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

No resumo são ressaltados o objetivo da pesquisa, o método utilizado, as discussões e os resultados com destaque apenas para os pontos principais. O resumo deve ser significativo, composto de uma sequência de frases concisas, afirmativas, e não de uma enumeração de tópicos. Não deve conter citações. Deve usar o verbo na voz ativa e na terceira pessoa do singular. O texto do resumo deve ser digitado, em um único bloco, sem espaço de parágrafo. O espaçamento entre linhas é simples e o tamanho da fonte é 12. Abaixo do resumo, informar as palavras-chave (palavras ou expressões significativas retiradas do texto) ou, termos retirados de thesaurus da área. Deve conter de 150 a 500 palavras. O resumo é elaborado de acordo com a NBR 6028.

Palavras-chave: Palavra-chave 1. Palavra-chave 2. Palavra-chave 3.

ABSTRACT

Resumo traduzido para outros idiomas, neste caso, inglês. Segue o formato do resumo feito na língua vernácula. As palavras-chave traduzidas, versão em língua estrangeira, são colocadas abaixo do texto precedidas pela expressão “Keywords”, separadas por ponto.

Keywords: Keyword 1. Keyword 2. Keyword 3.

LISTA DE FIGURAS

LISTA DE QUADROS

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

LISTA DE SÍMBOLOS

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.

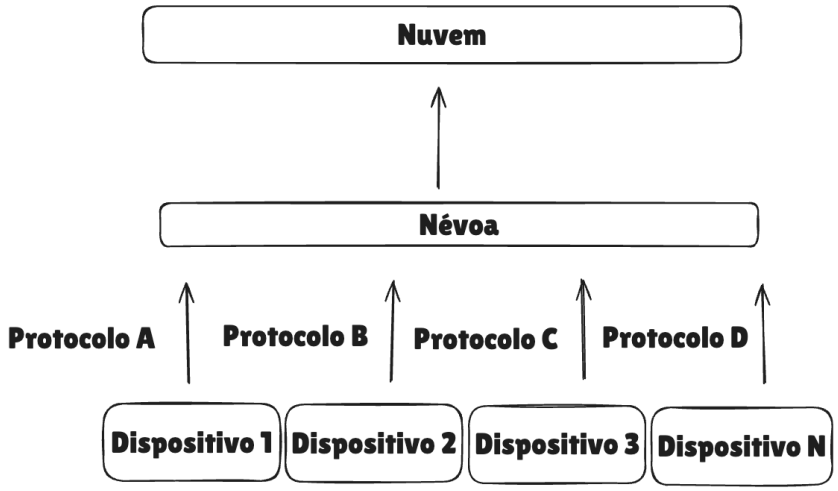
Na Figura 1, temos ilustrado uma arquitetura básica de névoa, os dispositivos enviam dados para os equipamentos da camada de névoa, onde ocorre o processamento local. Os dados processados podem então ser devolvidos para os dispositivos ou encaminhados à nuvem, dependendo do contexto da aplicação.

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.

Para abordar esses desafios, este trabalho propõe uma arquitetura de névoa modular com suporte à comunicação entre diferentes névoas, a qual é composta por três camadas principais, descritas a seguir:

- **Nó de Névoa Primário:** responsável por atuar como ponto de entrada da arquitetura de névoa. Ele realiza a conversão de protocolos de comunicação, o balanceamento de carga entre os demais nós de névoa e outros nós

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



Fonte: Do autor.

primários, além de localizar e coordenar os nós de névoa disponíveis.

- **Nó de Névoa:** recebe os dados encaminhados pelo nó primário, realiza o pré-processamento local conforme a lógica da aplicação, e transmite os resultados ao nó agregador.
- **Nó Agregador:** responsável por consolidar os dados processados pelos diversos nós de névoa em um único arquivo ou estrutura de dados, facilitando sua transmissão para a nuvem.

1.1 OBJETIVOS

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

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 esse trabalho, foram realizadas pesquisas nas bases de dados de artigos científicos, utilizando as seguintes palavras-chaves: 'fog', 'fog' AND 'architecture', e aplicado um filtro para os últimos 5 anos, ou seja, de 2020 até 2024, e obtivemos a quantidade de artigos que podemos visualizar na Tabela 1.

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

Palavra-chave	**Banco de Dados**	**Quantidade de Artigos**
'fog'	Scopus	18.970
	IEEE	6.925
'fog' AND 'architecture'	Scopus	3.144
	IEEE	1.749

Fonte: Do autor.

Com base nos dados da tabela fornecida, observamos que o Scopus lidera com a maior coleção de artigos indexados. Em contraste, o IEEE apresenta uma quantidade relativamente menor.

3.1 TRABALHOS CORRELATOS

Foram lidos títulos e resumo de 125 artigos, dos quais 25 foram lidos completamente, e desses foram selecionados 5 que apresentam propostas semelhantes à proposta deste trabalho e por isso serão detalhados nas subseções seguintes.

3.1.1 Inter-container Communication Aware Container Placement in Fog Computing

Neste trabalho, os autores propõem um algoritmo genético para a colocação de contêineres, destacando a abordagem via RDMA (Remote Direct Memory Access) por sua capacidade de reduzir significativamente a latência de rede e a utilização da CPU do host. O RDMA permite acesso direto à memória de outro host sem envolvimento do sistema operacional, resultando em uma comunicação mais rápida e eficiente. Comparado com os modos Host e Overlay, a comunicação RDMA oferece vantagens significativas em termos de desempenho e redução de latência. Além disso, o artigo aborda a importância do isolamento adequado entre contêineres para garantir a estabilidade e segurança do sistema, aspectos críticos em ambientes de computação em névoa onde múltiplos contêineres compartilham os mesmos recursos físicos.

Os resultados das simulações mostram que a inclusão de RDMA pode melhorar substancialmente o desempenho das aplicações, destacando a importância de selecionar a tecnologia de comunicação inter-contêiner mais adequada para cada situação.

Cada contêiner pode utilizar o protocolo de comunicação que melhor se adapta às suas necessidades específicas, proporcionando maior eficiência e flexibilidade na comunicação entre contêineres. Isso permite que os sistemas de computação em névoa mantenham a eficiência e a performance ao lidar com a comunicação entre diversos protocolos, assegurando que cada contêiner opere com o protocolo mais apropriado para suas funções.

3.1.2 EXEGESIS: Extreme Edge Resource Harvesting for a Virtualized Fog Environment

Neste trabalho, os autores propõem a arquitetura EXEGESIS, que aborda os desafios de comunicação e processamento de dados na névoa. A arquitetura de três camadas inclui a camada "mist", composta por dispositivos interconectados que formam vizinhanças locais; a camada "vFog", que permite interconexões dinâmicas entre essas vizinhanças; e a camada de nuvem, que oferece recursos abundantes e facilita a interconexão dos elementos vFog. Esta estrutura modular e hierárquica visa otimizar a utilização de recursos, reduzir a latência e melhorar a eficiência do sistema, permitindo a conversão e integração de protocolos diversos.

No contexto da comunicação com diversos protocolos, EXEGESIS propõe uma plataforma de orquestração que facilita a interoperabilidade entre dispositivos utilizando diferentes padrões e tecnologias. A camada "vFog"age como um intermediário virtual, coordenando os recursos e a comunicação entre as névoas e a nuvem, garantindo que dados críticos sejam processados e armazenados onde podem agregar mais valor.

3.1.3 Extending Scalability of IoT/M2M Platforms with Fog Computing

Neste trabalho, os autores propõem a migração do oneM2M, uma plataforma global de IoT/M2M, para uma arquitetura de névoa baseada em contêineres hierárquicos. Esta abordagem visa resolver os problemas de escalabilidade e latência ao trazer a capacidade de processamento e armazenamento para mais perto dos dispositivos finais.

Para garantir a comunicação eficiente entre os diversos componentes da arquitetura, o oneM2M suporta a ligação de suas interfaces de comunicação aos protocolos HTTP e CoAP, fornecendo APIs Restful sobre todas essas interfaces. Isso permite que diferentes tipos de dispositivos e serviços se comuniquem de maneira eficaz, utilizando o protocolo mais adequado para cada caso. Por exemplo, HTTP pode ser usado para comunicação de dados mais robusta e complexa, enquanto o CoAP é ideal para dispositivos com restrições de recursos e necessidade de comunicação leve.

3.1.4 Privacy preserving data aggregation with fault tolerance in fog-enabled smart grids

Neste trabalho, os autores propõem um esquema de agregação de dados seguro e tolerante a falhas para grades inteligentes (SG) habilitadas por computação em névoa. O estudo aborda os desafios de privacidade e segurança em SG, onde os medidores inteligentes (SM) coletam dados de consumo de eletricidade dos usuários e os transmitem para os provedores. Para garantir a privacidade, o esquema utiliza o criptossistema Boneh-Goh-Nissam (BGN) para criptografar os dados de medição e o algoritmo de assinatura digital de curva elíptica (ECDSA) para autenticação da fonte. A tolerância a falhas é assegurada ao permitir que os nós de névoa (FN) substituam os dados dos medidores com falha pelos últimos dados válidos armazenados, sem a necessidade de comunicação adicional com autoridades confiáveis.

O estudo também destaca a importância de proteger os dados dos consumidores em grades inteligentes, onde a agregação segura de dados é crucial para evitar a divulgação de informações sensíveis. A arquitetura baseada em névoa permite a agregação de dados próximo aos usuários finais, reduzindo a latência e os custos de comunicação. A utilização de criptografia homomórfica permite operações sobre dados criptografados, mantendo a privacidade dos usuários. A abordagem adotada assegura que, mesmo em presença de medidores defeituosos, a agregação de dados continua eficaz e eficiente, fornecendo resultados estimativos precisos para a gestão de eletricidade nas grades inteligentes.

3.1.5 FPDA: Fault-Tolerant and Privacy-Enhanced Data Aggregation Scheme in Fog-Assisted Smart Grid

Neste trabalho, os autores propõem a agregação de dados em redes inteligentes (SG) com foco na preservação da privacidade e tolerância a falhas. O FPDA propõe um método de agregação de dados que permite a disponibilidade de dados e a preservação da privacidade simultaneamente. A tolerância a falhas garante que a descriptografia possa ser realizada com sucesso mesmo que alguns medidores inteligentes (SMs) falhem, sem a necessidade de uma autoridade confiável centralizada ou atualizações de chaves após cada recuperação de falha.

A abordagem FPDA utiliza uma criptografia homomórfica aditiva, onde a descriptografia do texto cifrado agregado é equivalente à soma direta de todas as leituras em texto claro. O esquema é projetado para operar eficientemente em uma arquitetura assistida por névoa (fog), onde os nós de névoa (FNs) realizam a agregação de dados antes de transmiti-los ao centro de controle (CC). Quando alguns SMs falham, os FNs iniciam interações adicionais de solicitação-resposta com um número limitado de SMs para reconstruir as chaves parciais necessárias. Esse método, baseado em um

esquema de compartilhamento de segredo de limite (tSSS) estendido, permite que os SMs reconstruam segredos subsequentes sem revelar suas ações originais, mantendo assim a privacidade e a segurança.

3.2 COMPARATIVO COM TRABALHOS CORRELATOS

A tabela 2 apresenta um comparativo entre as propostas apresentadas na literatura e a proposta deste trabalho, destacando as características atendidas por cada uma. A proposta deste trabalho se diferencia das demais por algumas razões. Primeiramente, a arquitetura modular proposta para cada nó da névoa inclui múltiplas camadas, cada uma com funções específicas que facilitam a manutenção e a flexibilidade do sistema. Isso contrasta com outras propostas que geralmente utilizam uma aplicação monolítica ou o uso de containers. A abordagem monolítica, embora simplificada em termos de implantação inicial, apresenta desafios em termos de manutenção e escalabilidade. Já o uso de containers, apesar de proporcionar uma certa modularidade, pode enfrentar problemas relacionados ao empacotamento e à sobrecarga de gerenciamento de múltiplos containers.

Tabela 2 – Características atendidas por proposta

Característica	**Propostas que atendem**
Uso de RDMA	Proposta 1
Uso de Contêineres	Proposta 1; Proposta 2; Proposta 3
Arquitetura Modular	Este Trabalho
Suporte a Múltiplos Protocolos	Proposta 1; Proposta 2; Proposta 3; Este Trabalho
Camada de Serviço Flexível	Este Trabalho
Comunicação entre Névoas	Este Trabalho
Criptografia Homomórfica	Proposta 4
A aplicação no nó é monolítica	Proposta 4; Proposta 5

Fonte: Do autor.

Na arquitetura proposta nesse trabalho, o fluxo de dados inicia na borda, onde o nó primário da névoa atua como ponto de entrada, registrando os medidores conectados, realizando o balanceamento de carga e, quando necessário, redirecionando solicitações excedentes para outros domínios de névoa que disponham de maior capacidade de processamento.

Cada nó da névoa é estruturado em três camadas. A camada de protocolos é responsável por receber e transmitir dados em diferentes formatos. A camada de processamento gerencia os elementos essenciais para a operação do nó, realizando armazenamento temporário e interações internas necessárias para manter o funci-

onamento contínuo. Já a camada de serviço executa aplicações configuráveis que processam os dados conforme a lógica da aplicação, preparando-os para as próximas etapas.

Após o processamento nos nós especializados, as informações seguem para um nó agregador, que consolida dados provenientes de múltiplas névoas em arquivos estruturados, otimizando o tráfego antes do envio à nuvem e preparando o material para análises em larga escala.

Conforme apresentado anteriormente, essa proposta apresenta alguns diferenciais em relação aos trabalhos correlatos. A arquitetura proposta para a comunicação entre névoas será explicada em detalhes a seguir.

4 PROPOSTA

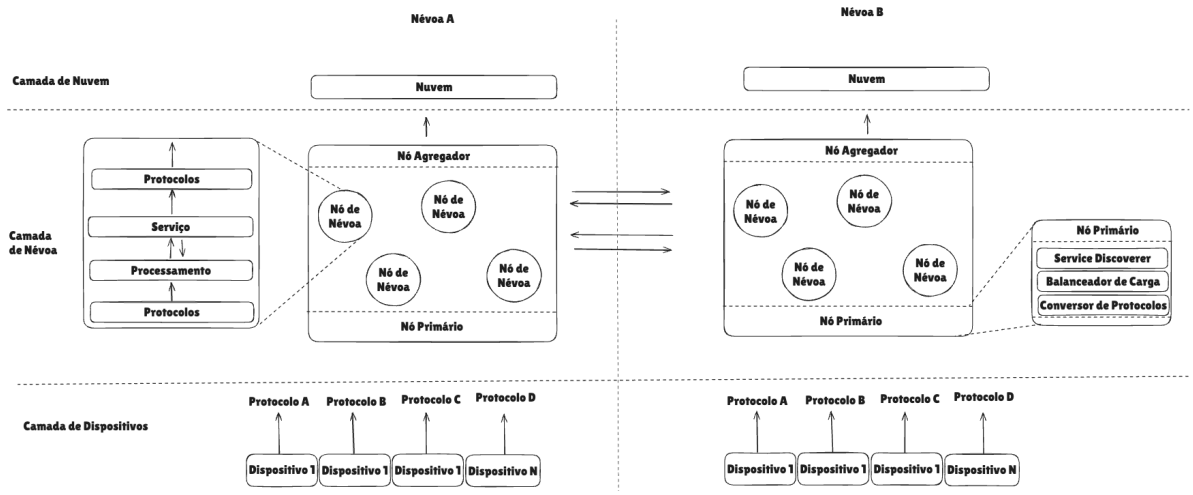
Este capítulo descreve a arquitetura modular de computação em névoa desenvolvida ao longo deste trabalho. O conteúdo está organizado para apresentar, inicialmente, a visão geral e o papel de cada tipo de nó. Em seguida, são detalhados os componentes, a organização interna dos nós de névoa, os mecanismos 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 Nó Primário

O nó primário funciona como o ponto inicial de contato para dispositivos e novos nós que ingressam em uma névoa. Assim que um nó de névoa é iniciado, ele envia ao nó primário um conjunto de atributos que descrevem sua capacidade, especializações e estado atual de operação. Essas informações são registradas e mantidas para permitir a localização de recursos disponíveis na própria névoa.

Ao receber uma requisição, o nó primário identifica os atributos necessários para o processamento e verifica, entre os nós registrados, quais correspondem ao perfil solicitado. O balanceamento entre os nós locais segue um padrão que procura alternar os destinos de forma equilibrada, em comportamento semelhante a um esquema round robin, sempre respeitando as características solicitadas por cada carga.

Se a análise indicar que não haverá nós suficientes na névoa local para atender à demanda, o nó primário envia uma solicitação para que todos os nós primários de outras névoas informem o estado atual de seus recursos. Esse processo lembra o funcionamento de um protocolo de fofoca, mas é otimizado para reduzir o tráfego de rede, mantendo apenas as trocas essenciais. Com base nas respostas, é selecionada uma névoa remota com disponibilidade, e a requisição é encaminhada para processamento.

O nó primário também é capaz de lidar com diferentes tipos de protocolos, incluindo aqueles que exigem atuação como intermediário na comunicação. Em todos os casos, realiza a adequação inicial do canal recebido, interpretando o conteúdo e convertendo-o para um formato interno uniforme antes de encaminhar para um nó local ou remoto.

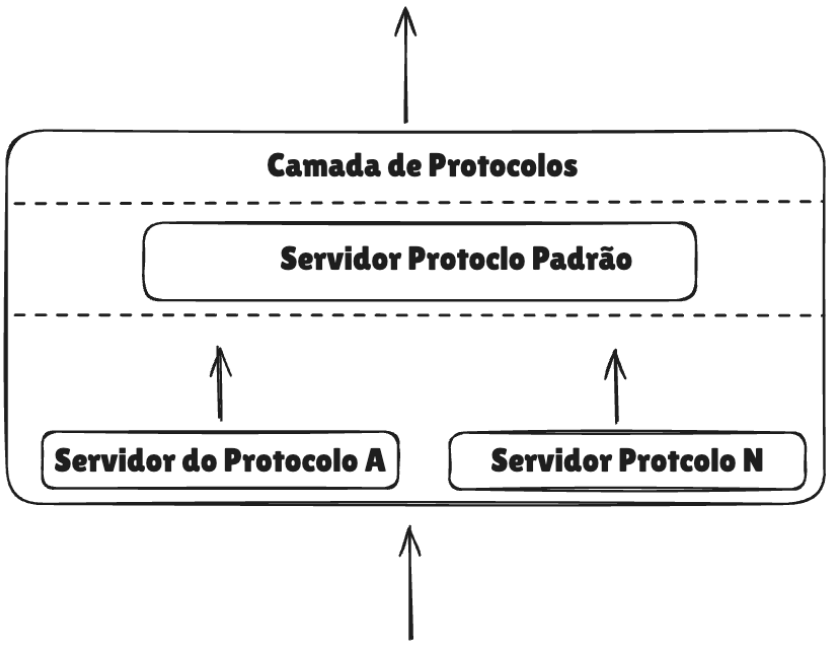
4.2.2 Nós de Névoa

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.

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,

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



Fonte: Do autor.

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

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.

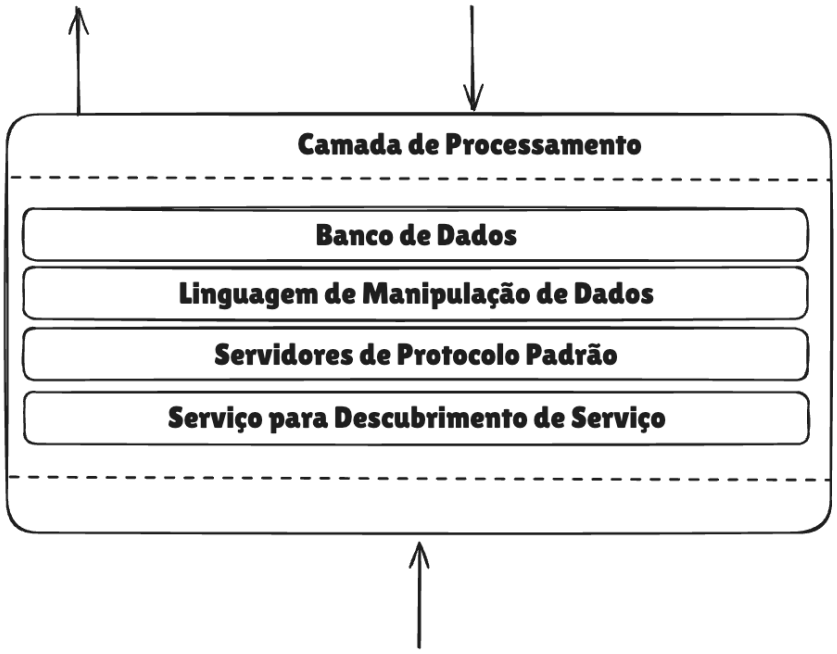
4.2.3 Nó Agregador

O nó agregador atua como ponto central para receber os resultados pré-processados provenientes dos nós de névoa. Em vez de cada nó de névoa transmitir seus dados diretamente à nuvem, o agregador coleta e organiza essas informações, reunindo-as em um único conjunto consolidado.

Esse processo tem como efeito a redução da quantidade de requisições enviadas à nuvem, agrupando múltiplos envios menores em transmissões mais amplas e organizadas. Embora a redução de tráfego seja um dos resultados esperados, o principal papel do agregador é otimizar o fluxo de comunicação entre a camada de névoa e a nuvem, facilitando o controle, a rastreabilidade e a padronização dos dados enviados.

O nó agregador também pode executar transformações adicionais sobre os

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



Fonte: Do autor.

dados recebidos, como ajustes de formato, enriquecimento de metadados ou aplicação de funções específicas definidas pela aplicação. Após esse tratamento, os dados são preparados em lotes ou fluxos contínuos e encaminhados à nuvem de acordo com a política de envio configurada.

5 METODOLOGIA

5.1 TIPO DE PESQUISA

Este trabalho caracteriza-se como uma pesquisa aplicada, de natureza tecnológica, com abordagem predominantemente qualitativa na fase de levantamento e definição de requisitos, e quantitativa na etapa de definição dos critérios de avaliação. O objetivo central é propor e modelar uma arquitetura modular de computação em névoa com suporte à comunicação entre múltiplas névoas.

5.2 PROCEDIMENTOS METODOLÓGICOS

5.2.1 Definição de Requisitos da Arquitetura

A partir das lacunas identificadas na literatura (conforme discutido no Capítulo 3.1), foram definidos requisitos funcionais e não funcionais, como:

- Suporte à comunicação entre diferentes domínios de névoa;
- Capacidade de integração de dispositivos heterogêneos;
- Modularidade para implementação de múltiplos serviços;
- Balanceamento de carga e distribuição dinâmica de tarefas;
- Redução de latência e otimização do tráfego de rede.

5.2.2 Modelagem da Arquitetura Modular

A arquitetura proposta foi estruturada em três camadas principais:

- **Nó Primário**: responsável pelo registro de dispositivos, balanceamento de carga entre nós locais e outros nós primários, e redirecionamento de tarefas.
- **Nós de Névoa**: responsáveis pela execução de serviços, pré-processamento e armazenamento temporário de dados.
- **Nó Agregador**: responsável por consolidar informações e preparar os dados para envio otimizado à nuvem.

Diagramas conceituais e funcionais foram elaborados para representar os componentes, fluxos de dados e protocolos de comunicação.

5.2.3 Implementação em Ambiente Controlado

A arquitetura foi implementada em ambiente local utilizando contêineres Docker, simulando medidores, nós de névoa e agregadores. Protocolos como HTTP e CoAP foram empregados para comunicação, e o gerenciamento de serviços foi desenvolvido em Python e Node.js. Essa configuração permitiu avaliar a viabilidade técnica e a escalabilidade horizontal da proposta.

5.2.4 Definição dos Critérios de Avaliação

Embora a validação experimental completa não tenha sido realizada no escopo deste trabalho, foram definidos critérios para avaliação futura:

- Tempo de resposta (ms);
- Latência média (ms);
- Taxa de entrega de pacotes (%);
- Escalabilidade (número de nós e dispositivos suportados).

5.3 AMBIENTE E FERRAMENTAS UTILIZADAS

A implementação e simulação utilizaram os seguintes recursos:

- Linguagens e frameworks: Python, Node.js;
- Infraestrutura de virtualização: Docker;
- Protocolos: HTTP, CoAP;
- Hardware: ambiente local com suporte a múltiplos contêineres e rede simulada;
- Ferramentas de modelagem: draw.io e ferramentas de diagramação compatíveis com LaTeX.

5.4 FLUXO METODOLÓGICO

6 TESTES

Aa.

6.1 CONTEXTUALIZAÇÃO DO PROBLEMA

Aa.

6.2 COMPUTAÇÃO EM NÉVOA

Aa.

6.3 PROTOCOLOS

Aa.

6.4 GRAPHQL

Aa.

6.5 HPCC SYSTEMS

Aa.

7 RESULTADOS

Este *template* contém algumas seções criadas na tentativa de facilitar seu uso. No entanto, não há um limite máximo ou mínimo de seção a ser utilizado no trabalho. Cabe a cada autor definir a quantidade que melhor atenda à sua necessidade.

8 CONCLUSÃO

As conclusões devem responder às questões da pesquisa, em relação aos objetivos e às hipóteses. Devem ser breves, podendo apresentar recomendações e sugestões para trabalhos futuros.

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.

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.

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

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.

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.