 2. ed. Sebastopol, CA: O'Reilly Media, 2005. ISBN 978-0596008956.

BORMANN, Carsten; CASTELLANI, Angelo P.; SHELBY, Zach. CoAP: An Application Protocol for Billions of Tiny Internet Nodes. **IEEE Internet Computing**, v. 16, n. 2, p. 62–67, mar. 2012.

BOUDIEB, A.; AYED, M.; HAROUS, S. Microservice Instances Selection and Load Balancing in Fog Computing. **Future Generation Computer Systems**, v. 154, p. 510–523, 2024.

CASSEL, Gustavo André Setti; ROSA RIGHI, Rodrigo da; COSTA, Cristiano André da; BEZ, Marta Rosecler; PASIN, Marcelo. Towards providing a priority-based vital sign offloading in healthcare with serverless computing and a fog-cloud architecture. **Future Generation Computer Systems**, v. 157, p. 51–66, 2024.

D'AGOSTINO, R.; LONGO, M.; PALMIERI, G. A Scalable Fog Computing Solution for Industrial Predictive Maintenance. **Electronics**, v. 14, n. 2, p. 1–18, 2025.

GOURLEY, David; TOTTY, Brian; SAYER, Marjorie; REDDY, Sailu; AGGARWAL, Anshu. **HTTP: The Definitive Guide**. Sebastopol, CA: O'Reilly Media, 2002. ISBN 978-1565925090.

HELLENIC DATA SERVICE. **Water Consumption Dataset**. [S.l.*:* s.n.]. `https://data.hellenicdataservice.gr/dataset/78776f38-a58b-4a2a-a8f9-85b964fe5c95`. Acesso em: 11 ago. 2025.

HERNÁNDEZ, Héctor et al. **Individual household electric power consumption Data Set**. [S.l.*:* s.n.]. `https://archive.ics.uci.edu/dataset/235/individual+household+electric+power+consumption`. Acesso em: 11 ago. 2025.

IEEE Standard for Adoption of OpenFog Reference Architecture for Fog Computing. [S.l.*:* s.n.], 2018. p. 1–176.

MARKAKIS, E. K.; SIDERIS, A.; MASTORAKIS, G.; MAVROMOUSTAKIS, C. X.; PALLIS, E.; CHARALABIDIS, Y. EXEGESIS: Extreme Edge Resource Harvesting for a Virtualized Fog Environment. **IEEE Communications Magazine**, v. 55, n. 7, p. 173–179, jul. 2017.

OLIVEIRA, J.; SANTOS, A.; MENDES, L. Enhancing Modular Application Placement in Hierarchical Fog Computing. **Computer Communications**, v. 222, p. 180–194, 2024.

RAHMAN, M.; HUSSAIN, M. Testbed Analysis of Multi-Fog Architecture for Interoperable IoT. **Wireless Networks**, v. 30, p. 451–463, 2024.

RAHMANI, Amir M.; GIA, Tuan Nguyen; NEGASH, Behailu; ANZANPOUR, Ali; AZIMI, Iman; LAGERSPETZ, Mikael; LILJEBERG, Pasi. Exploiting smart e-Health gateways at the edge of healthcare Internet-of-Things: A fog computing approach. **Future Generation Computer Systems**, v. 78, p. 641–658, 2018.

SILVEIRA, Rubens Prates da. **GraphQL**: A nova alternativa para criação de APIs. São Paulo: Casa do Código, 2019. ISBN 978-6586057956.

TANENBAUM, Andrew S.; BOS, Herbert. **Modern Operating Systems**. 4. ed. Boston: Pearson, 2015. ISBN 978-0133591620.

TANENBAUM, Andrew S.; WETHERALL, David J. **Computer Networks**. 5. ed. Boston: Pearson, 2011. ISBN 978-0132126953.

TAYLOR, Richard. **Mastering HPCC Systems: Platform Overview and History**. [S.l.]: Function, 2022. Kindle Edition.

WANG, Quan; DING, Weiping; XU, Xiaolong; WU, Fei; JIA, Weijia. WebSocket-based Real-time Communication for Industrial Automation. In*:* PROCEEDINGS of the 2013 IEEE International Conference on Industrial Technology (ICIT). [S.l.*:* s.n.], 2013. p. 1080–1085.