UNIVERSIDADE FEDERAL DE PERNAMBUCO
CENTRO DE INFORMÁTICA
PROGRAMA DE PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

ANDERSON MELO DE MORAIS

OMNIBLOCK: UMA BLOCKCHAIN PARA IOT COM INTEGRAÇÃO DE MÚLTIPLOS ALGORITMOS DE CONSENSO

Recife

2025

ANDERSON MELO DE MORAIS

OMNIBLOCK: UMA BLOCKCHAIN PARA IOT COM INTEGRAÇÃO DE MÚLTIPLOS ALGORITMOS DE CONSENSO

Tese apresentada ao Programa de Pós-graduação em Ciência da Computação do Centro de Informática da Universidade Federal de Pernambuco, como requisito para obtenção do grau de Doutor em Ciência da Computação.

Área de Concentração: Sistemas Distribuídos

Orientador: Nelson Souto Rosa

Coorientador: Fernando Antônio Aires Lins

Recife

2025

.Catalogação de Publicação na Fonte. UFPE - Biblioteca Central

Morais, Anderson Melo de.
Omniblock: Uma Blockchain para IoT com integração de múltiplos algoritmos de consenso / Anderson Melo de Morais. - Recife, 2025.
162f.: il.

Tese (Doutorado) - Universidade Federal de Pernambuco, Centro de Informática, Programa de Pós-Graduação em Ciência da Computação, 2025.
Orientação: Nelson Souto Rosa.
Coorientação: Fernando Antonio Aires Lins.
Inclui referências e apêndices.

1. Blockchain; 2. Internet das coisas; 3. Algoritmos de consenso; 4. Desempenho; 5. Segurança da informação. I. Rosa, Nelson Souto. II. Lins, Fernando Antonio Aires. III. Título.

UFPE-Biblioteca Central

Anderson Melo de Morais

“Omniblock: Uma Blockchain para IoT com Integração de Múltiplos Algoritmos de Consenso”

Tese de Doutorado apresentada ao Programa de Pós-Graduação em Ciência da Computação da Universidade Federal de Pernambuco, como requisito parcial para a obtenção do título de Doutor em Ciência da Computação. Área de Concentração: Redes de Computadores e Sistemas Distribuídos.

Aprovada em: 08/08/2025.

_____

Orientador: Prof. Dr. Nelson Souto Rosa

BANCA EXAMINADORA

_____

Prof. Dr. Paulo Roberto Freire Cunha
Centro de Informática / UFPE

_____

Prof. Dr. Carlos Andre Guimaraes Ferraz
Centro de Informática / UFPE

_____

Prof. Dr. Allan Edgard Silva Freitas
Instituto Federal da Bahia / IFBA

_____

Prof. Dr. Eduardo Luzeiro Feitosa
Instituto de Computação / UFAM

_____

Prof. Dr. Milton Vinicius Morais De Lima
CESAR School

Aos meus filhos, João Rafael e José Felipe, e a minha esposa Maria Dayanne, que me dão forças para seguir em frente.

AGRADECIMENTOS

A Deus, que em seus misteriosos desígnios me permitiu chegar até aqui.

A minha esposa Maria Dayanne e aos meus filhos, João Rafael e José Felipe, que estiveram ao meu lado, me encorajando em todos os momentos. A minha mãe, Sônia Melo, pela preocupação e esforço para garantir ao seu filho uma educação de qualidade.

Aos meus orientadores Nelson Rosa e Fernando Aires pela orientação, incentivo, seriedade, e compromisso. Agradeço o apoio, mesmo nas dificuldades, sempre me deram forças para seguir em frente.

Ao professor Gilson Moraes, que me acompanhou durante o ensino médio, me incentivando a ingressar na universidade e buscar sempre ir mais longe.

Aos membros da minha banca de tese, Carlos André Guimarães Ferraz, Eduardo Luzeiro Feitosa, Allan Edgard Silva Freitas, Paulo Roberto Freire Cunha e Milton Vinicius Morais de Lima, pelos valiosos conselhos durante esse processo. O conhecimento, a experiência e as críticas construtivas que compartilharam foram decisivos para a conclusão deste trabalho.

E a todos, professores, amigos e familiares, que contribuíram direta ou indiretamente para a realização deste trabalho.

Sabemos que Deus age em todas as coisas para o bem daqueles que o amam, dos que foram chamados de acordo com o seu propósito. (Rm 8:28).

RESUMO

A Internet das Coisas (do inglês *Internet of Things* (IoT)) tornou-se popular ao conectar dispositivos do cotidiano à Internet. No entanto, esses dispositivos frequentemente lidam com dados sensíveis dos usuários, como informações de saúde e localização, exigindo um processamento rápido das transações; por isso, é crucial garantir o desempenho do registro desses dados. A tecnologia Blockchain, originalmente criada para suportar criptomoedas, passou a ser adotada em diversas áreas, como saúde e logística, e, mais recentemente, na IoT, para proteger as informações. Porém, as blockchains tradicionais, baseadas em blocos encadeados e algoritmos de consenso de alto consumo computacional, não atendem satisfatoriamente às exigências da IoT, que requer confirmação rápida de blocos e baixo uso de CPU, memória e bateria. Além disso, escolher qual(is) algoritmo(s) de consenso usar em blockchains para IoT é um desafio em aberto. Considerando essas limitações, esta tese apresenta o desenvolvimento de uma blockchain, denominada *OmniBlock*, que combina diversos algoritmos de consenso. A tese também elabora uma estratégia, utilizando algoritmos genéticos, para seleção das combinações de algoritmos de consenso mais adequadas para IoT. A blockchain *OmniBlock* foi implementada utilizando *Directed Acyclic Graph* (DAG). A hipótese desta tese é de que o uso de uma blockchain baseada em DAG, que combine múltiplos algoritmos de consenso, melhora o desempenho do registro de dados, através da redução do tempo de criação de novos blocos e do consumo de recursos computacionais em ambientes IoT. A blockchain desenvolvida é avaliada através de experimentos que comparam a *OmniBlock*, com suas diversas combinações de algoritmos de consenso, a uma blockchain tradicional, com blocos lineares e que não utiliza combinações de algoritmos de consenso. Os resultados mostram uma redução no tempo de confirmação de blocos e no uso de recursos computacionais, contribuindo para o reforço do desempenho do registro de dados em ambientes IoT.

Palavras-chaves: Blockchain. Internet das Coisas. Algoritmos de Consenso. Desempenho. Segurança da Informação.

ABSTRACT

The Internet of Things (IoT) has become popular by connecting everyday devices to the internet. However, these devices often handle sensitive user data, such as health and location information, which requires fast transaction processing, therefore, ensuring the performance of data recording is crucial. Blockchain technology, originally created to support cryptocurrencies, has been adopted in various areas, such as healthcare and logistics, and more recently in the IoT, to protect information. However, traditional blockchains, based on chained blocks and computationally intensive consensus algorithms, do not satisfactorily meet the demands of the IoT, which requires fast block confirmation and low CPU, memory, and battery usage. Furthermore, choosing which consensus algorithm(s) to use in IoT blockchains is still an open challenge. Considering these limitations, this thesis presents the development of a blockchain, called *OmniBlock*, that combines several consensus algorithms. The thesis also develops a strategy, using genetic algorithms, for selecting the most appropriate consensus algorithm combinations for IoT. The *OmniBlock* blockchain was implemented using a *Directed Acyclic Graph* (DAG). The hypothesis of this thesis is that the use of a DAG-based blockchain, which combines multiple consensus algorithms, improves data logging performance by reducing the time to create new blocks and the consumption of computational resources in IoT environments. The developed blockchain is evaluated through experiments that compare *OmniBlock*, with its various combinations of consensus algorithms, to a traditional blockchain with linear blocks that does not use consensus algorithm combinations. The results show a reduction in block confirmation time and computational resource consumption, contributing to improved data logging performance in IoT environments.

Keywords: Blockchain. Internet of Things. Consensus Algorithms. Performance. Information Security.

LISTA DE FIGURAS

LISTA DE CÓDIGOS

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

AGA	*Adaptive Genetic Algorithm*
AMQP	*Advanced Message Queuing Protocol*
CoAP	*Constrained Application Protocol*
DAG	*Directed Acyclic Graph*
dPoS	*delegated Proof of Stake*
IoT	*Internet of Things*
MQTT	*Message Queue Telemetry Transport*
P2P	*Peer-to-peer*
PBFT	*Practical Byzantine Fault Tolerance*
PoA	*Proof of Authority*
PoB	*Proof of Burn*
PoC	*Proof of Capacity*
PoET	*Proof of Elapsed Time*
PoI	*Proof of Importance*
PoL	*Proof of Luck*
PoS	*Proof of Stake*
PoW	*Proof of Work*
RPL	Routing Protocol for Low-Power and Lossy Networks
SGA	*Simple Genetic Algorithm*
TEE	*Trusted Execution Environment*

SUMÁRIO

1 INTRODUÇÃO

Este capítulo apresenta o contexto da pesquisa, a formulação do problema e a hipótese, os objetivos da pesquisa, os procedimentos metodológicos e a estrutura deste trabalho.

1.1 CONTEXTO DA PESQUISA

A Internet das Coisas (*Internet of Things* ou IoT) ganha cada vez mais notoriedade e importância ao transformar objetos cotidianos em dispositivos inteligentes e autônomos (REYNA *et al.*, 2018). Para isso, esta tecnologia utiliza-se de dispositivos inteligentes, sensores, infraestrutura de rede e protocolos da Internet.

A IoT permite a coleta e a análise de dados, possibilitando a automação de processos, otimização na utilização de recursos e melhoria na tomada de decisão em diversos setores, como saúde, agricultura, transporte, manufatura e cidades inteligentes. As aplicações de IoT incluem desde o monitoramento da saúde pessoal (KASHANI *et al.*, 2021) até sistemas de monitoramento ambiental (ULLO; SINHA, 2020), rastreamento de ativos (WU *et al.*, 2022), controle de produção industrial (SOORI; AREZOO; DASTRES, 2023) e gestão de energia (BENHAMAID; BOUABDALLAH; LAKHLEF, 2022).

A Internet das Coisas possui um grande potencial para facilitar a vida das pessoas e melhorar a eficiência de processos empresariais. Para as pessoas, a IoT viabiliza o acesso a casas inteligentes, que podem ajustar automaticamente a temperatura, a iluminação e os eletrodomésticos de acordo com as preferências individuais, além de possibilitar o monitoramento da saúde por meio de dispositivos vestíveis que enviam dados em tempo real para profissionais de saúde. A IoT também possibilita a utilização de veículos inteligentes através da interconexão entre os veículos e a infraestrutura urbana, como semáforos e sinalizações inteligentes.

Para as empresas, a IoT oferece oportunidades de otimização de processos, redução de custos e criação de novos modelos de negócios (AHMETOGLU; COB; ALI, 2022). Isso inclui desde a automação de linhas de produção e gerenciamento inteligente de estoques até a melhoria da logística por meio do rastreamento em tempo real de ativos e produtos (WU *et al.*, 2022). Em setores como o agronegócio, a IoT pode aumentar a eficiência e a produtividade, reduzir desperdícios, melhorar a gestão das propriedades rurais e minimizar o impacto ambiental (ACHARYA *et al.*, 2022).

A IoT está revolucionando a forma como as pessoas interagem com o mundo ao seu redor, tornando-o mais conectado, inteligente e adaptável às suas necessidades. A IoT vem se tornando um elo entre diversas tecnologias, ao permitir o desenvolvimento de novas aplicações, conectar objetos físicos e apoiar a tomada de decisão (KIZZA, 2024).

Porém, em muitos ambientes IoT, existe a necessidade de que o registro de dados ocorra de forma rápida, devido à natureza sensível das informações coletadas e processadas pelos dispositivos. A agilidade no registro de dados reduz o risco de possíveis adulterações (MORRIS *et al.*, 2022). Além disso, muitos dispositivos IoT possuem limitações de recursos computacionais, o que dificulta a adoção de mecanismos robustos de verificação e proteção dos dados (MILLAR; HADDADI; MADHAVAPEDDY, 2025).

O armazenamento inapropriado dos dados pode resultar em violações de privacidade e, consequentemente, levar ao comprometimento dos dados. Por exemplo, câmeras de segurança conectadas podem ser hackeadas, permitindo adulteração não autorizada de imagens e vídeos privados de residências ou empresas.

Da mesma forma, dispositivos que monitoram a saúde podem transmitir dados pessoais para servidores remotos sem criptografia adequada, expondo informações sensíveis a possíveis ataques e modificações. Além disso, quando dados de localização ou de saúde são coletados e transmitidos, há o risco de que as informações sejam alteradas ou corrompidas, comprometendo a exatidão dos registros.

Em síntese, o problema do desempenho no registro de dados da IoT ressalta a necessidade de implementação de mecanismos que assegurem que as informações geradas pelos dispositivos sejam processadas e armazenadas de forma rápida e confiáveis ao longo de todo o ciclo de vida.

1.2 DESCRIÇÃO DO PROBLEMA

De acordo com Mathur *et al.* (2024), em ambientes IoT, muitos dispositivos e sensores apresentam recursos computacionais limitados, o que pode dificultar a execução de métodos tradicionais de segurança da informação, como algoritmos de criptografia.

A utilização de dispositivos IoT em diversos domínios, como em casas inteligentes, cidades inteligentes, IoT industrial e cuidados com a saúde (PERWEJ *et al.*, 2019), realça a necessidade da adoção de estratégias que melhorem o desempenho do registro de dados. É fundamental que as aplicações inteligentes sejam capazes de registrar dados de forma rápida e com um

consumo eficiente de recursos computacionais, para atender aos requisitos de cada domínio da IoT.

Uma tecnologia utilizada para o registro de dados nos ambientes IoT é a Blockchain (MATHUR *et al.*, 2023), que faz uso de criptografia e registro distribuído de informações para proporcionar confiabilidade e proteção dos dados. Devido à sua arquitetura descentralizada e mais resistente a ataques, a blockchain melhora a integridade do registro de dados gerados por dispositivos da IoT (SADAWI; HASSAN; NDIAYE, 2021).

No entanto, o uso de blockchain tradicional, embora eficaz em outros contextos, como sistemas de pagamentos e registro de documentos, apresenta desafios quando confrontado com as restrições de recursos computacionais e com a necessidade de respostas rápidas impostas pelos dispositivos IoT (ZHUANG *et al.*, 2024). O uso de blockchain para IoT enfrenta desafios como escalabilidade, privacidade, interoperabilidade e consumo eficiente de recursos computacionais.

Uma abordagem utilizada para melhorar o desempenho das blockchains é o uso de *Directed Acyclic Graph* (DAG) (DIGITALE; MARTIN; GLYMOUR, 2022). O uso de DAG como base de uma arquitetura blockchain oferece vantagens como escalabilidade, paralelismo e menor tempo de resposta.

As propriedades dos DAGs, como paralelismo na inclusão de transações e tempo de confirmação de transações reduzido, demonstram potencial para resolver alguns dos principais desafios enfrentados pelas blockchains tradicionais, como altos tempos de confirmação de blocos e capacidade de processar apenas um número limitado de transações por segundo.

Um elemento central da tecnologia Blockchain é o algoritmo de consenso, responsável por validar novos blocos antes de sua inclusão na cadeia (LIAO; CHENG, 2023). Porém, os algoritmos de consenso clássicos não são totalmente adequados para IoT, pois em sua maioria demandam alto poder de processamento e recursos computacionais. Dessa forma, múltiplas abordagens têm sido propostas com o objetivo de ajustar ou ampliar a capacidade dos algoritmos de consenso para torná-los mais adequados à IoT (TSANG *et al.*, 2019), (HUANG *et al.*, 2019), (ZILNIEKS, 2021).

Algumas soluções existentes integram mais de um algoritmo de consenso na mesma blockchain, para se adequar aos vários domínios da IoT e utilizar o algoritmo de consenso ou a combinação mais adequada em cada situação (TAPWAL *et al.*, 2021), (ZHIPENG, 2019), (RASOLROVEICY; FOKAEFS, 2020). Essas combinações permitem aproveitar os pontos fortes de cada algoritmo de consenso e melhorar o desempenho da blockchain.

No entanto, de acordo com Uveise e Fathima (2024), a escolha de qual algoritmo de consenso mais eficiente a ser utilizado é um problema em aberto para blockchains em ambientes IoT. Embora existam diversos algoritmos disponíveis, não há uma única solução que se destaque como a melhor para todos os cenários. Isso se deve aos *trade-offs* envolvidos: cada algoritmo apresenta vantagens e desvantagens em termos de desempenho, latência, segurança e adequação aos recursos dos dispositivos IoT.

Reduzir o tempo de confirmação de blocos e o consumo de recursos computacionais está diretamente relacionado à melhora do desempenho de uma blockchain para IoT. Um tempo de confirmação mais curto diminui a janela de oportunidade para ataques e modificações da blockchain, fazendo com que os registros de dados sejam considerados definitivos de forma mais rápida e confiável (ISLAM *et al.*, 2025).

Além disso, a utilização de algoritmos de consenso que demandam menos processamento e memória torna viável a participação de um maior número de dispositivos na validação de transações. Esse aumento no número de validadores ativos fortalece a resiliência da rede contra falhas e tentativas de adulteração, elevando, assim, o desempenho do registro de dados.

Uma estratégia que pode ser utilizada para selecionar algoritmos de consenso em uma blockchain é o uso de Algoritmos Genéticos (AGs) (HOLLAND, 1975). De acordo com (YANG; WANG; WANG, 2020), o uso de AGs pode melhorar aspectos importantes do processo de consenso, como o tempo de confirmação de blocos.

Diante da necessidade de registrar de forma consistente os dados provenientes de dispositivos IoT, o problema de pesquisa tratado nesta tese é: *como melhorar o desempenho do registro de dados em ambientes IoT considerando as limitações de recursos dos dispositivos e o grande volume de dados gerados neste tipo de ambiente?*

1.3 HIPÓTESE

Diante da necessidade de melhorar o desempenho do registro de dados da IoT e considerando que a tecnologia Blockchain pode ser utilizada para este propósito, foi formulada a seguinte hipótese: *O uso de uma blockchain baseada em DAG, que combine múltiplos algoritmos de consenso, melhora o desempenho do registro de dados, através da redução do tempo de criação de novos blocos e do consumo de recursos computacionais em ambientes IoT.*

1.4 OBJETIVOS

O objetivo principal desta tese é desenvolver uma blockchain, baseada em DAG, que combine múltiplos algoritmos de consenso, para melhorar o desempenho do registro de dados em aplicações IoT. Assim, para atingir esse objetivo principal, foram definidos os seguintes objetivos específicos:

- *Projetar uma blockchain utilizando DAG:* desenvolver uma infraestrutura de registro distribuído que utilize DAG como base para o registro de dados IoT e que suporte a execução das combinações de algoritmos de consenso;
- *Combinar diversos algoritmos de consenso:* a realização de combinações entre diferentes algoritmos de consenso tem como objetivo melhorar o desempenho do registro de dados dos diferentes domínios da IoT em blockchain; e
- *Propor uma estratégia de seleção de combinações de algoritmos de consenso:* é necessário definir uma estratégia para escolher qual combinação será mais adequada para cada cenário da IoT, através da utilização de algoritmos genéticos.

1.5 PROCEDIMENTOS METODOLÓGICOS DA PESQUISA

Este trabalho é classificado como uma pesquisa aplicada pois busca apresentar soluções para um problema específico (PRODANOV; FREITAS, 2013). O método científico utilizado é o hipotético-dedutivo, que busca construir e testar uma possível solução (GIL, 2008).

A Figura 1.1 apresenta as cinco fases de desenvolvimento desta pesquisa. A primeira fase consistiu no planejamento da revisão de literatura, onde foram definidos o problema e as questões de pesquisa, bem como as *strings* de busca, as fontes de consulta e os critérios de inclusão e exclusão de artigos.

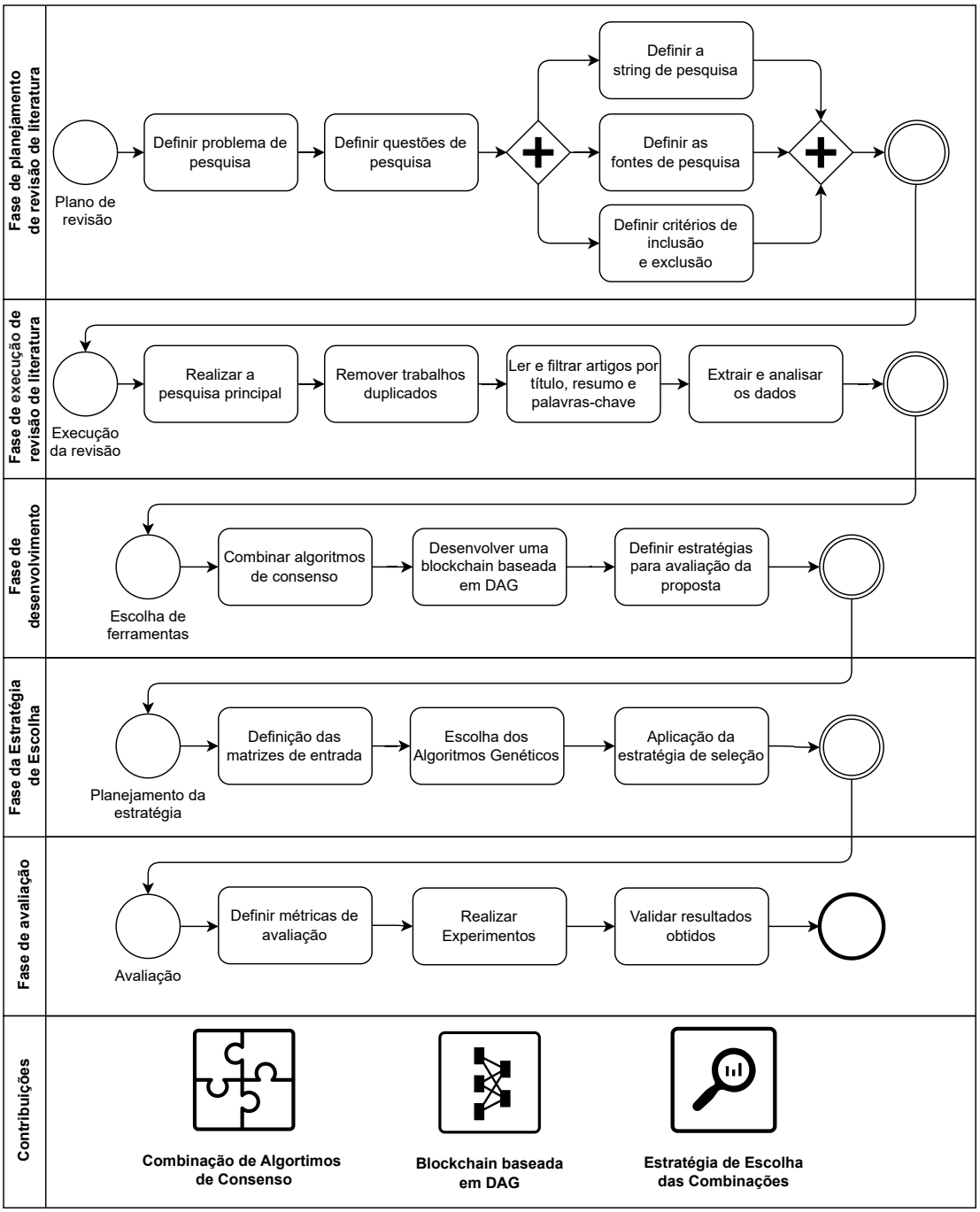
A segunda fase se deu através da execução da revisão de literatura, onde diversos artigos foram consultados com o objetivo de extrair informações relevantes e identificar os principais desafios em aberto na literatura relacionados a esta pesquisa.

Na terceira fase foi realizada a criação da blockchain, denominada *OmniBlock*, e a combinação entre diversos algoritmos de consenso.

Na quarta fase, foi elaborada uma estratégia para a escolha de algoritmos de consenso a serem utilizados para o registro de dados da IoT.

Na quinta fase, foi realizada a avaliação da solução proposta, através da definição de métricas e avaliação de desempenho.

Figura 1.1 – Procedimentos metodológicos da pesquisa



Fonte: O autor (2024).

1.6 CONTRIBUIÇÕES

A principal contribuição desta tese é o projeto e desenvolvimento da Blockchain *OmniBlock*. Outras contribuições deste trabalho são organizadas de acordo com cada etapa de seu desenvolvimento e são resumidas a seguir:

- Uso de DAG, que oferece vantagens em relação aos modelos de blockchains tradicionais, como maior velocidade na criação de blocos e menor consumo de recursos computacionais;
- Combinação dos principais algoritmos de consenso disponíveis atualmente. O uso dessas combinações potencializa os pontos fortes de cada algoritmo de consenso, como segurança e velocidade de execução, ao mesmo tempo em que busca mitigar as limitações e cada algoritmo;
- Definição de uma estratégia de escolha de algoritmos de consenso utilizando algoritmos genéticos. Com isso, é possível indicar quais combinações de algoritmos de consenso são mais adequadas para cada domínio da IoT.

Todas essas contribuições são parte fundamental da Blockchain *OmniBlock*, e colaboram com o seu objetivo principal, que é o de melhorar o desempenho do processo de registro de dados da IoT.

1.7 ESTRUTURA DO DOCUMENTO

Os capítulos restantes desta tese estão organizados da seguinte forma:

- Capítulo 2: neste capítulo são fornecidas informações sobre o funcionamento da tecnologia Blockchain, Internet das Coisas, *Directed Acyclic Graph* (DAG) e Algoritmos Genéticos;
- Capítulo 3: este capítulo descreve uma revisão sistemática da literatura que teve como propósito analisar os principais trabalhos relacionados ao problema de pesquisa abordado nesta tese;

- Capítulo 4: neste capítulo são detalhados o projeto e a implementação da Blockchain *OmniBlock*. Além disso, o capítulo apresenta as interações entre os algoritmos de consenso suportados por essa blockchain e descreve a elaboração da estratégia de escolha das combinações através do uso dos algoritmos genéticos;

- Capítulo 5: este capítulo apresenta uma avaliação de desempenho realizada com a Blockchain *OmniBlock*. Ainda neste capítulo são avaliados os algoritmos genéticos utilizados para a escolha dos algoritmos de consenso na blockchain.

- Capítulo 6: este capítulo apresenta as conclusões desta tese. São destacadas as principais contribuições, limitações e resultados obtidos, além de apontar as direções e trabalhos futuros.

2 CONCEITOS BÁSICOS

Neste capítulo, são introduzidos os conceitos básicos necessários para a compreensão desta tese. Mais especificamente, são apresentados e discutidos conceitos relacionados à Internet das Coisas, Blockchain, Algoritmos de Consenso, *Directed Acyclic Graph* (DAG) e Algoritmos Genéticos.

2.1 *INTERNET OF THINGS*

A *Internet of Things* (IoT) fornece capacidade computacional a objetos simples do cotidiano e possibilita conectá-los à Internet. A IoT abrange o processamento autônomo de dados e a comunicação entre diferentes dispositivos sem intervenção humana (ATZORI, 2017).

A disponibilidade é um requisito fundamental para a IoT. Para atendê-lo, as aplicações precisam do suporte de diversos dispositivos e protocolos de comunicação, desde sensores que captam dados até servidores para análise e extração de informações. Tudo isso requer a integração de dispositivos móveis e dispositivos de borda, como roteadores e controladores, que gerenciam a coleta e o processamento de informações (SAGIRLAR *et al.*, 2018).

O número de dispositivos IoT está crescendo, com isso, o tráfego de dados entre eles tem aumentado significativamente. Diante disso, é necessário melhorar a segurança e a privacidade dos dados na Internet das Coisas. De acordo com Mendez, Papapanagiotou e Yang (2018), os dispositivos IoT exigem um conjunto de requisitos de segurança, como autenticação segura, transmissão segura de dados, segurança dos dados usados pelos dispositivos e acesso aos dados por pessoas autorizadas.

2.1.1 Domínios da IoT

A IoT é uma tecnologia aplicada em múltiplos domínios, onde cada um possui exigências operacionais e restrições distintas. Enquanto alguns ambientes priorizam baixa latência e alta disponibilidade, por exemplo, aplicações industriais ou veículos conectados, outros domínios exigem forte privacidade e baixo consumo de energia, por exemplo, dispositivos domésticos e sensores agrícolas. A seguir, são descritas as características dos principais domínios da IoT.

- *Smart Home* (Casas Inteligentes): consistem em residências equipadas com sensores e

dispositivos conectados (iluminação, climatização, segurança, eletrodomésticos). Tem o objetivo de oferecer conforto, automação de tarefas, e segurança aos usuários. Porém esse domínio enfrenta desafios relacionados a privacidade dos moradores, interoperabilidade entre dispositivos de fabricantes diferentes, e limitações de energia/processamento em dispositivos embarcados (LEE *et al.*, 2020);

- *Healthcare* (Saúde): neste domínio são utilizados dispositivos médicos conectados, monitoramento remoto de pacientes e gestão de prontuários eletrônicos. Permite o monitoramento contínuo, respostas mais rápidas a emergências, telemedicina e maior qualidade nos cuidados médicos. No entanto, este domínio enfrenta desafios de privacidade e conformidade legal, requer alta disponibilidade e baixa latência (ZAABAR *et al.*, 2021);

- *Smart Agriculture* (Agricultura Inteligente): utiliza sensores no campo (solo, clima, irrigação) para otimizar a produtividade e o uso eficiente de água, insumos, monitoramento remoto e agricultura de precisão. Porém a conectividade em áreas rurais ainda é restrita, além disso, nesse domínio são necessários dispositivos resistentes a condições ambientais e soluções de baixo custo e manutenção mínima (LEI *et al.*, 2022);

- *Industrial IoT* (IoT Industrial): integra equipamentos industriais, automação de linhas de produção e manutenção preditiva. O uso de IoT nesse domínio busca aumentar a eficiência operacional, redução de paradas, e maior segurança do processo operacional. A IIoT requer baixa latência, tolerância a falhas, integração com sistemas legados e segurança contra ataques (ZUBAYDI; VARGA; MOLNÁR, 2023);

- *Smart Cities* (Cidades Inteligentes): integra sensores urbanos para gestão de transporte, iluminação, saneamento, segurança pública e meio ambiente. Tem o objetivo de otimizar serviços públicos, redução de custos, e melhoria da mobilidade e qualidade de vida. Esse domínio necessita de alta escalabilidade, privacidade dos dados, e coordenação entre múltiplos atores públicos e privados (PAUL *et al.*, 2021);

- *Smart Logistics* (Logística Inteligente): aplica IoT no rastreamento de cargas, gestão de armazéns e otimização de rotas. Busca oferecer rastreabilidade em tempo real da cadeia de suprimentos, redução de perdas, e automação de processos. Porém esse domínio possui desafios, como interoperabilidade entre diferentes empresas, segurança contra adulteração de dados e conectividade durante o transporte (UGOCHUKWU *et al.*, 2022);

- *Smart Energy* (Energia Inteligente): consiste em medidores inteligentes conectados para monitoramento da rede elétrica, balanceamento de carga e integração de fontes renováveis. Esse domínio proporciona redução de desperdícios e integração de sistemas de geração distribuída. No entanto necessita de baixa latência, alta disponibilidade para controle de rede e segurança contra ataques (KHAN *et al.*, 2023); e
- *Internet of Vehicles* (Internet dos Veículos): consiste na capacidade de conexão entre os veículos e a infraestrutura pública, para dar suporte a mobilidade, aumento da segurança viária e gerenciamento do tráfego. Esse domínio necessita de alta latência, resistência a ataques, alta conectividade ao longo das vias urbanas e privacidade dos dados de localização dos usuários (JAVED *et al.*, 2020).

Diante dos diferentes requisitos apresentados por cada domínio da IoT, torna-se necessária a implementação de estratégias que assegurem imutabilidade, rastreabilidade e segurança contra ataques e modificações.

2.1.2 Elementos da IoT

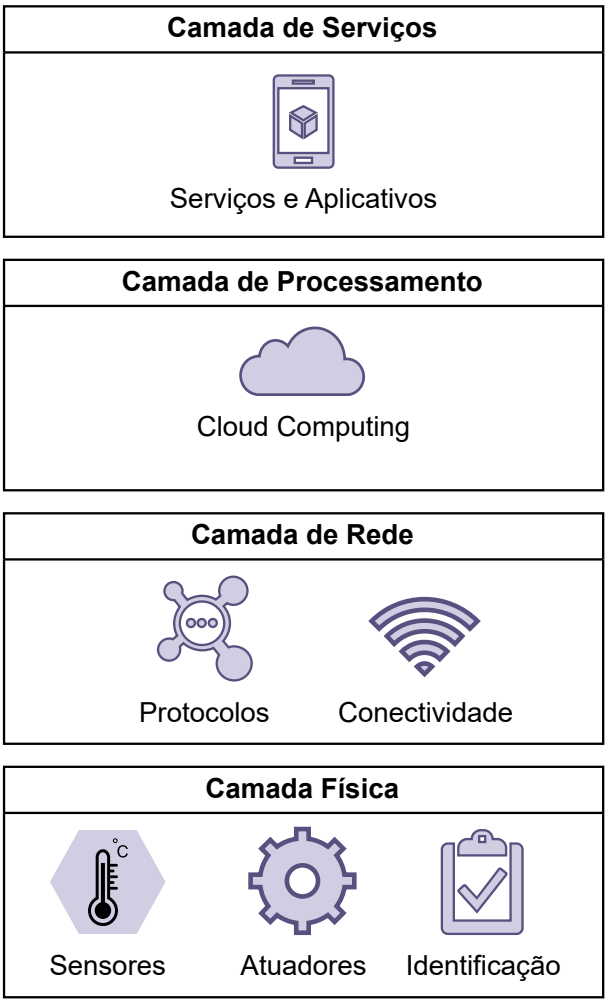
A IoT pode ser entendida como uma arquitetura em camadas hierárquica, cada uma reunindo componentes e funções fundamentais para o seu funcionamento (ALI *et al.*, 2022). As principais camadas da IoT são representadas na Figura 2.1, e seus componentes são descritos a seguir.

a) Camada Física:

   — Sensores: têm a função de reunir dados coletados por objetos dentro da rede e enviá-los a um servidor, banco de dados ou nuvem. Os sensores IoT podem coletar dados de de temperatura, umidade, luz, movimento, gás, pressão, som, dentre outros;

   — Atuadores: consistem em dispositivos que convertem sinais digitais ou comandos recebidos de sistemas de controle em ações físicas ou mudanças no ambiente. Por exemplo, abertura de válvulas, acionamento de motores, e controle de iluminação; e

Figura 2.1 – Camadas da Internet das Coisas



Fonte: O autor (2025).

— Identificação: atribuem identificadores únicos, por exemplo, etiquetas RFID, endereços MAC e IPv6, a cada dispositivo, garantindo rastreabilidade e endereçamento individual na rede.

b) Camada de Rede:

— Conectividade: engloba tecnologias de comunicação sem fio (Wi-Fi, Bluetooth LE, Zigbee, *LoRaWan*, NB-IoT) e cabeada (*Ethernet*), escolhidas conforme requisitos de distância, vazão e consumo de energia;

— *Edge/Fog Computing*: nós de borda executam pré-processamento e filtragem de dados próximos aos dispositivos para reduzir latência e tráfego de rede; e

— Protocolos de Transporte, Roteamento e Aplicação: existem diversos protocolos na IoT, como *Message Queue Telemetry Transport (MQTT)* (MISHRA; KERTESZ,

2020), *Constrained Application Protocol (CoAP)* (ARVIND; NARAYANAN, 2019), *Advanced Message Queuing Protocol (AMQP)* (NAIK, 2017), e *Routing Protocol for Low-Power and Lossy Networks (RPL)* (WINTER *et al.*, 2012). Esses protocolos permitem a comunicação entre dispositivos IoT e plataformas de nuvem, facilitando a transmissão de informações, controle remoto de dados e gerenciamento de dispositivos.

c) Camada de Processamento:

— Plataforma em Nuvem: oferece armazenamento em larga escala, poder de processamento e serviços de *analytics* (*big data, machine learning*) que extraem valor dos dados coletados.

d) Camada de Serviços:

— Serviços e Aplicativos: são compostos por *dashboards*, sistemas de monitoramento, automação predial, *smart grid*, rastreamento logístico, entre outros, que utilizam os dados processados para fornecer funcionalidades ao usuário final.

Esses elementos formam a base da IoT, permitindo a coleta, processamento e comunicação de dados entre dispositivos, o que viabiliza aplicações inteligentes e conectadas em diversos contextos.

2.1.3 Requisitos de Segurança

Muitos dispositivos IoT possuem limitações de recursos computacionais, como memória e processamento. Por isso, adotar mecanismos de segurança convencionais usados na Internet em geral não é suficiente. Os mecanismos de segurança convencionais são projetados para funcionar em dispositivos sem restrição severa de recursos, como *desktops* e *laptops*.

A IoT Security Foundation (2021) define um conjunto de requisitos de segurança para dispositivos IoT, organizados em quatro tópicos principais:

- *Identidade e autenticação*: cada dispositivo deve possuir um identificador único e imutável. Além disso, o sistema deve suportar autenticação robusta, impedindo senhas fracas ou repetidas e bloqueando possíveis ataques de força bruta;

- *Controle de acesso e credenciais*: toda senha ou chave precisa ser personalizada no primeiro uso e armazenada de forma criptografada. O acesso privilegiado (por exemplo, para configuração ou administração) só é permitido a contas autorizadas;
- *Atualização e integridade*: os dispositivos devem permitir atualizações seguras de *firmware* e configurações, verificando sempre a assinatura digital antes de instalar qualquer pacote; e
- *Governança e processos organizacionais*: é preciso designar um responsável formal por segurança de IoT e adotar políticas claras de gestão de vulnerabilidades, incluindo detecção, divulgação, correção e suporte .

Esses requisitos são fundamentais para a proteção das aplicações IoT. A adoção de soluções de segurança adequadas ao contexto da IoT é essencial para prevenir acessos indevidos, vazamentos de informação e comprometimento da integridade dos dados.

De acordo com a IoT Security Foundation (2021), existem diversas ameaças que podem comprometer a segurança de aplicações IoT. A seguir, são descritas as principais ameaças.

- *Identidade fraca e autenticação insuficiente:* ausência de identificadores únicos e uso de credenciais ou senhas fracas facilitam acesso não autorizado e clonagem de dispositivos. Em IoT essa pode ser uma ameaça crítica, pois os dispositivos operarem em ambientes não supervisionados;
- *Gestão de atualizações insegura ou inexistente:* falta de mecanismos seguros de atualização (*updates* não assinados, canais não autenticados) deixa dispositivos expostos a explorações e adulterações;
- *Comunicação e armazenamento sem proteção:* o tráfego de dados não criptografado permitem interceptação, alteração de comandos e adulteração de mensagens;
- *Interfaces de gestão e APIs expostas/inseguras:* consoles e APIs sem controlo de acesso ou validação de entradas podem permitir a realização de controle remoto e extração de dados;
- *Falta de monitoramento, e detecção de intrusão:* dispositivos que não geram *logs* úteis podem provocar dificuldades para identificar comprometimento dos dados ou responder a incidentes rapidamente; e

- *Falhas de governança, ciclo de vida e suporte:* ausência de papéis e políticas de segurança resultam em dispositivos inseguros e ineficazes, propensos a incidentes.

Diante dessas ameaças, é necessária a implementação de estratégias que garantam segurança e imutabilidade aos dados provenientes de dispositivos e sensores. A tecnologia Blockchain pode ser utilizada para mitigar algumas das potenciais ameaças de segurança da IoT.

2.2 BLOCKCHAIN

A tecnologia Blockchain realiza o registro de transações compartilhadas através de nós, em um sistema distribuído organizado como uma rede *Peer-to-peer* (P2P) (MONRAT; SCHELÉN; ANDERSSON, 2019). A blockchain é um ambiente seguro para registro de transações, pois, uma vez adicionado um novo bloco, ele não pode ser removido ou modificado sem que isso exija um grande poder computacional, ou seja, percebido pelos demais nós da rede.

Alguns nós, chamados de mineradores, realizam validação de transações por meio de algoritmos de consenso para confirmar informações e produzir novos blocos para a cadeia; este processo é conhecido como mineração. No contexto de IoT, a blockchain pode ser utilizada para autenticar, autorizar e auditar os dados gerados pelos dispositivos (HUANG; YANG; AJAY, 2023).

Cada bloco possui um cabeçalho que registra informações relevantes para a sua identificação, como o *hash* do bloco anterior e o *hash* de identificação do bloco. O *hash* é um código numérico que garante que as transações são válidas. Uma vez encontrado, o bloco pode finalmente ser adicionado à cadeia (MONRAT; SCHELÉN; ANDERSSON, 2019). No bloco também é armazenado o *timestamp*, que registra o momento exato em que ele foi criado.

2.2.1 Estrutura de um Bloco

Os blocos são estruturas de dados que armazenam as informações contidas na rede. Cada bloco guarda uma referência para o seu anterior; desta forma, é possível percorrer toda a cadeia até chegar ao bloco zero, também chamado bloco gênesis. A seguir são descritos os principais componentes de um bloco (MONRAT; SCHELÉN; ANDERSSON, 2019):

- *Transação:* é o conjunto de dados armazenado em cada bloco. Quando um nó recebe uma nova transação, ele a reenvia aos demais para que toda a rede contenha o mesmo

conteúdo. Após um novo bloco ser validado e incluído na cadeia, as transações nele contidas se tornam públicas e inalteráveis;

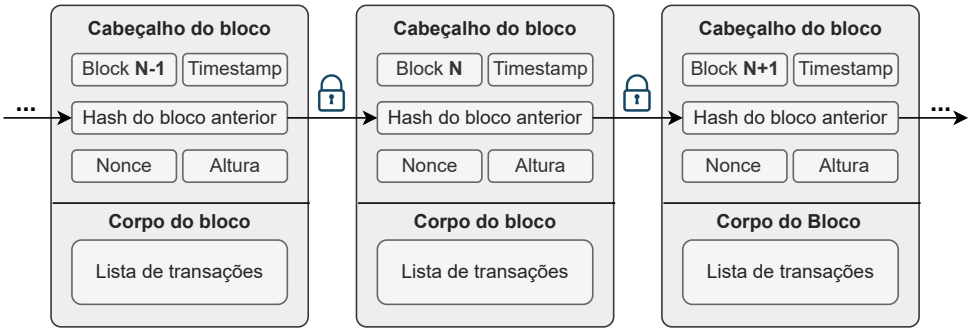
- *Hash do cabeçalho:* é um número que tem a função de identificar o bloco dentro da cadeia. Após todas as transações terem sido validadas e incluídas no bloco, o *hash* do cabeçalho é gerado. Este dado é incluído no bloco posterior para garantir o encadeamento dos blocos;
- *Hash do bloco anterior:* no cabeçalho de cada bloco é armazenado o *hash* de identificação do bloco anterior; isto permite o encadeamento sucessivo dos blocos. Desta forma, é possível percorrer toda a cadeia de blocos;
- *Nonce:* consiste em uma variável usada para alterar o resultado gerado pelo cabeçalho. Ela é utilizada para mostrar que um bloco foi validado e que atende aos critérios estabelecidos pela rede;
- *Altura do bloco:* os blocos são incluídos na cadeia de forma sequencial. A diferença entre a posição de um bloco e o bloco gênesis da cadeia é denominada altura do bloco;
- *Timestamp:* junto a cada bloco é armazenado o momento em que ele foi gerado. Este registro é um número que representa a data e a hora exata da sua criação; e
- Corpo do bloco: é onde ficam armazenadas as transações validadas. Representa o conteúdo principal do bloco e serve como base para o cálculo do *hash* do cabeçalho.

A Figura 2.2 apresenta as principais estruturas que compõem uma blockchain. O conteúdo da transação é armazenado no corpo do bloco. O encadeamento dos blocos é um dos fatores que faz as blockchains terem alto nível de segurança e imutabilidade.

2.2.2 Classificação quanto ao controle de acesso

A blockchain pode ser classificada como pública ou privada, de acordo com seu modelo de permissão e controle de acesso. A Figura 2.3 mostra as principais características desses dois grupos. Em uma blockchain pública, o acesso acontece de forma anônima e criptografada, e as regras de funcionamento são definidas pelos algoritmos de consenso; qualquer participante pode se tornar minerador para validar novos blocos. Suas principais aplicações são para projetos abertos e de grande escala (FERDOUS; CHOWDHURY; HOQUE, 2021). As blockchains públicas

Figura 2.2 – Estruturas básicas de uma blockchain

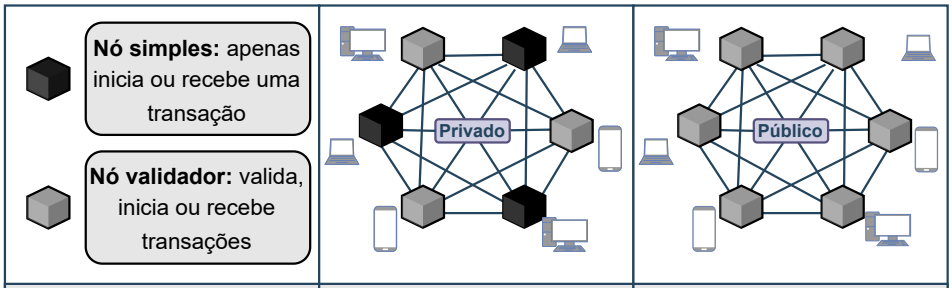


Fonte: O autor (2023).

mais conhecidas e utilizadas são a Bitcoin (Bitcoin Core Developers, 2025) e a Ethereum (Ethereum Foundation, 2025)

Na blockchain privada, o acesso à rede necessita de autorização e as regras de funcionamento são definidas pela comunidade de usuários. Apenas alguns nós podem ser escolhidos para validar novos blocos. Sua principal aplicação é em ambientes corporativos (YANG *et al.*, 2020a). A blockchain mais utilizada em projeto privados é a Hyperledger Fabric (Hyperledger Foundation, 2025)

Figura 2.3 – Tipos de blockchain



Acesso à rede:	Necessita de autorização	Acesso aberto
Quem regulamenta:	Legislações e regulações	Algoritmos de consenso
Quem são os validadores:	Grupo pré-selecionado	Anônimos
Potenciais aplicações:	Ambientes corporativos	Aplicações abertas

Fonte: O autor (2021).

As blockchains possuem dois tipos de nós (LAO *et al.*, 2020):

- *Full Nodes:* mantêm uma cópia completa da blockchain em seu armazenamento. Esses nós participam da criação, validação e adição de novos blocos. Desta forma, os *full nodes*

mantêm a segurança, o consenso e a confiabilidade dos dados da blockchain. Os nós completos requerem hardware e processamento de alto desempenho; e

- *Light Nodes:* são utilizados para armazenar uma cópia da blockchain. Os *light nodes* obtêm os novos blocos consultando os *full nodes*, e os adicionam em suas cadeias.

A classificação das blockchains quanto ao controle de acesso é fundamental para definir seu nível de descentralização, transparência e governança. Com isso, é possível escolher o tipo de blockchain mais adequado ao contexto da aplicação, seja ele um ambiente público e aberto ou um sistema restrito e corporativo.

2.2.3 Fundamentos de uma blockchain

Os objetivos fundamentais de uma blockchain são garantir segurança, escalabilidade e descentralização, características conhecidas como trilema blockchain (WERTH *et al.*, 2023). No entanto, de acordo com Vitalik Buterin (BUTERIN *et al.*, 2014), ao tentar aprimorar uma dessas características, as outras tendem a ser comprometidas. Por exemplo, aumentar a escalabilidade para processar um maior número de transações pode levar a um modelo mais centralizado, onde menos nós têm a capacidade de validar as transações, ou pode comprometer a segurança se os mecanismos de consenso forem simplificados demais.

Da mesma forma, fortalecer a segurança com protocolos robustos de consenso geralmente exige mais recursos computacionais e pode reduzir a escalabilidade. Portanto, o trilema ressalta que encontrar um equilíbrio entre esses três requisitos é um dos maiores desafios ao projetar e implementar uma blockchain.

De acordo com o teorema CAP Brewer (2000), em sistemas distribuídos não é possível garantir simultaneamente os três atributos: *Consistência*, todos os nós veem os mesmos dados ao mesmo tempo; *Disponibilidade*, cada requisição recebe uma resposta, mesmo que parcial; e *Tolerância a Falhas*, o sistema continua a operar mesmo se ocorrer uma falha na comunicação entre os nós.

O trilema blockchain está diretamente relacionado ao teorema CAP, pois ambos abordam as limitações do design de sistemas distribuídos, evidenciando a necessidade de escolhas que afetam o equilíbrio entre essas propriedades.

Como a blockchain é inerentemente distribuída, o seu desenvolvimento deve buscar equilibrar esses três pilares para garantir a consistência e o desempenho durante o registro dos

dados.

2.3 ALGORITMOS DE CONSENSO

Os algoritmos de consenso desempenham um papel fundamental na blockchain, pois são responsáveis por gerar novos blocos com segurança e confiabilidade (LIAO; CHENG, 2023). Os algoritmos de consenso foram propostos inicialmente para o uso de criptomoedas (YOUSUF *et al.*, 2021). Posteriormente, eles expandiram-se para diversas outras aplicações, incluindo contratos inteligentes, registros de propriedade, até começarem a ser adotados também no registro de dados em blockchains para IoT.

A primeira blockchain, desenvolvida para a criptomoeda Bitcoin, adotou o *Proof of Work (PoW)* como algoritmo de consenso (NAKAMOTO, 2008). Com o passar dos anos, surgiu a necessidade de projetar algoritmos de consenso mais eficientes que reduzissem o uso de memória e CPU, como o *Proof of Stake (PoS)* (HASSIJA *et al.*, 2019).

Ao utilizar PoW em uma blockchain, qualquer nó, com recursos computacionais suficientes, pode ser escolhido para verificar transações e criar um novo bloco. Através de um mecanismo de competição, alguns nós podem resolver um desafio computacional e obter uma recompensa através do processo conhecido como mineração. No trilema da blockchain, o PoW privilegia segurança e descentralização, frequentemente em detrimento da escalabilidade. Em termos de CAP, o PoW tende a priorizar consistência e tolerância a falhas (NAKAMOTO, 2008).

O algoritmo PoS busca garantir a segurança da blockchain enquanto reduz o consumo de recursos computacionais, como CPU e memória, e melhora a eficiência do registro de transações. O PoS é um algoritmo no qual qualquer participante da rede com uma quantidade suficiente de "*tokens*" pode utilizar suas criptomoedas por algum tempo e validar transações. O PoS possui maior escalabilidade, porém pode apresentar um menor nível de segurança e descentralização, em comparação ao PoW (KIAYIAS *et al.*, 2017).

Atualmente, os algoritmos PoW e PoS têm sido utilizados em muitas aplicações. Porém, ao longo do tempo, surgiram diversos outros algoritmos de consenso; alguns consistem em variações desses dois algoritmos, outros foram projetados para propósitos e aplicações específicas (ALI *et al.*, 2018)(LAO *et al.*, 2020). A seguir, são apresentados alguns dos principais algoritmos de consenso:

- *delegated Proof of Stake (dPoS)*: no algoritmo dPoS, blocos são gerados e assinados

por nós delegados. Os delegados são escolhidos através de votação pelos demais nós da blockchain. Ao selecionar delegados confiáveis, o algoritmo dPoS elimina a necessidade do novo bloco aguardar a confirmação de todos os nós antes de ser inserido na cadeia. O dPoS prioriza escalabilidade, porém é mais centralizado, devido a necessidade da assinatura dos nós delegados (LARIMER, 2014);

- *Proof of Luck (PoL)*: este algoritmo usa a geração de números aleatórios a partir de uma plataforma *Trusted Execution Environment* (TEE) para, através de um sorteio aleatório, escolher um nó líder para o processo de consenso. O algoritmo PoL oferece validação de transações com baixa latência, tempo de confirmação determinístico e baixo consumo de energia, porém pode apresentar um nível de segurança menor. O PoL enfatiza a disponibilidade, podendo apresentar um nível de segurança menor, ao depender da confiança no hardware TEE (MILUTINOVIC *et al.*, 2016);

- *Proof of Capacity (PoC)*: para obter o consenso, o algoritmo PoC usa espaço disponível no disco rígido em vez do poder de processamento computacional. Isso significa que quanto maior o espaço no disco rígido fornecido, maior será a probabilidade da mineração do novo bloco ser bem-sucedida. De forma semelhante ao PoW, possui foco em segurança e descentralização, porém a necessidade de armazenamento de dados pode comprometer a escalabilidade (DZIEMBOWSKI *et al.*, 2015);

- *Proof of Authority (PoA)*: no PoA, a validação de transações é realizada por nós conhecidos e confiáveis, chamados validadores. Ao contrário de algoritmos como PoW ou PoS, o PoA realiza a validação de transações de forma mais rápida e eficiente, priorizando a identidade e autoridade dos validadores. O PoA é comumente empregado em ambientes empresariais ou redes privadas, onde a identidade dos participantes é conhecida. O PoA favorece a escalabilidade, em detrimento da descentralização e, potencialmente, de parte da segurança (ANGELIS *et al.*, 2018);

- *Proof of Burn (PoB)*: o algoritmo PoB adota uma abordagem de consenso alternativa, "queimando" moedas em vez de usar recursos computacionais. O PoB funciona como mineração virtual e queima de moedas virtuais. Cada minerador pode escrever blocos proporcionais à quantidade de moedas que está disposto a queimar. Quanto mais moedas o minerador estiver disposto a queimar, mais poderosas serão suas plataformas de mineração e maiores as chances dele encontrar um novo bloco. O PoB tende a priorizar

segurança, devido ao custo econômico, porém podem haver implicações na escalabilidade (P4TITAN, 2014);

- *Proof of Importance (PoI)*: o processo de consenso realizado pelo PoI introduz o conceito de importância para medir a capacidade de um minerador criar novos blocos. O número de moedas disponíveis e o número de transações bem-sucedidas validadas por um minerador determina a sua importância dentro da blockchain e a probabilidade dele ser escolhido para realizar novas validações. Porém, neste algoritmo, há o risco de sobrecarga aos minerados que possuem maior importância. O PoI busca oferecer alta escalabilidade, no entanto, pode comprometer a descentralização, ao considerar apenas alguns nós como mais "importantes" (BACH; MIHALJEVIC; ZAGAR, 2018); e
- *Proof of Elapsed Time (PoET)*: o funcionamento do PoET é semelhante ao PoW, mas com menor consumo de recursos computacionais e de energia. Neste algoritmo, antes de criar um novo bloco, cada nó recebe um tempo, que é determinado aleatoriamente pelo hardware. O nó escolhido para criar um novo bloco será aquele cujo temporizador expira primeiro. O PoET prioriza escalabilidade e disponibilidade ao confiar em hardware seguro, porém o nível de segurança depende da qualidade da infraestrutura de hardware utilizada (SALIMITARI; CHATTERJEE; FALLAH, 2020).

Existem ainda, outros algoritmos de consenso que foram desenvolvidos especificamente baseados em tolerância a falhas bizantinas e suas variações. A falha bizantina ocorre quando um ou mais componentes presentes na rede falham ou apresentam comportamento malicioso (QI; GUAN, 2023). Portanto, é fundamental a adoção de estratégias para contornar a falha e permitir que o sistema continue funcionando de forma confiável. Um dos algoritmos mais conhecidos a lidar com falhas bizantinas é o *Practical Byzantine Fault Tolerance (PBFT)* (LAO *et al.*, 2020).

O PBFT é considerado o primeiro algoritmo de consenso a lidar com falhas bizantinas de forma assíncrona. Durante o processo de consenso, um nó cria uma solicitação e a envia para os outros nós da rede, que a encaminham para os seus pares até que uma resposta possa ser retornada após três rodadas de verificação. Mesmo que algum nó esteja com defeito ou seja malicioso, os outros continuam a executar o processo de verificação. O algoritmo PBFT apresenta baixo consumo de recursos computacionais, alta taxa de transferência e baixa latên-

cia na validação de novos blocos, porém apresenta limitações significativas de escalabilidade (CASTRO; LISKOV *et al.*, 1999);

Existem diversos outros algoritmos projetados para serem tolerantes a falhas, não apenas falhas bizantinas (nós maliciosos), como também falhas de parada, como, por exemplo, o *Raft*.

O *Raft* consiste em um algoritmo de tolerância a falhas baseado em votação. Embora seu modo de execução seja semelhante ao PBFT, ele não suporta diretamente falhas bizantinas, relacionadas a comportamentos maliciosos na rede. Porém, o *Raft* é muito eficiente na identificação de falhas de parada, onde até 50% dos nós podem deixar de funcionar. O *Raft* privilegia consistência e disponibilidade, mas seu funcionamento pode comprometer a descentralização e a segurança (ONGARO; OUSTERHOUT, 2015). Outros algoritmos de consenso, projetados para serem tolerantes a falhas, são:

- *Tendermint*: este algoritmo consiste em um protocolo de consenso tolerante à falhas bizantinas assíncrono. O *Tendermint* utiliza o PBFT de forma otimizada e requer apenas apenas duas rodadas de votação para chegar a um consenso. Os nós participantes da rede *Tendermint* são chamados de validadores, que se revezam propondo novos blocos de transações e votando neles. Este algoritmos busca oferecer consistência e segurança, porém pode apresentar comprometimento da descentralização e escalabilidade (KWON, 2014);

- *Ripple*: o algoritmo de consenso *Ripple* usa sub-redes coletivamente confiáveis dentro de uma rede mais extensa para chegar a um consenso sobre o problema bizantino. A validação de transações no *Ripple* possui um consumo de energia menor em comparação a ambientes que usam PoW. Os novos blocos são confirmados em segundos e requerem pouco processamento (TODD, 2015). O *Ripple* favorece a disponibilidade, ao custo de reduzir a descentralização, pois confia em subconjuntos de nós confiáveis para decisão; e

- *Stellar*: no algoritmo *Stellar*, cada nó seleciona um conjunto de outros nós confiáveis, denominados *quóruns*. O processo de consenso ocorre em rodadas, onde os nós trocam mensagens para propor e validar transações. Quando uma nova transação é enviada para verificação, ela é considerada aprovada se for autenticada por todos os nós que fazem parte daquele *quóruns* de nós selecionados. Este algoritmo busca um equilíbrio

entre disponibilidade e escalabilidade, mas pode apresentar problemas de consistência (MAZIERES, 2015).

Esses algoritmos demonstram diferentes abordagens para lidar com falhas bizantinas, oferecendo alternativas mais eficientes em comparação aos algoritmos tradicionais, como o PoW. Porém, a escolha do algoritmo de consenso mais adequado depende dos requisitos específicos de cada área de aplicação, e da necessidade de conciliar segurança, desempenho e consumo eficiente de recursos computacionais.

2.4 *DIRECTED ACYCLIC GRAPH*

Um *Directed Acyclic Graph* (DAG) é uma estrutura de dados orientada que consiste em vértices interconectados por arestas direcionadas, onde nenhuma aresta forma um *loop* (THULASIRAMAN; SWAMY, 1992).

Um DAG é um tipo de grafo cujas arestas possuem apenas uma direção (unidirecional) para conectar um vértice a outro. Em termos de ordenação topológica, os vértices devem seguir uma sequência tal que: para cada aresta, o vértice inicial da aresta ocorre antes na sequência do que o vértice final da aresta. Isso torna a ordenação acíclica, de modo que, uma vez visitado um vértice, ele não pode ser visitado novamente (THULASIRAMAN; SWAMY, 1992).

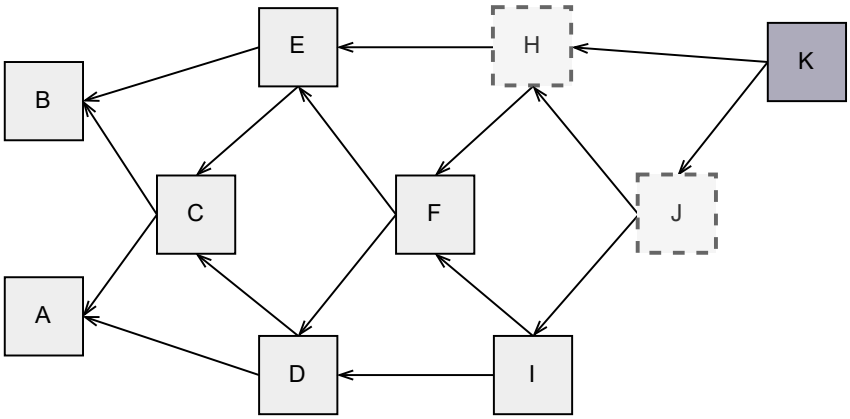
Assim como uma blockchain, que usa uma cadeia linear de blocos, uma estrutura de dados baseada em DAG também pode ser utilizada como um banco de dados distribuído para armazenar dados criptográficos de forma imutável e transparente (WANG *et al.*, 2021). A principal diferença está na arquitetura e nos algoritmos de consenso utilizados, o que favorece o DAG em termos de eficiência. Na literatura, estruturas DAG para registro distribuído são frequentemente denominadas *ledgers* (FAN *et al.*, 2021); neste trabalho, adotou-se a convenção de chamá-las de blockchain baseada em DAG.

Em uma blockchain baseada em DAG, quando uma nova transação é criada, os participantes da rede precisam resolver um desafio de *hash* criptográfico semelhante ao da blockchain tradicional. Posteriormente, o participante que criou a nova transação deve realizar a validação de duas transações anteriores, para finalmente poder adicionar o novo registro à blockchain. A Figura 2.4 apresenta um DAG, onde para incluir a nova transação K, é necessário validar as transações anteriores H e J.

Em DAG o processo de consenso não é imposto por um minerador único ou por rodadas de

votação como ocorre em algumas blockchains clássicas, mas ocorre no próprio ato de criação e referência de novas transações. Ao validar um conjunto de transações anteriores, a confiança daquelas transações aumenta. Assim, a escolha de quais transações anteriores escolher define quais ramos do grafo crescem e convergem para o estado aceito pela maioria, oferecendo paralelismo e maior taxa de processamento em comparação à blockchain linear. Essa dinâmica traz vantagens, como redução de latência, maior paralelização e escalabilidade, mas também exige proteção contra ataques de inserção de cadeias parasitas (CHEN *et al.*, 2022).

Figura 2.4 – Estrutura básica de um DAG



Fonte: O autor (2023).

2.4.1 IOTA e *Tangle*

A adoção de DAGs em implementações de blockchain tem sido objeto de estudo e experimentação. Em particular, as propriedades dos DAGs demonstraram potencial para resolver alguns dos desafios de escalabilidade enfrentados pelas blockchains tradicionais. A IOTA, por exemplo, é uma criptomoeda que emprega uma estrutura de DAG chamada *Tangle*, explorando sua escalabilidade e características de inserção e validação de transações assíncronas (SILVANO; MARCELINO, 2020).

De acordo com Son, Lee e Jang (2020), a arquitetura do *Tangle* difere de uma blockchain tradicional, pois utiliza uma estrutura onde os nós são representados pelas transações, em contraste com a blockchain tradicional, onde os nós são os dispositivos participantes da rede. Na prática, o Tangle é composto por blocos que contêm apenas uma transação cada.

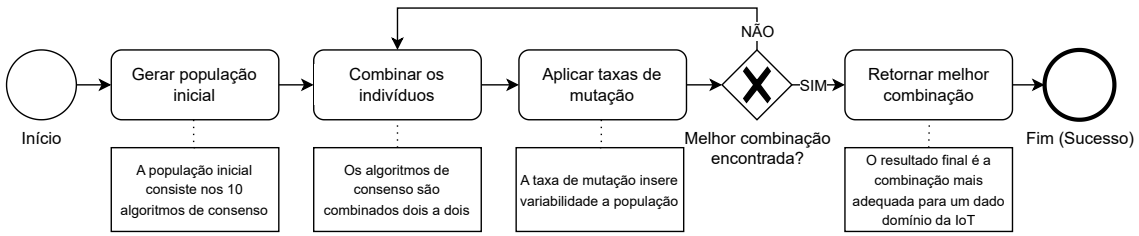
Cada nova transação registrada é marcada com um valor que representa o seu peso. O

peso de cada transação determina sua importância dentro da DAG. O *Tangle* define o peso cumulativo de uma transação como a soma dos pesos de outros nós que aprovam direta ou indiretamente a transação. Dessa forma, o peso cumulativo de uma transação crescerá à medida que novas validações são feitas, o que aumenta a confiabilidade de uma informação registrada no *Tangle* (SON; LEE; JANG, 2020).

2.5 ALGORITMOS GENÉTICOS

Os algoritmos genéticos (AGs) são uma classe de métodos de otimização inspirados nos princípios de seleção natural e evolução das espécies (HOLLAND, 1975). Os AGs funcionam evoluindo uma população de soluções candidatas por meio de processos iterativos, como seleção, cruzamento e mutação, buscando soluções quase ótimas para problemas complexos que necessitam de otimização. A Figura 2.5 mostra o funcionamento dos AGs.

Figura 2.5 – Funcionamento dos Algoritmos Genéticos



Fonte: O autor (2023).

O funcionamento dos AGs inicia-se gerando uma população inicial, em seguida são feitas combinações para gerar pares de indivíduos e produzir descendentes. Na etapa seguinte, são aplicadas taxas de mutação para inserir variabilidade na população; cada combinação é avaliada através de uma função, ao final a melhor combinação de indivíduos é retornada.

Em muitas aplicações, a utilização dos algoritmos genéticos ajuda a superar as limitações das técnicas tradicionais de otimização, pois eles são projetados para manter a diversidade na população de indivíduos iniciais, e têm a capacidade de se adaptar às mudanças na estrutura do problema que possam ocorrer ao longo do tempo. Atualmente, existem muitas variações de algoritmos genéticos, incluindo o *Simple Genetic Algorithm* (SGA) (VOSE, 1999) e o *Adaptive Genetic Algorithm* (AGA) (JAKOBOVIĆ; GOLUB, 1999).

2.5.1 *Simple Genetic Algorithm*

O *Simple Genetic Algorithm* (SGA) é a forma clássica dos algoritmos genéticos (HOLLAND, 1975), (GOLDBERG, 1986).

O SGA é um algoritmo de otimização que simula a evolução natural, onde uma população de soluções possíveis é aprimorada por meio de seleção, cruzamento e mutação para convergir para uma solução aproximada. O Código 2.1 apresenta o pseudocódigo do SGA utilizado neste trabalho.

Código 2.1 – SGA

```
1  Funcao AlgoritmoGeneticoSimples(W, N, G, μ, γ):
2      W  ← Pesos de cada dominio IoT
3      N  ← Tamanho da populacao
4      G  ← Numero de geracoes
5      μ  ← Taxa de mutacao
6      γ  ← Taxa de crossover
7      X  ← Conjunto de algoritmos permitidos para a posicao x
8      Y  ← Conjunto completo de algoritmos de consenso disponiveis
9
10     P ← GerarPopulacao(N)
11     // Cada cromossomo c e um par [x,y], tal que x ∈ X, y ∈ Y e x ≠ y
12     Para i de 1 ate G:
13         // Avalia a fitness de toda a populacao usando os pesos de W
14         Para cada cromossomo c ∈ P:
15             c.fitness ← Fitness(c, W)
16             // Calcula a fitness de c e armazena em um atributo de c
17         P_sel ← SelecionarMelhores(P, N/2)
18         // Seleciona os N/2 cromossomos com maior c.fitness
19         P_novo ← Reproduzir(P_sel, γ, μ)
20         // Aplica crossover (taxa γ) e mutacao (taxa μ), tal que x ≠ y
21         P ← P_novo
22     C_best ← max Fitness(c, W)
23     // C_best contem o cromossomo [x, y] de melhor fitness
24     Retornar C_best
```

Fonte: O autor (2025)

O resultado esperado como saída do SGA é um par de algoritmos de consenso $[x, y]$, que representa uma combinação de algoritmos de consenso, onde x é diferente de y. Para isso, os parâmetros de entrada utilizados são:

- W: vetor de pesos extraídos da matriz de decisão correspondente aos domínio de IoT;
- N: tamanho da população;

- G: número total de gerações;
- μ: taxa de mutação, que é a probabilidade de um gene (ou seja, um dos algoritmos presente no par) sofrer mutação;
- γ: taxa de *crossover*, ou taxa de cruzamento, corresponde a probabilidade de realizar o cruzamento ao gerar novos descendentes;
- X: consiste em um subconjunto de algoritmos de consenso (PoA, PoS, PoL, PoI, PoB, dPoS e PoET) que podem ser escolhidos para a primeira posição do cromossomo;
- Y: conjunto completo de todos os algoritmos de consenso disponíveis para a combinação; e
- Restrição: as posições x e y do cromossomo devem sempre possuir algoritmos diferentes ($x \neq y$).

A execução do algoritmo começa gerando uma população inicial P contendo N cromossomos (Linha 10). Cada cromossomo corresponde a um par $[x, y]$, onde x é escolhido aleatoriamente do conjunto de algoritmos permitidos X, e y é escolhido aleatoriamente do conjunto total de algoritmos Y. Uma verificação garante que, se x e y forem idênticos, y será selecionado novamente até que sejam diferentes. O objetivo é encontrar a melhor combinação para cada domínio de IoT definido em W.

Para cada cromossomo $c \in P$, a função de *fitness* $f(c, W)$ é calculada (Linha 15). A função de *fitness* combina os pesos (fornecidos para cada algoritmo de consenso) usando uma média ponderada baseada no vetor de pesos W. Esta função serva para avaliar se uma combinação específica de algoritmos de consenso é adequada para atender aos requisitos do domínio IoT considerado.

Em seguida, o algoritmo SGA seleciona os primeiros $N/2$ cromossomos da população atual com base em seus valores de aptidão (Linha 17). Essa seleção garante que apenas as combinações que obtiveram o melhor desempenho sejam transferidas para a próxima fase, que consiste na reprodução.

Na fase seguinte, os pares de cromossomos selecionados passam então por um cruzamento (Linha 19). A operação de cruzamento cria descendentes, pegando o primeiro gene de um dos pais e o segundo gene do outro, garantindo que, se o par resultante tiver algoritmos duplicados, ajustes sejam feitos.

Cada descendente é então sujeito à mutação, onde um gene (posição x ou y) pode ser substituído por um algoritmo escolhido aleatoriamente do respectivo conjunto (para x, de X; para y, de Y), garantindo novamente que $x \neq y$.

As Linhas 12-22 são repetidas por um número de G gerações. A cada nova geração, a população evolui idealmente em direção a melhores combinações (maior aptidão).

Após a execução de G gerações, o algoritmo SGA seleciona o cromossomo que obteve o maior valor de aptidão (Linha 24), este será indicado como a melhor combinação de algoritmos de consenso para o domínio IoT considerado. Por fim, o valor de C_{best} é retornado.

2.5.2 *Adaptive Genetic Algorithm*

O *Adaptive Genetic Algorithm* (AGA) (JAKOBOVIĆ; GOLUB, 1999), é uma variação do SGA que introduz mecanismos de autoajuste para os parâmetros de mutação e *crossover* durante o processo evolutivo. Ao contrário do SGA que utiliza taxas fixas, o AGA adapta esses parâmetros dinamicamente com base na diversidade da população ou em outros indicadores de desempenho. Isto ajuda a evitar a convergência prematura e melhora a capacidade do algoritmo de explorar o espaço de soluções.

O funcionamento do AGA também necessita de uma população inicial de soluções candidatas. Porém, a sua abordagem adaptativa torna o AGA especialmente indicado para a resolução de problemas complexos ou dinâmicos, onde os cenários de busca podem variar ao longo do tempo. Sua capacidade de adaptar as taxas conforme à necessidade da execução confere ao AGA um desempenho superior em termos de tempo de execução e qualidade das soluções encontradas.

Na implementação utilizada nesta tese, as taxas de mutação e cruzamento foram definidas como 0,05 e 0,5, respectivamente. De acordo com Lobo, Lima e Michalewicz (2007), uma taxa de mutação em torno de 0,05 é considerada moderada, pois garante a introdução de variações sem destruir as boas características já desenvolvidas. Uma taxa de cruzamento em torno de 0,5 é frequentemente empregada para incentivar a recombinação intensiva de características genéticas. De acordo com os autores, valores de taxas maiores podem ser agressivos e modificar demais os indivíduos da população (LOBO; LIMA; MICHALEWICZ, 2007). O Código 2.2 apresenta a AGA utilizada neste trabalho.

Código 2.2 – AGA

```
Funcao AlgoritmoGeneticoAdaptativo(W, N, G, μ, γ):
    W ← Pesos de dominio
    N ← Tamanho da populacao
    G ← Numero de geracoes
    μ ← Taxa de mutacao inicial
    γ ← Taxa de crossover inicial
    X ← Conjunto de algoritmos permitidos para a posicao x
    Y ← Conjunto completo de algoritmos de consenso disponiveis

    P ← GerarPopulacao(N)
    // Cada cromossomo c e um par [x,y], tal que x ∈ X, y ∈ Y e x ≠ y
    Para g de 1 ate G:
        // Avalia a fitness de toda a populacao usando os pesos de W
        Para cada cromossomo c em P:
            c.fitness ← Fitness(c, W)
            // Calcula a fitness de c e armazena em um atributo de c
        D ← CalcularDiversidade(P)
        // Calcula a diversidade
        μ_adapt, γ_adapt ← AjustarTaxas(D, μ, γ)
        // Ajusta as taxas de mutacao e crossover conforme a diversidade
        P_sel ← SelecionarMelhores(P, N/2)
        // Seleciona os N/2 cromossomos com maior c.fitness
        P_novo ← Reproduzir(P_sel, γ_adapt, μ_adapt)
        // Aplica crossover e mutacao com taxas adaptadas
        P ← P_novo
    C_best ← max Fitness(c, W)
    // C_best contem o cromossomo [x, y] de melhor fitness
    Retornar C_best
```

Fonte: O autor (2025)

Também no AGA, o resultado esperado como retorno é um par de algoritmos $[x, y]$, que representa uma interação de algoritmos de consenso, com x diferente de y. Para isso, os parâmetros de entrada utilizados são:

- W: vetor de pesos extraído da matriz de decisão correspondente aos requisitos do domínio específico da IoT;
- N: tamanho da população;
- G: número de gerações;
- μ: taxa de mutação, valor que representa a probabilidade inicial de mutação de um gene. No AGA a taxa de mutação pode ser alterada durante a execução;

- γ: taxa de *crossover*, ou cruzamento, corresponde à probabilidade inicial de ocorrerem cruzamentos entre indivíduos;
- X: subconjunto de algoritmos de consenso que podem ser escolhidos para o primeiro gene do cromossomo;
- Y: conjunto completo de todos os algoritmos de consenso disponíveis para a realização das combinações; e
- Restrição: as posições x e y do cromossomo devem sempre possuir algoritmos diferentes ($x \neq y$).

A execução do algoritmo AGA começa gerando uma população inicial P contendo N cromossomos (Linha 10). Cada cromossomo corresponde a um par $[x, y]$, onde x é escolhido aleatoriamente do conjunto de algoritmos permitidos X e y é escolhido aleatoriamente a partir do conjunto de algoritmos Y, respeitando a restrição de que $x \neq y$.

Na segunda etapa da execução do algoritmo, para cada cromossomo $c \in P$, é calculado o valor de *fitness*, ou de aptidão $f(c, W)$ (Linha 15), que é definido como a média ponderada dos pesos associados aos dois algoritmos no cromossomo em comparação com os requisitos do domínio IoT representado no vetor W.

Na próxima etapa do algoritmo, a diversidade D da população atual é calculada (Linha 17). Ou seja, medindo o número de valores de *fitness* únicos. Com base na taxa de diversidade D, a taxa de mutação μ e a taxa de cruzamento γ poderão ser ajustadas. Se a diversidade for baixa, o algoritmo AGA aumenta μ e diminui γ para promover a exploração de novos possíveis indivíduos. Se a diversidade for alta, o AGA diminui μ e aumenta γ para refinar as soluções mais promissoras.

Em seguida, os $N/2$ melhores cromossomos da população P são selecionados com base em sua aptidão para formar um *pool* de acasalamento P_{sel} (Linha 21). A partir daí, uma nova população P é então gerada realizando cruzamento e mutação nos genes selecionados (Linha 23). Nessa etapa, a função de cruzamento pode trocar o segundo gene entre os selecionados inicialmente, e a função de mutação altera um gene, garantindo que o novo gene gerado não duplique o outro no cromossomo.

As Linhas 12-25 são repetidas por um número de G gerações. A cada nova geração, os indivíduos vão sendo melhorados para apresentarem a melhor aptidão. Finalmente, o cromossomo C_{best}, aquele que apresentou o maior valor de aptidão, é retornado e indicado como

sendo a melhor combinação de algoritmos entre consenso para um determinado domínio IoT (Linha 28).

2.6 CONSIDERAÇÕES FINAIS

Este capítulo apresentou os conceitos fundamentais para a compreensão desta pesquisa. Foi apresentada uma visão geral da Internet das Coisas, suas características e requisitos principais. A tecnologia Blockchain também foi descrita, juntamente com os principais algoritmos de consenso que a compõem. Foram introduzidos ainda os *Directed Acyclic Graph*, estruturas de dados que atualmente vêm sendo utilizadas em implementações de blockchain. Por fim, foram apresentados os algoritmos genéticos, que são utilizados em problemas que requerem otimização.

3 TRABALHOS RELACIONADOS

Este capítulo apresenta os trabalhos relacionados a esta tese. Inicialmente, são apresentados trabalhos relacionados ao uso de combinações de algoritmos de consenso em blockchains para IoT. Em seguida, são descritas propostas de blockchains baseadas em DAG. Finalmente, são apresentados trabalhos relacionados ao uso de algoritmos genéticos em blockchains para IoT.

3.1 VISÃO GERAL

A revisão da literatura foi realizada em três fases, cada uma com foco em um dos aspectos da hipótese de pesquisa desta tese: combinações entre diferentes algoritmos de consenso; blockchains baseadas em DAG; e uso de algoritmos genéticos em blockchains para IoT. O detalhamento da metodologia e dos passos adotados na revisão podem ser encontrados no Apêndice A.

O objetivo da primeira fase da pesquisa foi conhecer as principais propostas relacionadas ao uso de combinações de múltiplos algoritmos de consenso em uma mesma blockchain e, com isso, identificar as motivações que levaram os autores a desenvolver tais combinações. A segunda fase teve o objetivo de identificar trabalhos relacionados ao uso de blockchains baseadas em DAG para IoT, e compreender quais são as vantagens do uso dessa estrutura em comparação com as blockchains lineares tradicionais. Por fim, na terceira fase da pesquisa, foram buscados trabalhos referentes ao uso de algoritmos genéticos para otimizações de blockchains para IoT.

Nas seções seguintes são detalhadas cada uma das fases desta revisão de literatura.

3.2 COMBINAÇÕES DE ALGORITMOS DE CONSENSO EM BLOCKCHAINS PARA IOT

Os algoritmos de consenso utilizados em blockchains tradicionais são pouco eficientes para IoT, pois normalmente possuem alta demanda de recursos computacionais, como memória, processamento e consumo de energia (TAPWAL *et al.*, 2021). Diante disso, vários autores propõem a realização de combinações entre diferentes algoritmos de consenso na mesma blockchain (XU *et al.*, 2019), (BAI; XIA; FU, 2019), (TAPWAL *et al.*, 2021), (ZHIPENG, 2019), (RASOLROVEICY; FOKAEFS, 2020).

Segundo Alrubei, Ball e Rigelsford (2021), a combinação de algoritmos de consenso é promissora, pois permite utilizar algoritmos já conhecidos com eficiência comprovada. Assim, combinações entre eles podem melhorar o desempenho do registro de dados em blockchains para IoT.

Na prática, a combinação de algoritmos de consenso é uma estratégia que permite aproveitar as vantagens individuais de diferentes algoritmos, permitindo que uma blockchain seja mais flexível e adaptável aos variados cenários de IoT. Por exemplo, pode-se combinar a segurança de algoritmos como PoW com o consumo eficiente de recursos computacionais do PoS (BAI; XIA; FU, 2019), ou a rápida confirmação de blocos de algoritmos como PBFT ou *Raft* (FU *et al.*, 2021).

Entretanto, a combinação também pode apresentar desvantagens, como o aumento da complexidade do sistema, dificuldade de manter a consistência entre os nós e possíveis conflitos entre algoritmos de consenso distintos. Estas desvantagens exigem um cuidado especial em termos de validação de transações, sincronização dos nós e gerenciamento da comunicação entre eles para garantir o desempenho da blockchain.

A Tabela 3.1 apresenta as propostas de combinação de algoritmos de consenso existentes atualmente.

Tabela 3.1 – Combinações de algoritmos de consenso existentes na literatura

	Combinação de Algoritmos de Consenso
Bai, Xia e Fu (2019)	PoW-PoS
Lunardi *et al.* (2019)	PBFT-WBC
Zhipeng (2019)	PoW-PBFT
Xu *et al.* (2019)	PoC-VCF
Wang, Li e Li (2020)	dPoS-PoP
Rasolroveicy e Fokaefs (2020)	PoET-Raft-PBFT
Tapwal *et al.* (2021)	PoW-PoS-PBFT
Fu *et al.* (2021)	Raft-PBFT
Alrubei, Ball e Rigelsford (2022)	PoW-PoA
Luo, Yu e Luo (2023)	Raft-PBFT
Huang, Yang e Ajay (2023)	dPoS-PBFT
Yusof, Zahilah e Othman (2024)	PoA-dPoS
Uveise e Fathima (2024)	PoA-dPoS
Reshi e Sholla (2025)	PoA-dPoS

Fonte: O autor (2023).

As combinações podem ser organizadas em quatro conjuntos principais: combinações que integram o PoW com outros algoritmos; aquelas que usam o PBFT combinado com outros algoritmos; combinações entre PoA-dPoS; e combinações entre PoP-dPoS, e PoC-VCF.

No primeiro conjunto, o uso de PoW tem o objetivo de melhorar a segurança do registro de dados. Bai, Xia e Fu (2019) combinam PoW-PoS para tirar proveito da segurança do PoW contra ataques, enquanto o uso de PoS proporciona melhor escalabilidade ao reduzir o custo computacional dos nós validadores.

Dando continuidade, Zhipeng (2019) combina PoW-PBFT: o PoW atua na seleção de blocos para fortalecer a segurança, e o PBFT coordena o consenso em rodadas de validação, diminuindo a latência e melhorando a vazão, o que resulta em maior descentralização e desempenho em comparação ao uso isolado de cada algoritmo.

Finalmente, Alrubei, Ball e Rigelsford (2022) combinam PoW-PoA, unindo a segurança do PoW com a alta velocidade de execução do PoA para reduzir o tempo de confirmação das transações em cenários de IoT.

No segundo conjunto de trabalhos, várias soluções exploram o PBFT por sua capacidade de execução rápida em ambientes distribuídos. Neste contexto, Lunardi *et al.* (2019) propõem uma blockchain modular que combina o PBFT com um algoritmo de Consenso Baseado em Testemunhas (do inglês *Witness Based Consensus* - WBC), para reduzir a taxa de latência e aumentar a vazão.

Posteriormente, Rasolroveicy e Fokaefs (2020) apresentam um algoritmo autoadaptativo que escolhe dinamicamente o algoritmo de consenso entre *Raft*, PoET e PBFT conforme métricas de custo, desempenho e segurança, otimizando também o consumo de energia.

De forma semelhante, Tapwal *et al.* (2021) realizam a combinação do PoW, PoS e PBFT numa solução para IoT Industrial. O PBFT é usado devido ao processamento rápido de transações; o PoW pela segurança; e o PoS para pelo consumo mais eficiente de energia.

Fu *et al.* (2021) e Luo, Yu e Luo (2023) propõem a combinação Raft-PBFT. De acordo com os autores, a integração desses dois algoritmos resulta em uma solução de alta escalabilidade, alta vazão e baixa latência.

Por fim, Huang, Yang e Ajay (2023) combinam dPoS-PBFT, delegando a geração de blocos ao dPoS e usando o PBFT para manter a integridade dos dados com baixo consumo de recursos.

No terceiro conjunto, três trabalhos exploram a combinação PoA-dPoS: Mohanta *et al.* (2024), Yusof, Zahilah e Othman (2024) e Reshi e Sholla (2025). O objetivo dessa combinação

é unir a agilidade do PoA na seleção de validadores, com a economia de recursos do dPoS, que delega o direito de criar blocos a um conjunto de nós escolhidos, tornando o consenso simultaneamente rápido e menos custoso computacionalmente.

Por fim, no quarto conjunto de trabalhos, Wang, Li e Li (2020) apresentam uma combinação entre o algoritmo de consenso *Proof of Probability* (PoP) e o *Delegated Proof of Stake* (dPoS), buscando compensar as deficiências de segurança do PoP com a eficiência do dPoS para evitar ataques maliciosos.

De forma semelhante, Xu *et al.* (2019) descrevem um algoritmo de consenso híbrido composto pelo *Proof of Credit* (PoC) e um algoritmo baseado em votação. O objetivo dessa combinação é obter latência reduzida, integrando o PoC a um algoritmo de baixo custo computacional.

3.3 BLOCKCHAINS BASEADAS EM DAG PARA IOT

Duas arquiteturas de blockchain são normalmente adotadas em IoT: a blockchain linear e a blockchain baseada em DAG. As arquiteturas baseadas em DAG, como o Tangle da IOTA (SILVANO; MARCELINO, 2020), permitem a validação paralela de transações, resultando em maior vazão, baixa latência e melhor escalabilidade, o que é vantajoso em aplicações de IoT.

Apesar disso, cada uma das arquiteturas apresenta vantagens e desvantagens específicas: a blockchain linear é mais consolidada e amplamente testada, porém pode apresentar baixo desempenho e alto consumo de recursos computacionais. Por outro lado, as arquiteturas baseadas em DAG oferecem maior vazão e escalabilidade para ambientes com alto volume de dados e dispositivos com recursos limitados. No entanto, estas arquiteturas ainda estão em fase de evolução e podem apresentar desafios consideráveis na garantia da segurança e do desempenho.

Em Chen *et al.* (2024), por exemplo, os autores propõem uma blockchain baseada em DAG com o objetivo de melhorar a segurança e o desempenho em sistemas de IoT Industrial, utilizando criptografia leve para dispositivos com recursos limitados e um sistema de gerenciamento de autenticação e autorização descentralizados que aproveita a vazão e a baixa latência inerentes à DAG.

Entretanto, segundo Hellani *et al.* (2021), blockchains para IoT enfrentam limitações devido à estrutura linear e ao alto consumo de recursos. Em resposta a esse desafio, os autores combinam blockchain tradicional no *backend* com o Tangle (baseada em DAG) no *frontend*,

para permitir transações escaláveis.

As novas soluções de blockchain baseadas em DAG precisam ser adaptadas às necessidades de processamento de dados em tempo real das aplicações IoT. Diante disso, Liao *et al.* (2024) apresentam o RT-DAG (*Real-time DAG*), um protótipo de blockchain baseado em DAG que suporta transações em tempo real por meio de atribuições de prioridades, controle de concorrência e estratégia de balanceamento de carga.

Blockchains baseadas em DAG também têm sido propostas para domínios de aplicação específicos. Zhang *et al.* (2023) exploram o uso de uma blockchain privada baseada em DAG para o setor de *Smart* Energia. Nesta solução, o registro paralelo de dados da blockchain baseada em DAG suporta o envio simultâneo de transações provenientes de um grande número de dispositivos conectados. Contudo, a escalabilidade em cenários maiores ainda requer mais experimentação.

Em outro domínio de aplicação, Yang *et al.* (2020b) desenvolveram o LDV (*Lightweight DAG for Vehicles*), uma blockchain leve baseada em DAG para IoT Veicular. A solução proposta busca reduzir o volume de dados registrados, fazendo com que cada nó armazene apenas os dados mais relevantes. Com isso, o consenso se torna mais leve, economizando recursos como energia. Porém, não são apresentados detalhes do processo de consenso e nem uma análise da segurança da LDV.

Fei *et al.* (2021) propõem uma blockchain baseada em DAG para rastreabilidade de produtos industriais, e apresentam um esquema de autenticação próprio. Apesar dos testes de segurança preliminares, os autores reconhecem a necessidade de avaliação de desempenho em ambientes IIoT de larga escala.

Por sua vez, Cullen *et al.* (2020), usam uma blockchain baseada em DAG e concentram-se nas vulnerabilidades de segurança da IOTA Tangle. Os autores analisam um cenário de ataque de gasto duplo (KUMAR *et al.*, 2023), para interromper a imutabilidade dos registros. O trabalho estuda as vulnerabilidades da IOTA e apresenta uma extensão do algoritmo de consenso para melhorar a segurança contra ataques de gasto duplo na blockchain.

Em resposta aos desafios de confiabilidade, Ilakkiya e Rajaram (2024) desenvolveram uma blockchain baseada em DAG tolerante a falhas. A topologia da rede é separada em *clusters*, onde é utilizado um método, com vários critérios para transferência segura de dados. O método sugerido mostrou maior eficiência na taxa de entrega de pacotes, redução do tempo de confirmação de transações e tolerância a falhas.

Alguns trabalhos também usam blockchains baseadas em DAG com novas propostas de

algoritmos de consenso. Fu *et al.* (2022) apresentam o Teegraph, um algoritmo de consenso que supera o Hashgraph (BAIRD, 2016), um dos mais populares em DAGs atualmente, em economia de energia e recursos computacionais. No entanto, o Teegraph ainda necessita de avaliação experimental em ambientes IoT mais realistas.

De modo semelhante, Zheng *et al.* (2024) propõem o S-DAG (*Sharded DAG*), uma arquitetura de dupla camada que processa transações na blockchain e armazena cabeçalhos na DAG. S-DAG usa um balanceamento de carga adaptativo para o gerenciamento periódico da rede, e a estrutura de bloco árvore 3D-Merkle para consultas tridimensionais. Os experimentos demonstraram o desempenho superior da solução proposta em comparação à soluções tradicionais.

Ainda tratando de algoritmos de consenso, Ren e Chen (2023) propuseram uma blockchain baseada em DAG para IIoT (do Inglês *Industrial IoT*) e um novo algoritmo de consenso. Nesta proposta, a coleta de dados e o envio de novos blocos para a blockchain podem ser realizados simultaneamente, pois a autenticidade dos dados é verificada pelo algoritmo de consenso desenvolvido e o novo bloco só pode ser confirmado após atender a todas as condições de validação.

Finalmente, para Li, Huang e Zhang (2022), a estrutura de cadeia única da blockchain tradicional limita a taxa de vazão da IoT. Diante disso, os autores propuseram uma blockchain baseada em DAG, onde é definida uma nova regra que usa pesos para a seleção dos pais de um novo bloco. Os autores ainda propuseram o *Tree-Based Gossip Protocol* (Protocolo de Conversa Baseado em Árvore – TBGP), um algoritmo de consenso baseado em árvore, que reduz a redundância de comunicação e melhora a vazão.

Como contraponto às abordagens puramente paralelas, Qu *et al.* (2024) desenvolveram o TidyBlock, um algoritmo que organiza transações em blocos dentro de uma DAG para facilitar a verificação e otimizar a análise de dados em IoT.

Por fim, Chen *et al.* (2024) retornam ao tema de tolerância a falhas através do desenvolvimento do algoritmo de consenso chamado DAG-D (*Double-DAG*), que oferece um protocolo de autenticação para evitar ataques maliciosos. Além disso, os autores propuseram um método para construir os nós da DAG e aumentar a taxa de transferência durante o registro de transações.

Por outro lado, Wang *et al.* (2021) apontam que, embora as DAGs sejam promissoras, falta uma compreensão aprofundada de suas características. Por isso, os autores desenvolveram um conjunto de testes de *benchmark* em uma rede IOTA privada, analisando suas características

em três aspectos: desempenho; segurança; e consistência. Os testes realizados mostraram que as DAGs podem apresentar diversos gargalos, como em operações de consultas, o que limita a vazão e aumenta a latência. Com isso, há a necessidade de otimizações em operações de inserção e consultas na DAG.

3.4 USO DE ALGORITMOS GENÉTICOS EM BLOCKCHAINS PARA IOT

Os Algoritmos Genéticos (AGs) podem ser utilizados como uma ferramenta para otimizar diversos aspectos em uma blockchain para IoT: a eficiência na gestão de recursos; a escolha de algoritmos de consenso; e a melhoria do desempenho da rede. Com isso, os AGs podem tornar a tecnologia Blockchain mais adequada ao uso em cenários da IoT.

Seguindo essa perspectiva, Babu *et al.* (2024) propõem uma estratégia de otimização de consumo de energia e criação de nós, usando AGs e programação linear (PL). A otimização baseada em AG concentra-se em melhorar o consumo de energia pelos nós na blockchain, enquanto a criação de nós via PL busca aprimorar o processo de consenso através da seleção de um conjunto de nós confiáveis, visando melhorar a privacidade de dados e o desempenho da rede.

Diferentemente de Babu et al. (2024), que focam no consumo de energia e na criação de nós, Guruprakash e Koppu (2020) propõem melhorias no processo de criptografia, transmissão e validação de transações. Para isso, os autores adicionam uma camada de criptografia baseada no método *Elliptic Curve ElGamal* (EC-ElGamal) (SUTIKNO; SURYA; EFFENDI, 1998), e utilizam um AG para gerar chaves SHA-384. Estas chaves são usadas na geração de *hashs* para reduzir o tempo de processamento de transações e acelerar a validação de blocos.

Diferentemente de Guruprakash e Koppu (2020), que aplicam AGs na geração de chaves, Abbas *et al.* (2021) orientam-se para a detecção de nós maliciosos internos e para a otimização de rotas entre os nós da blockchain. Para isso, os autores propõem um algoritmo de autenticação leve baseado em blockchain onde as credenciais dos sensores são armazenadas. O cálculo das rotas e a verificação da existência de nós maliciosos é realizado através de um AG, que mantém uma lista de nós maliciosos na blockchain.

Ampliando o escopo das aplicações de AG em blockchain, Abidallah *et al.* (2025) investigaram o uso de AGs para reduzir o consumo de energia, aumentar a velocidade de processamento de transações e reforçar a integridade de dados. Além de apontar áreas promissoras como saúde, gestão de energia e cadeias de suprimentos, os autores discutem desafios do

custo computacional e escalabilidade, sugerindo futuras combinações com IA e algoritmos de consenso adaptativos.

Por fim, Alofi *et al.* (2022) analisam o consumo de energia provocado por algoritmos de consenso, como o PoW, em blockchains para IoT. Para mitigar esse problema, eles apresentam uma estratégia para a seleção de mineradores utilizando AGs. A solução proposta adapta as taxas de cruzamento e mutação para gerar conjuntos de mineradores que minimizem o consumo de energia sem comprometer a integridade do consenso. O trabalho demonstra, através de experimentos, que o uso de AG reduz o consumo de energia em comparação a métodos tradicionais, ao mesmo tempo que mantém níveis satisfatórios de segurança na distribuição de poder de mineração.

3.5 ANÁLISE COMPARATIVA

Quatro critérios foram usados para comparação dos trabalhos relacionados com a proposta apresentada neste tese: se o trabalho usa blockchain baseada em DAG; se há utilização de algoritmos genéticos; se há combinação entre diferentes algoritmos de consenso; e se são apresentadas estratégias de escolhas de algoritmos de consenso. A Tabela 3.2 apresenta a análise comparativa dos trabalhos relacionados.

Considerando estes quatro critérios, alguns trabalhos analisados realizam combinações entre diferentes algoritmos de consenso, de forma semelhante ao que foi implementado nesta tese. Por exemplo, a integração entre os algoritmos PoS e PoW (BAI; XIA; FU, 2019); PoS e PBFT (TAPWAL *et al.*, 2021); PoW e PBFT (ZHIPENG, 2019) e PoA e dPoS (RESHI; SHOLLA, 2025).

Outros trabalhos fazem uso de algoritmos genéticos (ABBAS *et al.*, 2021), (ALOFI *et al.*, 2022), (GURUPRAKASH; KOPPU, 2020), (ABIDALLAH *et al.*, 2025). O uso de AGs permite melhorar diversos aspectos da blockchain, como reduzir o consumo de energia, o tempo de confirmação de transações e o consumo de recursos computacionais.

Da mesma forma, algumas propostas desenvolvem estratégias de escolha utilizando algoritmos genéticos (ABBAS *et al.*, 2021), (ALOFI *et al.*, 2022), (BABU *et al.*, 2024). Essas estratégia permitem escolher o minerador mais adequado para o processo de consenso, ou as melhores rotas para transmissão dos dados, ou ainda o algoritmo de consenso mais eficiente para a validação de um novo bloco.

No entanto, como é possível observar na Tabela 3.2 nenhum dos trabalhos analisados

abrange os quatro critérios considerados. Em particular, *OmniBlock* é baseada em DAG, suporta múltiplos algoritmos de consenso, ao mesmo tempo que usa AGs para a escolha dos algoritmos de consenso a serem utilizados.

No entanto, como mostra a Tabela 3.2, nenhum dos trabalhos analisados abrange simultaneamente os quatro critérios definidos. A *OmniBlock*, por sua vez, reúne essas características: sua estrutura baseada em DAG proporciona um rápido processamento de transações; o suporte a múltiplos algoritmos de consenso oferece flexibilidade na escolha dos algoritmos mais adequados a cada cenário; e o emprego de algoritmos genéticos para seleção dinâmica de combinações entre algoritmos de consenso possibilita otimizar o tempo de validação de blocos e o consumo de recursos computacionais. Essas combinações de características tornam a *OmniBlock* promissora para ambientes IoT com recursos restritos.

Diferentemente de outros trabalhos, que adotam um único algoritmo de consenso ou combinações pontuais para um domínio específico, a *OmniBlock* suporta múltiplas combinações de algoritmos de consenso (por exemplo, combinações como PoS-PBFT, PoA-Raft, PoB-PoL, dentre outras), o que possibilita adaptar a configuração de consenso às exigências heterogêneas dos diversos domínios da IoT, para atender requisitos de latência, escalabilidade e velocidade de confirmação de novas transações.

Para automatizar o processo de escolha dos algoritmos de consenso e suas combinações, empregam-se dois algoritmos genéticos que exploram o conjunto de algoritmos disponíveis e otimizam os *trade-offs* entre tempo de criação de novos blocos e consumo de recursos computacionais. Isso permite a utilização de uma estratégia leve e efetiva para selecionar, em tempo operacional, integrações de consenso que melhor atendam aos requisitos do cenário em execução.

Tabela 3.2 – Análise comparativa entre os trabalhos relacionados.

	Uso de blockchain baseada em DAG	Uso de AGs em blockchain	Combinação de Algoritmos de Consenso	Estratégia de escolha usando AGs
(BAI; XIA; FU, 2019)			✓	
(LUNARDI *et al.*, 2019)			✓	
(XU *et al.*, 2019)			✓	
(ZHIPENG, 2019)			✓	
(WANG; LI; LI, 2020)			✓	
(CULLEN *et al.*, 2020)	✓			
(GURUPRAKASH; KOPPU, 2020)		✓		
(RASOLROVEICY; FOKAEFS, 2020)			✓	
(YANG *et al.*, 2020b)	✓			
(FU *et al.*, 2021)			✓	
(HELLANI *et al.*, 2021)	✓			
(ABBAS *et al.*, 2021)		✓		✓
(FEI *et al.*, 2021)	✓			
(TAPWAL *et al.*, 2021)			✓	
(WANG *et al.*, 2021)	✓			
(ALRUBEI; BALL; RIGELSFORD, 2022)			✓	
(FU *et al.*, 2022)	✓			
(LI; HUANG; ZHANG, 2022)	✓			
(ALOFI *et al.*, 2022)		✓		✓
(HUANG; YANG; AJAY, 2023)			✓	
(REN; CHEN, 2023)	✓			
(LUO; YU; LUO, 2023)			✓	
(ZHANG *et al.*, 2023)	✓			
(YUSOF; ZAHILAH; OTHMAN, 2024)			✓	
(UVEISE; FATHIMA, 2024)			✓	
(CHEN *et al.*, 2024)	✓			
(LIAO *et al.*, 2024)	✓			
(ILAKKIYA; RAJARAM, 2024)	✓			
(ZHENG *et al.*, 2024)	✓			
(QU *et al.*, 2024)	✓			
(CHEN *et al.*, 2024)	✓			
(BABU *et al.*, 2024)		✓		✓
(RESHI; SHOLLA, 2025)			✓	
(ABIDALLAH *et al.*, 2025)		✓		
OmniBlock	✓	✓	✓	✓

Legenda: AGs: Algoritmos Genéticos

Fonte: O autor (2025).

3.6 CONSIDERAÇÕES FINAIS

Este capítulo apresentou os principais trabalhos relacionados a esta tese. Na primeira parte, foram analisados trabalhos relacionados à combinação de algoritmos de consenso em blockchain para IoT. Em seguida, o capítulo apresentou blockchains baseadas em DAG para o registro de dados da IoT. Por fim, na terceira parte da pesquisa bibliográfica, foram coletados e analisados trabalhos referentes ao uso de algoritmos genéticos em blockchain para a Internet das Coisas. No próximo capítulo, será descrita a blockchain proposta nesta tese.

4 OMNIBLOCK: UMA BLOCKCHAIN BASEADA EM DAG COM MÚLTIPLOS ALGORITMOS DE CONSENSO

Este capítulo detalha a blockchain proposta nesta tese: *OmniBlock*. Inicialmente, a Seção 4.1 apresenta uma visão geral das principais características da blockchain, e sua implementação utilizando DAG. Em seguida, na Seção 4.2, são apresentadas e explicadas as combinações dos algoritmos de consenso utilizados. Finalmente, a Seção 4.3 descreve a estratégia de seleção destas combinações usando algoritmos genéticos.

4.1 VISÃO GERAL

A blockchain *OmniBlock* foi desenvolvida com o objetivo de melhorar o desempenho do registro de dados de dispositivos na Internet das Coisas. A *OmniBlock* busca superar as limitações das blockchains tradicionais (por exemplo, a alta demanda de recursos computacionais e longo tempo de confirmação de blocos), oferecendo confirmação mais rápida de blocos e menor consumo de recursos, como CPU e memória.

A *OmniBlock* utiliza *Directed Acyclic Graph* (DAG) e suporta múltiplos algoritmos de consenso como estratégia para enfrentar os desafios de desempenho e eficiência no consumo de recursos computacionais. A adoção de uma estrutura baseada em DAG permite uma validação de transações mais rápida, diferentemente das estruturas lineares tradicionais, possibilitando maior paralelização das transações e reduzindo a latência na sua confirmação (WANG *et al.*, 2022).

A escolha de uma arquitetura baseada em DAG surge da necessidade de abordar os desafios apresentados pelas aplicações em ambientes IoT, como a necessidade de processamento rápido de transações e economia de recursos. A proposta de combinação de múltiplos algoritmos de consenso em *OmniBlock* é motivada pela busca de um equilíbrio entre segurança, escalabilidade e descentralização (BUTERIN *et al.*, 2014). Cada algoritmo contribui com suas características mais fortes, criando uma solução que aborda vulnerabilidades individuais e melhora o desempenho no registro dos dados.

4.1.1 Projeto da *OmniBlock*

Os principais objetivos considerados na concepção e implementação da *OmniBlock* são apresentados a seguir:

- **Diminuir o tempo de processamento de transações**: a *OmniBlock* busca reduzir o tempo de confirmação de transações, através do processamento em paralelo de transações da DAG, mantendo baixa latência mesmo em ambientes com grande volume de dados gerados por dispositivos IoT;
- **Combinar vários algoritmos de consenso**: a combinação de algoritmos como *Proof of Authority* (PoA), *Proof of Work* (PoW), *Proof of Stake* (PoS) e *Practical Byzantine Fault Tolerance* (PBFT) busca combinar as vantagens individuais de cada um para melhorar a desempenho e a eficiência da blockchain;
- **Reduzir o consumo de CPU e memória**: o uso de combinações de algoritmos permite comparar qual combinação oferece o consumo mais eficiente de recursos como CPU e memória; e
- **Reduzir o tráfego de rede**: a utilização da *OmniBlock* busca reduzir o tráfego para evitar a sobrecarga da rede.

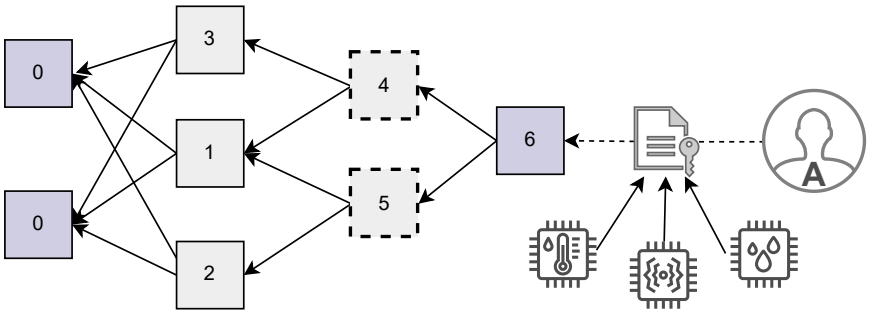
4.1.2 Desenvolvimento da *OmniBlock*

A Figura 4.1 mostra o funcionamento da *OmniBlock*, onde um usuário *A* deseja registrar dados de sensores IoT. Para isso, deve ser criado um novo bloco de dados, representado pelo número 6. Neste ponto, a blockchain escolherá qual combinação de algoritmos de consenso utilizar. Esta escolha depende de qual domínio (e.g., *Smart Home*, *Smart City*) da IoT esses dados são provenientes.

A *OmniBlock* foi prototipada utilizando contêineres Docker, onde cada contêiner representa um nó da blockchain, que armazena uma cópia de todos os dados registrados.

A estrutura base sobre a qual a *OmniBlock* foi implementada é inspirada na arquitetura Tangle, da IOTA (SILVANO; MARCELINO, 2020). Nessa estrutura, cada novo bloco deve validar 2 anteriores antes de ser inserido. Isso permite que, quanto mais transações forem adicionadas,

Figura 4.1 – Funcionamento da blockchain baseada em DAG utilizada nesta tese.



Fonte: O autor (2023).

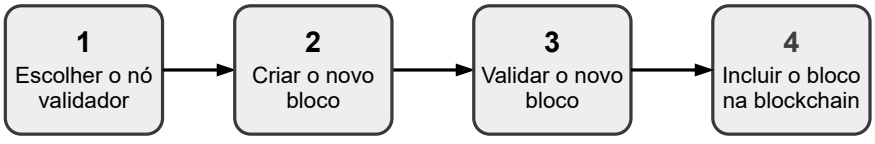
mais verificações são feitas nos registros passados, o que contribui para melhorar a imutabilidade dos dados. Por exemplo, na Figura 4.1, o nó 6, ao ser criado, deve validar os nós 4 e 5 antes de ser confirmado.

As principais ferramentas utilizadas para o desenvolvimento da *OmniBlock*, das combinações de algoritmos de consenso e dos algoritmos genéticos, são descritas no Apêndice E.

4.1.3 Processo de consenso

O processo de consenso pode ser compreendido em 4 etapas distintas, conforme mostrado em Figura 4.2. A primeira etapa consiste em escolher o nó validador, aquele que será utilizado para iniciar o processo de validação da transação e criar o novo bloco. A segunda etapa envolve a criação do bloco que contém os valores a serem registrados na blockchain. A terceira etapa, que não é executada explicitamente por alguns algoritmos de consenso, como o PoW, consiste em validar o novo bloco antes da sua inclusão na blockchain. A etapa final do processo de consenso consiste na inclusão e a replicação do novo bloco entre todos os participantes da blockchain.

Figura 4.2 – Etapas do processo de consenso.



Fonte: O autor (2025).

Nesta tese, as combinações de algoritmos de consenso implementadas consistem na utilização de dois algoritmos (por exemplo, combinação A-B), onde o primeiro algoritmo indicado

na combinação (A) sempre será utilizado na primeira etapa do processo de consenso, para escolha do nó validador, e o segundo (B), será utilizado nas demais etapas.

Por exemplo, ao realizar uma combinação entre os algoritmos PoA-PBFT, o PoA será utilizado na etapa 1 do consenso, ou seja, na escolha do nó validador; e o PBFT será utilizado nas demais etapas, para criar, validar e incluir o novo bloco na blockchain.

4.2 COMBINAÇÕES DE ALGORITMOS DE CONSENSO

Os algoritmos de consenso implementados na *OmniBlock* foram selecionados com base em suas características específicas e adequação aos requisitos da IoT: *Proof of Authority* (PoA), *Proof of Importance* (PoI), *Proof of Burn* (PoB), *Proof of Luck* (PoL), *Proof of Work* (PoW), *Proof of Stake* (PoS), *Delegated Proof of Stake* (dPoS), *Practical Byzantine Fault Tolerant* (PBFT), *Proof of Elapsed Time* (PoET), e *Raft*.

Os critérios utilizados para escolha dos algoritmos de consenso foram: *tempo de confirmação de blocos*; *consumo de recursos computacionais*; e *tolerância a falhas*. Estes critérios são fundamentais para conferir ao algoritmo de consenso a capacidade de atender às necessidades específicas dos dispositivos e domínios da IoT. No entanto, nem todos os algoritmos de consenso vão apresentar essas características simultaneamente, por exemplo, alguns vão ser boa tolerantes a falhas, mas apresentar um tempo de confirmação de blocos alto.

O PoA foi escolhido por sua baixa latência, permitindo validação rápida através de validadores identificados e confiáveis (HUSSEIN; SALAMA; EL-RAHMAN, 2023). O PoI foi selecionado por avaliar a relevância dos nós e incentivar o comportamento honesto dos participantes no ambiente IoT (AUHL *et al.*, 2022).

Por sua vez, a opção pelo PoB foi motivada por proporcionar uma redução do consumo de recursos computacionais na criação de blocos, uma vez que exige a queima de *tokens*, desestimulando comportamentos maliciosos e melhorando a desempenho da rede (MENON *et al.*, 2021). De forma semelhante, o PoS incentiva os participantes a usarem seus recursos financeiros de forma honesta e consciente, além de apresentar um baixo consumo de energia (KHAN; HARTOG; HU, 2022).

O algoritmo PoL foi utilizado por sua capacidade de distribuir de forma justa e aleatória a oportunidade de criação de blocos, sem demandar a alta utilização de recursos computacionais (KIM *et al.*, 2024). O dPoS foi selecionado para melhorar a escalabilidade e a velocidade do processo de consenso, delegando o processo de validação a um conjunto fixo de nós representantes

(SAAD; RADZI; OTHMAN, 2021).

O PoW foi escolhido pela sua segurança contra modificações (ÁLVAREZ; GRAMLICH; SEDLMEIR, 2024), enquanto o PoET foi considerado por buscar garantir uma seleção justa e de baixo consumo de recursos, algo desejável em ambientes IoT (MENON *et al.*, 2021).

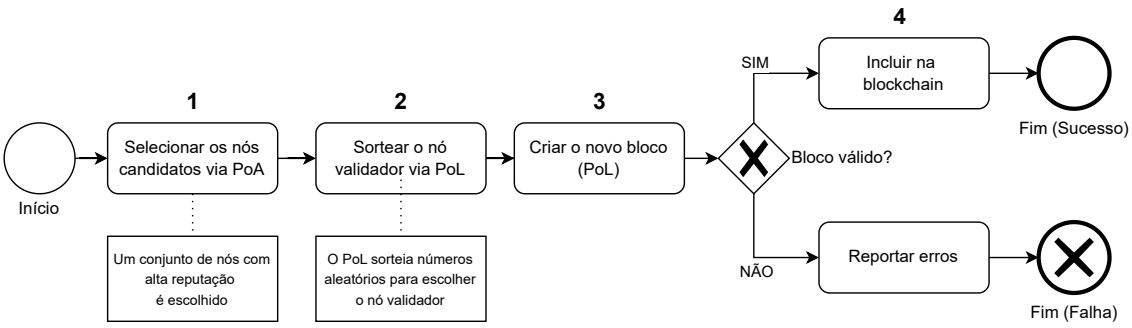
O PBFT foi selecionado por sua tolerância a falhas e segurança em redes descentralizadas (KHAN; HARTOG; HU, 2022), enquanto o *Raft* foi escolhido por sua rapidez na eleição de líderes, facilitando um consenso distribuído eficiente e garantindo alta consistência e disponibilidade (HOWARD; MORTIER, 2020).

4.2.1 *Proof of Autority* (PoA) e *Proof of Luck* (PoL)

A combinação PoA-PoL une a seleção rápida e confiável de validadores, baseada na autoridade (PoA), com o método de seleção baseada na sorte (PoL). O PoA é adotado para identificar nós confiáveis, enquanto o PoL utiliza um método aleatório de sorteio que distribui de forma equitativa a oportunidade de criação de blocos. A Figura 4.3 mostra o funcionamento desta combinação.

No processo de consenso (ver Figura 4.2), o PoA é aplicado na primeira etapa para selecionar um conjunto de nós com maior autoridade e reputação na rede (1), qualificando-os como candidatos a validadores. Em seguida, é utilizado o PoL, que realiza um sorteio entre os nós candidatos (2), o vencedor poderá realizar a criação (3), e a inclusão do novo bloco na blockchain (4).

Figura 4.3 – Funcionamento da combinação entre o *Proof of Autority* e o *Proof of Luck*.



Fonte: O autor (2025).

A principal vantagem dessa combinação é evitar a sobrecarga dos nós de maior reputação

no momento da criação dos blocos. Ao utilizar o algoritmo PoA para escolher um grupo de possíveis validadores e o PoL para uma distribuição justa das oportunidades de criação de blocos, o processo de consenso se torna menos centralizado e mais resistente a ataques e manipulações.

O Código 4.1 mostra o algoritmo de implementação da combinação PoA-PoL. Na Linha 3, é escolhido um grupo de nós com mais autoridade (PoA), nas Linhas 7-8 o PoL é utilizado para sortear um nó que será escolhido como validador do novo bloco. Na Linha 11, o nó escolhido realiza a criação do novo bloco.

Código 4.1 – Combinação do *Proof of Autority* e o *Proof of Luck*

```
1  Funcao IntegracaoPoAPoL(transacao):
2      //Passo 1: Selecionar candidatos a validacao
3      nosPoA[] ← SelecionarNosPoA()
4
5      // Passo 2: Sortear o validador
6      NodesValidadores(nosPoA):
7          numero_aleatorio ← GerarNumeroAleatorio(nosPoA)
8          validador ← nosPoA[numero_aleatorio]
9
10         // Passo 3: Criar o novo bloco
11         bloco ← CriarBlocoPoA(transacao, validador)
12
13         // Passo 5: Realizar validacao
14         Se BlocoValido(bloco):
15             PropagarBloco(bloco)
16         Senao:
17             TratarErro(transacao)
18             TratarFalhaSelecionador(transacao)
```

Fonte: O autor (2025)

Apesar das vantagens, a combinação entre PoA-PoL pode apresentar desvantagens. Por exemplo, o uso do PoA tende a concentrar a autoridade de validação em poucos nós, o que pode aumentar o risco de centralização. Além disso, a componente aleatória do PoL pode ocasionar tempos de espera inconsistentes na criação dos blocos, impactando a previsibilidade do sistema.

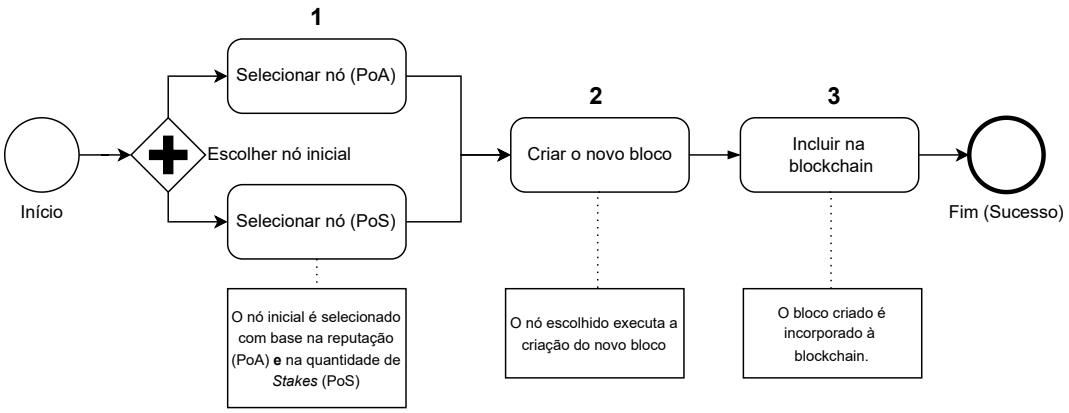
Por fim, a combinação desses dois algoritmos, e também das demais combinações, pode adicionar complexidade à implementação do processo de consenso, especialmente em ambientes com recursos limitados.

4.2.2 *Proof of Autority* (PoA) e *Proof of Stake* (PoS)

A combinação PoA-PoS baseia-se na busca pelo equilíbrio entre desempenho e descentralização. O PoA seleciona os nós de validação com base na autoridade e reputação, enquanto o PoS introduz a participação financeira dos nós, onde, para gerar um novo bloco, o participante deve possuir moedas disponíveis. A Figura 4.4 mostra o diagrama de funcionamento desta combinação.

Nesta combinação, o nó de validação inicial é selecionado entre aqueles com melhor reputação (PoA) e oferece a maior quantidade de *stakes* (1). Esta combinação tem o objetivo de não sobrecarregar o nó de maior reputação, pois outros nós podem ser escolhidos com base na sua quantidade de *stakes*. Em seguida, o nó escolhido cria o novo bloco (2), e o inclui na blockchain (3).

Figura 4.4 – Funcionamento da combinação entre o *Proof of Autority* e *Proof of Stake*.



Fonte: O autor (2025).

O PoA adota critérios de reputação para escolher nós validadores, reduzindo a latência na confirmação das transações (HUSSEIN; SALAMA; EL-RAHMAN, 2023). A necessidade de uso de *stakes* exigido pelo PoS estimula a responsabilidade e o interesse dos participantes na segurança da blockchain. O Código 4.2 mostra o algoritmo da combinação PoA-PoS. Nas Linhas 3-4 um nó validador é escolhido, em seguida um novo bloco é criado (Linha 9), e inserido na blockchain (Linha 13).

Código 4.2 – Combinação do *Proof of Autority* e *Proof of Stake*

```
1 Funcao IntegracaoPoAPoS(transacao):
2     // Passo 1: Selecao dos nos iniciais
3     noPoA = SelecionarNoPoA()
```

```
    noPoS = SelecionarNoPoS()

    Se NoPoAAutorizado(noPoA):
        Se NoPoSComStakeSuficiente(noPoS):
            // Passo 2: Criacao do novo bloco
            blocoTemporario = CriarBlocoPoW_PoS(noPoA, noPoS, transacao)
            // Passo 3: Validacao do novo bloco
            Se BlocoValido(blocoTemporario):
                // Passo 4: Propagacao do novo bloco
                PropagarBloco(blocoTemporario)
            Senao:
                // Passo 5: Tratamento de falhas
                LidarComFalhas(transacao)
        Senao:
            LidarComFaltaDeStake(transacao)
    Senao:
        LidarComFaltaDeAutoridade(transacao)
```

Fonte: O autor (2025)

A utilização de PoS pode apresentar desafios relacionados à distribuição inicial de moedas e à concentração de poder nas mãos de poucos participantes com maior poder econômico. Em ambientes IoT, onde muitos usuários não podem investir grandes quantidades de *tokens*, isso pode resultar em um número reduzido de nós validadores, comprometendo a descentralização da rede. É importante mitigar essa desvantagem por meio de políticas de distribuição que garantam maior diversidade de validadores. Portanto, é importante considerar esta potencial desvantagem ao implementar esta combinação em ambientes IoT.

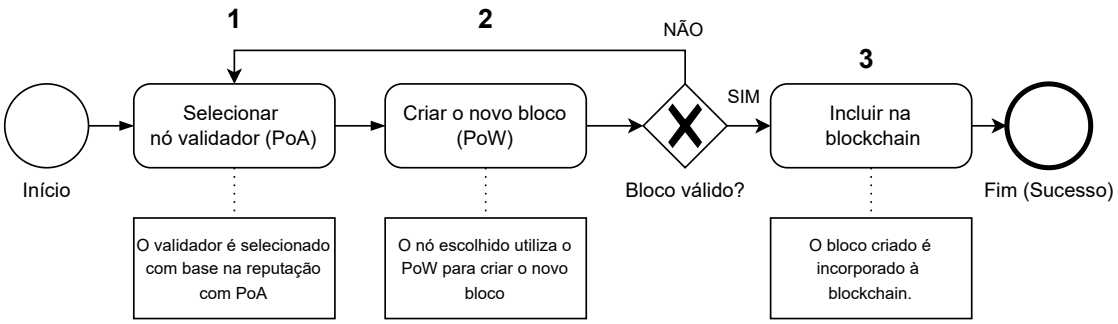
4.2.3 *Proof of Autority* (PoA) e *Proof of Work* (PoW)

Nesta combinação, PoA foi utilizado devido ao seu rápido tempo de confirmação de transações, especificamente para uso em redes privadas onde os participantes são conhecidos (HUSSEIN; SALAMA; EL-RAHMAN, 2023). O PoW foi utilizado por sua segurança e resistência a ataques (ÁLVAREZ; GRAMLICH; SEDLMEIR, 2024), que são particularmente relevantes para melhorar a imutabilidade dos dados. A Figura 4.5 apresenta o diagrama de funcionamento da combinação entre esses algoritmos.

Na primeira etapa do consenso, o algoritmo PoA seleciona o nó de validação com maior autoridade (1); na segunda etapa, o nó escolhido executa o algoritmo PoW para criar um bloco e incorporá-lo à *OmniBlock*(2). Por fim, o novo bloco é incluído na blockchain (3).

Desta forma, o PoW pode ser executado com um nível de dificuldade reduzido e ainda assim, mantém a imutabilidade dos dados, pois sua execução é realizada por um nó confiável.

Figura 4.5 – Funcionamento da combinação entre o *Proof of Autority* e *Proof of Work*.



Fonte: O autor (2023).

A combinação PoA-PoW resulta em um processo de criação de blocos mais rápido, acelerando a confirmação de transações em comparação com soluções que empregam apenas um desses algoritmos.

Aproveitando o melhor dos algoritmos, esta combinação busca otimizar o registro de dados em cenários dinâmicos e descentralizados, mantendo um bom desempenho, requisito essencial em qualquer ambiente de blockchain. O Código 4.3 mostra o algoritmo da combinação entre os algoritmos PoA-PoW. Na Linha 6 encontra-se a função de criação de um novo bloco; se o bloco criado for válido será propagado na blockchain (Linha 11).

Código 4.3 – Combinação do *Proof of Autority* e *Proof of Work*

```
1  Funcao IntegracaoPoAPoW(transacao):
2      // Passo 1: Selecionar no validador
3      noPoA = SelecionarNoPoA()
4
5      // Passo 2: Criar novo bloco
6      blocoTemporario = CriarBlocoPoW(noPoA, transacao)
7
8      // Passo 3: Validacao do novo bloco
9      Se BlocoValido(blocoTemporario):
10         // Passo 4: Propagacao do bloco
11         PropagarBloco(blocoTemporario)
12     Senao:
13         // Passo 5: Se falhar, escolher outro no
14         NovoNoPoA = SelecionarNovoNoPoA()
15         blocoFinal = CriarBlocoPoW(NovoNoPoA, transacao)
16
```

```
17          // Passo 6: Validacao do novo bloco
18          Se BlocoPoWValido(blocoFinal):
19              // Passo 7: Propagacao do bloco
20              PropagarBlocoPoW(blocoFinal)
21          Senao:
22              // Passo 8: Tratamento de falha
23              LidarComFalhas(transacao)
```

Fonte: O autor (2023)

No entanto, esta combinação pode apresentar algumas desvantagens. Por exemplo, a eficiência do PoA pode ser comprometida se participantes mal-intencionados conseguirem ganhar autoridade, enquanto o PoW pode aumentar o consumo de bateria e recursos computacionais. Portanto, é importante considerar cuidadosamente estas potenciais desvantagens ao implementar esta combinação em um ambiente IoT.

4.2.4 *Proof of Elapsed Time* (PoET) e *Proof of Autority* (PoA)

Nesta combinação, o PoET foi adotado por sua aleatoriedade na escolha de nós e a utilização de um tempo de espera para determinar a criação do bloco (KAUR *et al.*, 2021). O PoA, por sua vez, foi utilizado por apresentar uma validação de blocos rápida e confiável, baseada na autoridade dos nós (HUSSEIN; SALAMA; EL-RAHMAN, 2023). A Figura 4.6 apresenta o diagrama de funcionamento desta combinação.

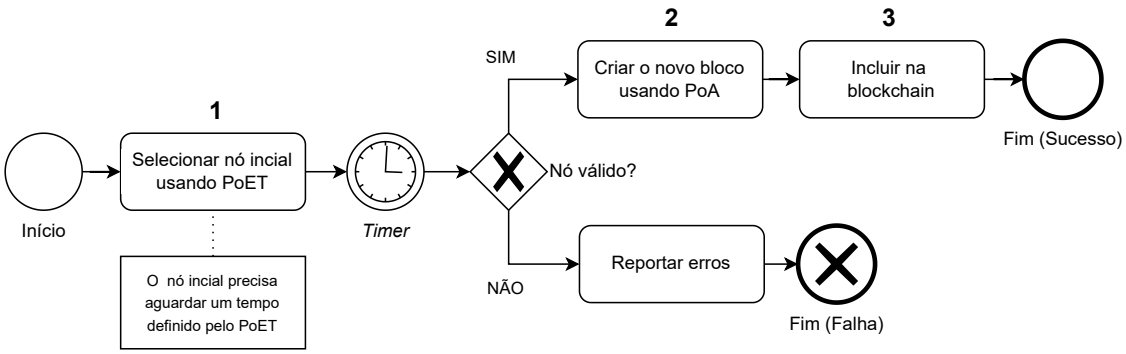
No início do processo de consenso, o PoET é utilizado para escolher o nó de validação inicial com base no sorteio de um período de espera aleatório (1). O nó que obtiver o menor tempo de espera e despertar primeiro é escolhido. Em seguida, esse nó aplica o PoA para criar o novo bloco (2), e incluí-lo na blockchain (3).

A principal vantagem dessa combinação é a redução do consumo de recursos computacionais, pois são utilizados dois algoritmos conhecidos por sua eficiência nesse quesito; e que mantêm a velocidade de criação de blocos. O Código 4.4 mostra o algoritmo da combinação PoET-PoA. Na Linha 4 é realizado o sorteio do tempo aleatório utilizado pelo PoET. Após a finalização do tempo de espera (Linha 5) o processo de consenso pode continuar.

Código 4.4 – Combinação do *Proof of Elapsed Time* e o *Proof of Autority*.

```
1 Funcao IntegracaoPoETPoA(transacao):
2     // Passo 1: Selecionar validador (PoET)
3     noElegivel ← SelecionarNoPoET()
4     tempo_espera ← SortearTempoEsperaPoET(noElegivel)
```

Figura 4.6 – Funcionamento da combinação entre o *Proof of Elapsed Time* e o *Proof of Autority*.



Fonte: O autor (2025).

```
    Aguarde(tempo_espera)

    Se NoElegivelValido(noElegivel):
        // Passo 2: Criar novo bloco
        bloco ← CriarBlocoPoA(transacao, noElegivel)

        // Passo 3: Validacao do novo bloco
        Se BlocoValido(bloco):
            // Passo 4: Propagacao do bloco na Blockchain
            PropagarBloco(bloco)
        Senao:
            // Passo 5: Tratamento de erros
            TratarErro(transacao)
    Senao:
         Retornar ErroDeSelecao(transacao)
```

Fonte: O autor (2025)

Apesar de oferecer uma seleção justa e eficiente de nós, a combinação entre os algoritmos PoET-PoA apresenta algumas limitações. O uso do PoET em cenários reais requer hardwares específicos, como módulos Intel SGX[1], o que pode restringir a confiabilidade em ambientes que não há a disponibilidade desses hardwares.

Além disso, o uso do PoA tende a centralizar a autoridade em poucos nós, o que pode comprometer a descentralização do sistema e criar pontos de falha. Por fim, a aleatoriedade inerente ao PoET pode ocasionar variações imprevisíveis nos tempos de criação dos blocos, afetando o desempenho da blockchain em cenários de alta demanda.

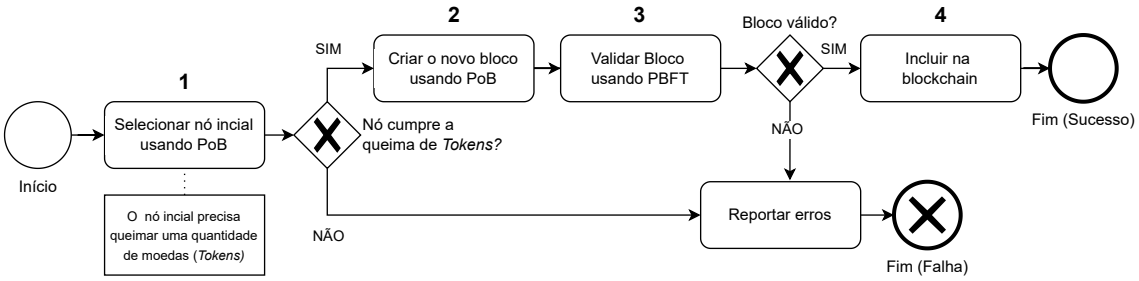
[1] SCX - *Software Guard Extensions*. Mais informações: https://www.intel.com.br/software-guard-extensions

4.2.5 *Proof of Burn* (PoB) e *Practical Byzantine Fault Tolerance* (PBFT)

Nesta combinação, o PoB foi escolhido por oferecer economia de recursos computacionais ao processo de consenso. Por sua vez, o PBFT apresenta um consenso tolerante a falhas em ambientes distribuídos (KHAN; HARTOG; HU, 2022). Enquanto o PoB incentiva o comprometimento dos validadores através da queima de *tokens*, o PBFT garante que o bloco seja validado por um conjunto de nós confiáveis. A Figura 4.7 apresenta o diagrama de funcionamento desta combinação.

Na primeira etapa do processo de consenso, o PoB é utilizado para escolher um candidato a validador (1), o nó escolhido precisa queimar uma quantidade pré-determinada de *tokens*. Em seguida, o novo bloco é gerado (2), e passa por um processo de validação utilizando o PBFT (3), no qual os demais nós participantes realizam verificações até atingir o consenso e incluir o bloco na blockchain (4).

Figura 4.7 – Funcionamento da combinação entre o *Proof of Burn* e o *Practical Byzantine Fault Tolerance*.



Fonte: O autor (2025).

A vantagem dessa combinação está na melhora da segurança, pois o PoB desestimula comportamentos maliciosos ao impor um custo econômico para a criação de blocos, enquanto o PBFT assegura uma validação rápida e confiável mesmo na presença de falhas ou nós comprometidos (KHAN; HARTOG; HU, 2022). O Código 4.5 mostra o algoritmo da combinação PoB-PBFT. Após a criação do novo bloco pelo PoB (Linha 7), o PBFT é utilizado para validar e propagar o bloco na blockchain (Linhas 10-12).

Código 4.5 – Combinação do *Proof of Burn* e o *Practical Byzantine Fault Tolerance*.

```
1 Funcao IntegracaoPoB_PBFT(transacao):
2     // Passo 1: Selecionar no inicial (PoB)
3     noCandidato ← SelecionarNoPoB()
4
```

```
5       Se NoComprometido(noCandidato):
6           //Passo 2: Criar bloco
7           bloco ← CriarBlocoPoB(transacao, noCandidato)
8
9           // Passo 3: Validacao usando PBFT
10          Se PBFTConsensoAtingido(bloco):
11              // Passo 4: Propagar o bloco
12              PropagarBloco(bloco)
13          Senao:
14              // Passo 5: Tratar falhas
15              TratarFalhaPBFT(transacao)
16      Senao:
17           TratarFalhaSelecao(transacao)
```

Fonte: O autor (2025)

Embora a combinação entre os algoritmos PoB-PBFT possa melhorar a segurança ao exigir o comprometimento econômico por meio da queima de *tokens* e utilizar o consenso PBFT, ela também apresenta algumas limitações. O custo associado à queima de *tokens* pode representar uma barreira para a participação de nós que não dispõem de recursos financeiros. Ao mesmo tempo, o algoritmo PBFT pode dificultar a escalabilidade, aumentando a latência e a complexidade do consenso em ambientes com muitos nós. Dessa forma, a combinação entre PoB-PBFT pode ser menos adequada para cenários com grande quantidade de dispositivos e tráfego intenso, onde a escalabilidade e a minimização de custos computacionais são prioridades.

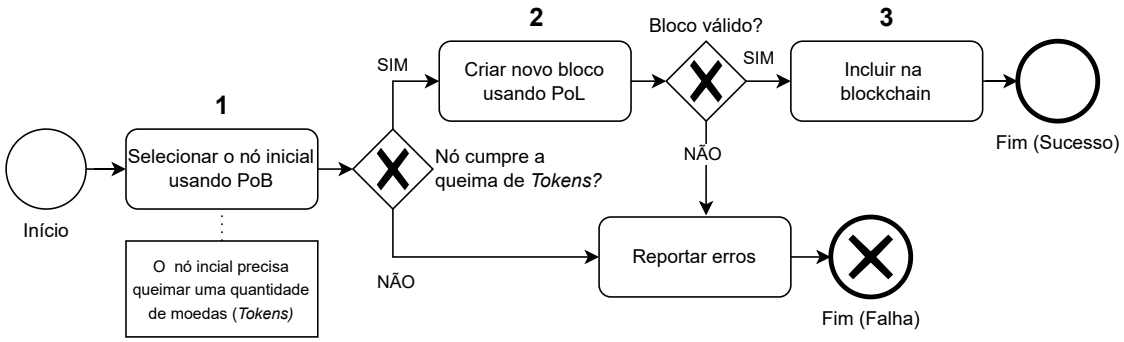
4.2.6 *Proof of Burn* (PoB) e *Proof of Luck* (PoL)

O uso do PoB proporciona compromisso econômico ao exigir que os nós queimem moedas para demonstrar seu comprometimento com a rede, enquanto o PoL utiliza uma estratégia de sorte para distribuir de forma equitativa a oportunidade de criação de blocos. A Figura 4.8 apresenta o diagrama de funcionamento desta combinação.

No processo de consenso, o PoB é aplicado inicialmente para selecionar o nó validador (1), ou seja, aquele que realiza a queima de *tokens* e, assim, se qualifica para participar da validação. Em seguida, esse nó utiliza o PoL para criar um novo bloco (2), em seguida o bloco criado é incluído na blockchain (3).

Esta combinação busca aliar segurança e descentralização: o PoB desestimula comportamentos maliciosos ao impor um custo econômico significativo, enquanto o PoL proporciona

Figura 4.8 – Funcionamento da combinação entre o *Proof of Burn* e *Proof of Luck*



Fonte: O autor (2025).

um processo de criação de blocos menos intensivo em recursos e distribuído de maneira justa entre os participantes (KIM *et al.*, 2024).

O Código 4.6 mostra o algoritmo da combinação entre o algoritmos de consenso PoB-PoL. Se o nó inicial escolhido for válido, ele poderá criar o novo bloco (Linha 11), e o propagar na blockchain (Linha 15).

Código 4.6 – Combinação do *Proof of Burn* e *Proof of Luck*

```
1  Funcao IntegracaoPoB_PoL(transacao):
2      // Passo 1: Selecionar nos candidatos
3      nosCandidatos ← SelecionarNosPoB()
4
5      Se NosComprometidos(nosCandidatos):
6          // Passo 2: Sortear validador (PoL)
7          noValidador ← SortearNoValidadorPoL(nosCandidatos)
8
9          Se NoValido(no_Validador):
10             //Passo 3: Criar bloco
11             bloco ← CriarBloco(transacao, noValidador)
12
13             // Passo 4: Validar e propagar o bloco
14             Se BlocoValido(bloco):
15                 PropagarBloco(bloco)
16             Senao:
17                 // Passo 5: Tratar erros
18                 TratarErroValidacao(bloco)
19         Senao:
20             TratarErroPoL(transacao)
21     Senao:
22         TratarErroPoB(transacao)
```

Fonte: O autor (2025)

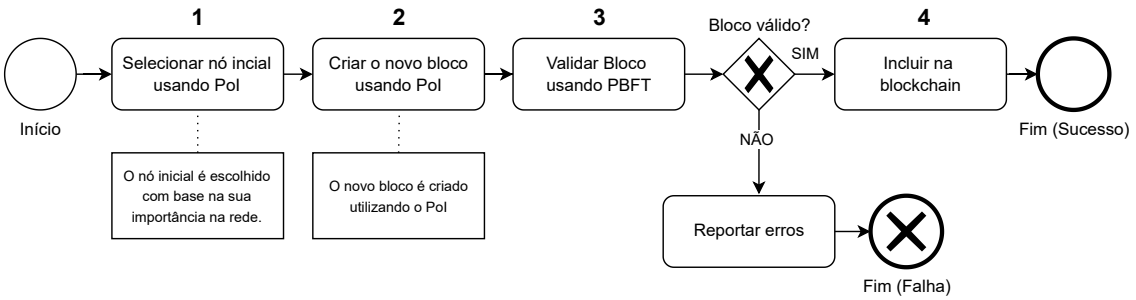
Apesar de oferecer vantagens na segurança e descentralização, a combinação entre PoB-PoL pode apresentar limitações. A queima de *tokens* no PoB pode ser onerosa, impactando a liquidez e desencorajando a participação, enquanto o mecanismo aleatório do PoL pode gerar tempos de espera imprevisíveis na criação de blocos. Além disso, a combinação dos dois pode aumentar a complexidade da implementação e do ajuste de parâmetros, dificultando a escalabilidade em redes de grande porte.

4.2.7 *Proof of Importance* (PoI) e *Practical Byzantine Fault Tolerance* (PBFT)

Esta combinação realiza a seleção de validadores baseada na importância do nós, através do PoI. Este algoritmo avalia a relevância dos nós a partir de métricas como *stakes*, atividade e interação na rede. Enquanto o PoI identifica nós com maior comprometimento e desempenho, o PBFT oferece um consenso eficiente, mesmo na presença de nós maliciosos. A Figura 4.9 apresenta o diagrama que representa o funcionamento desta combinação.

Na primeira etapa do processo de consenso, o PoI é aplicado para selecionar o nó com maior importância, qualificando-o como validador (1). Em seguida, esse nó cria o novo bloco (2), e utiliza o PBFT para validação (3), garantindo que um grupo de nós confiáveis atinja consenso antes da inclusão final do bloco na blockchain (4).

Figura 4.9 – Funcionamento da combinação entre o *Proof of Importance* e *Practical Byzantine Fault Tolerance*.



Fonte: O autor (2025).

A principal vantagem dessa integração é a combinação de uma seleção de validadores baseada em desempenho (PoI) com um algoritmo de consenso resiliente (PBFT), que aumenta a segurança e a descentralização da rede (KHAN; HARTOG; HU, 2022). A combinação entre

esses algoritmos pode ser adequada para ambientes da Internet das Coisas que exigem alta confiabilidade e desempenho.

O Código 4.7 mostra o algoritmo da combinação entre os algoritmos PoI e PBFT. O processo de consenso verifica se o nó escolhido inicialmente atende aos requisitos de desempenho (Linha 7). O bloco criado é validado e propagado através do PBFT entre os demais nós da blockchain (Linhas 12-16).

Código 4.7 – Combinação do *Proof of Importance* e *Practical Byzantine Fault Tolerance*

```
1  Funcao IntegracaoPoI_PBFT(transacao):
2
3      // Passo 1: Selecionar no inicial (PoI)
4      noCandidato ← SelecionarNoPoI()
5
6      // Passo 2: Verificar se o candidato e valido
7      Se NoImportante(noCandidato):
8          // Passo 3: Criar novo bloco
9          bloco ← CriarBloco(transacao, nodoCandidato)
10
11         // Passo 4: Validar bloco (PBFT)
12         consenso ← IniciarPBFT(bloco)
13
14         Se (consenso):
15             // Passo 5: Propagar o bloco (PBFT)
16             PropagarBloco(bloco)
17         Senao:
18              TratarErroPBFT(transacao)
19     Senao:
20         TratarErroSelecao(transacao)
```

Fonte: O autor (2025)

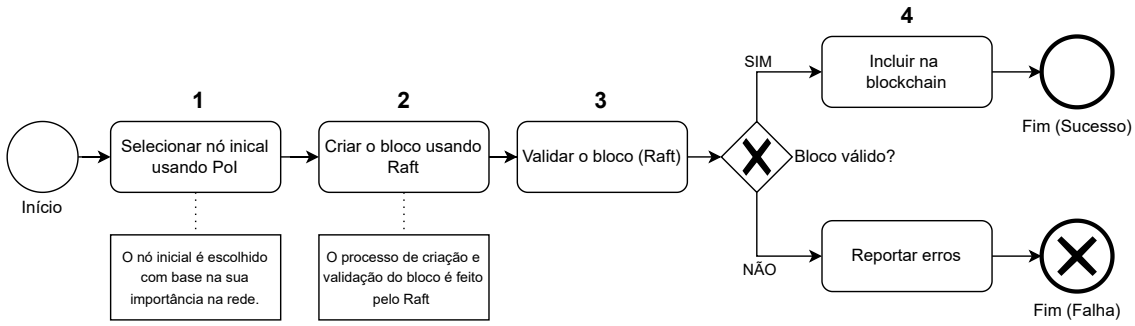
Apesar de oferecer uma abordagem tolerante à falhas, a combinação PoI-PBFT apresenta limitações. O PoI depende de indicadores de reputação, que podem ser manipulados, ou não refletir com precisão a verdadeira relevância dos nós, afetando a seleção dos validadores. Além disso, o PBFT, embora eficiente em redes menores, enfrenta desafios de escalabilidade e aumento de latência em ambientes com muitos participantes, o que pode degradar o desempenho do sistema em larga escala.

4.2.8 *Proof of Importance* (PoI) e Raft

A combinação entre os algoritmos PoI-Raft integra a seleção criteriosa de validadores baseada na importância (PoI), com a agilidade e eficiência do Raft, que realiza um processo de consenso distribuído e tolerância a falhas (HOWARD; MORTIER, 2020). A Figura 4.10 apresenta o diagrama desta combinação.

No processo de consenso, o PoI é utilizado inicialmente para identificar e selecionar o nó com melhor desempenho, atribuindo-lhe a responsabilidade de propor o novo bloco (1). Em seguida, o nó escolhido utiliza o Raft para liderar o consenso, gerando (2), validando (3), e inserindo o bloco na blockchain (4), mesmo diante de eventuais falhas, como, por exemplo, nós indisponíveis e atrasos na rede.

Figura 4.10 – Funcionamento da combinação entre o *Proof of Importance* e o Raft.



Fonte: O autor (2025).

Essa combinação resulta em uma seleção eficiente e orientada por métricas de importância (PoI) com um algoritmo de consenso ágil e confiável (Raft). Assim, há um ganho de descentralização e escalabilidade, características essenciais para ambientes IoT que exigem eficiência na gestão dos dados.

O Código 4.8 mostra o algoritmo da combinação entre PoI e Raft. Se o nó inicial for válido (Linha 6), o algoritmo Raft poderá ser utilizado para validar e propagar o novo bloco na blockchain (Linhas 11-13).

Código 4.8 – Combinação do *Proof of Importance* e o Raft

```
1 Funcao IntegracaoPoI_Raft(transacao):
2     // Passo 1: Selecionar no inicial (PoI)
3     noCandidato ← SelecionarNoPoI()
4
```

```
    // Passo 2: Verificar se candidato e valido
    Se NoImportante(noCandidato):
        // Passo 3: Iniciar o Raft para criar novo bloco
        bloco ← IniciarRaft(transacao, noCandidato)

        // Passo 4: Validar o bloco
        Se BlocoValido(bloco):
            // Passo 5: Propagar o bloco
            PropagarBloco(bloco)
        Senao:
            // Tratamento de erros
            TratarErroRaft(transacao)
    Senao:
         TratarErroSelecao(transacao)
```

Fonte: O autor (2025)

Embora a combinação PoI-Raft apresente pontos fortes, como agilidade na confirmação de blocos, existem também limitações. A verificação da importância dos nós pode ser complexa e suscetível a manipulações, o que pode comprometer a justiça na seleção. Além disso, o Raft, embora eficiente em redes de tamanho moderado, pode enfrentar problemas de segurança e aumento de latência quando aplicado em grandes redes IoT, devido ao *overhead* de comunicação entre os nós (HOWARD; MORTIER, 2020).

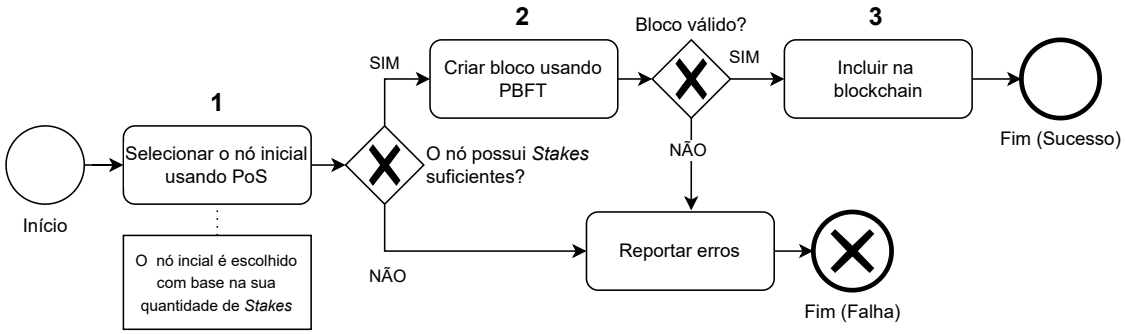
4.2.9 *Proof of Stake* (PoS) e *Practical Byzantine Fault Tolerance* (PBFT)

O algoritmo PoS apresenta um consumo eficiente de energia em comparação com outros algoritmos como o PoW (KHAN; HARTOG; HU, 2022). Além disso, o uso de PoS limita a capacidade de criar novos blocos à quantidade de moeda mantida pelos participantes, incentivando um comportamento honesto. O PBFT contribui com a capacidade de tolerar falhas e participantes maliciosos. A Figura 4.11 descreve o funcionamento desta combinação.

Esta combinação busca tornar o processo de escolha do nó líder do PBFT mais seguro. Portanto, no início do processo de consenso, é utilizado o PoS para escolher o nó inicial entre aqueles que oferecem a maior quantidade de *stakes* (1). Em seguida, o nó escolhido executa o PBFT para criar um novo bloco (2), realizar a sua validação e adicioná-lo à blockchain (3).

A combinação PoS-PBFT busca melhorar a eficiência do processo de consenso usando a participação financeira como um indicador de confiança (PoS) e a capacidade de alcançar consenso rápido (PBFT). Isso resulta em uma redução na latência de confirmação de transa-

Figura 4.11 – Funcionamento da combinação entre o *Proof of Stake* e *Practical Byzantine Fault Tolerance*



Fonte: O autor (2025).

ções na blockchain. O Código 4.9 mostra o algoritmo da combinação entre PoS e PBFT. O processo de consenso verifica se o nó escolhido cumpre o PoS (Linha 6), em seguida o PBFT é utilizado para criar o novo bloco (Linha 8), e o propagar na blockchain (Linha 13).

Código 4.9 – Combinação do *Proof of Stake* e *Practical Byzantine Fault Tolerance*

```
1  Funcao IntegracaoPoSPBFT(transacao):
2      // Passo 1: Selecionar no inicial (PoS)
3      noPoS = SelecionarNoPoS()
4
5      //Passo 2: Verificar se no escolhido cumpre o PoS
6      Se NoParticipaDoConsensoPoS(noPoS):
7          // Passo 3: Criar novo bloco
8          blocoTemporario = IniciarPBFT(transacao, noPoS)
9
10         // Passo 4: Validacao usando PBFT
11         Se PBFTConsensoAtingido(blocoTemporario):
12             // Passo 5: Propagar o bloco
13             PropagarBlocoPBFT(blocoTemporario)
14         Senao:
15             // Passo 6: Tratamento de falhas
16             LidarComFalhasPBFT(transacao)
17     Senao:
18         LidarComFaltaDeParticipacao(transacao)
```

Fonte: O autor (2023)

Uma possível desvantagem do uso do PoS é o problema da "riqueza acumulada", que ocorre quando os participantes com mais moedas têm influência desproporcional sobre o processo de consenso. Isto pode levar à centralização do poder em poucos participantes, tornando a rede menos descentralizada e sujeita à manipulação por parte destes detentores de grandes

quantidades de moedas.

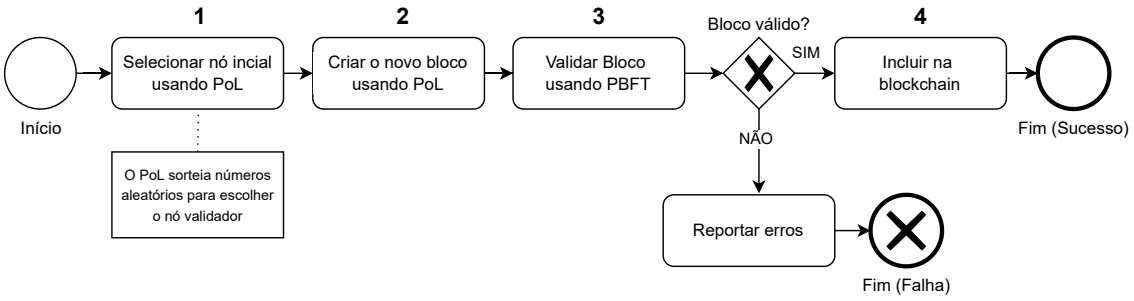
Esta centralização pode comprometer a segurança e a resistência da blockchain a ataques maliciosos, pois um participante com grande quantidade de moedas teria maior controle sobre o registro de transações e, consequentemente, capacidade para realizar ataques. Portanto, a distribuição inicial e contínua de moedas e as regras de implementação para mitigar a centralização são importantes ao usar o PoS.

4.2.10 *Proof of Luck* (PoL) e *Practical Byzantine Fault Tolerance* (PBFT)

Esta combinação integra a escolha justa e simplicidade de validadores do PoL (KIM *et al.*, 2024), com a segurança e tolerância a falhas do PBFT. O algoritmo PoL distribui de forma equitativa a oportunidade de criação de blocos entre os nós, enquanto o PBFT garante um consenso ágil e tolerante a falhas, mesmo em ambientes adversos. A Figura 4.12 descreve o funcionamento desta combinação.

Na primeira etapa do processo de consenso, o algoritmo PoL é utilizado para definir, de maneira aleatória, o nó candidato a criar um novo bloco (1). Em seguida, esse nó utiliza o algoritmo PBFT para realizar a criação (2), e posteriormente, a validação do novo bloco (3); através de um grupo de nós confiáveis, o consenso é atingido e o novo bloco incluído na blockchain (4).

Figura 4.12 – Funcionamento da combinação entre o *Proof of Luck* e o *Practical Byzantine Fault Tolerance*.



Fonte: O autor (2025).

A principal vantagem dessa combinação é a melhora de segurança e descentralização no processo de escolha do nó inicial do PBFT. A utilização do PoL proporciona uma distribuição justa das oportunidades de criação de blocos para todos os nós (KIM *et al.*, 2024). O Código 4.10 mostra o algoritmo da combinação PoL-PBFT. Inicialmente é feito um sorteio do nó validador

(Linha 3), o escolhido utilizará o PBFT para criar um novo bloco (Linha 7), e o propagar aos demais nós na blockchain (Linha 14).

Código 4.10 – Combinação do *Proof of Luck* e o *Practical Byzantine Fault Tolerance*

```
1  Funcao IntegracaoPoL_PBFT(transacao):
2      // Passo 1: Selecionar no validador
3      noSorteado ← SortearNoPoL()
4      // Passo 2: Verificar validade do no
5      Se NoValido(noSorteado):
6          // Passo 3: Criar bloco
7          bloco ← CriarBloco(transacao, noSorteado)
8
9          // Passo 4: Validacao usando PBFT
10         consenso ← PBFTConsenso(bloco)
11
12         // Passo 5: Incluir o bloco na Blockchain
13         Se (consenso):
14             PropagarBloco(bloco)
15         Senao:
16             TratarFalhaPBFT(bloco)
17     Senao:
18          TratarErroSelecaoPoL(transacao)
```

Fonte: O autor (2025)

A combinação PoL-PBFT, embora combine simplicidade e justiça na seleção de nós, pode enfrentar limitações importantes, como por exemplo, a aleatoriedade do PoL. Apesar de eficiente, o PoL pode gerar tempos de resposta imprevisíveis, o que compromete a previsibilidade do tempo de resposta da rede.

Além disso, o algoritmo PBFT, por depender de intensa comunicação entre os nós validadores, sofre com problemas de escalabilidade em redes grandes, aumentando a latência e o consumo de recursos. Dessa forma, essa combinação pode não ser ideal em cenários com grande número de dispositivos ou alta frequência de transações, como em aplicações IoT em larga escala.

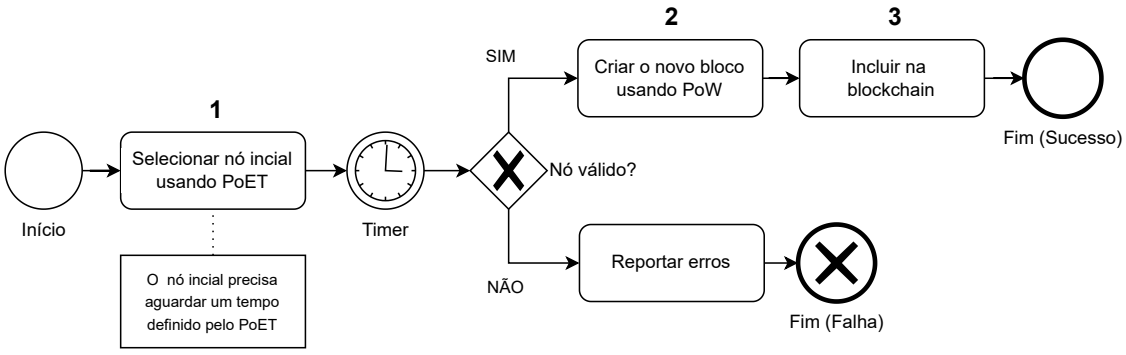
4.2.11 *Proof of Elapsed Time* (PoET) e *Proof of Work* (PoW)

O algoritmo de consenso PoET é utilizado por sua eficiência na escolha de nós validadores, pois distribui de forma justa a oportunidade de criação de blocos por meio de tempos de espera aleatórios (KAUR *et al.*, 2021). O PoW contribui para o processo de consenso fornecendo

segurança e resistência contra ataques e modificações (ÁLVAREZ; GRAMLICH; SEDLMEIR, 2024). A Figura 4.13 apresenta o diagrama de funcionamento desta combinação.

No início do processo de consenso, o algoritmo PoET é executado para determinar o nó com o menor tempo de espera, tornando-o apto a propor um novo bloco (1). Em seguida, o nó escolhido utiliza o PoW para criar um novo bloco (2), resolvendo um desafio criptográfico que reforça a imutabilidade e segurança da blockchain. Por fim, o bloco criado é inserido na blockchain (3).

Figura 4.13 – Funcionamento da combinação entre o *Proof of Elapsed Time* e o *Proof of Work*



Fonte: O autor (2025).

A principal vantagem dessa combinação é a busca por um consumo eficiente de recursos computacionais. Ao combinar a abordagem leve e justa do PoET com o poder de processamento e a resistência à manipulações do PoW, essa combinação se mostra promissora para cenários que exigem um desempenho otimizado, como em aplicações de IoT e outras soluções que lidam com dados críticos. O Código 4.11 mostra o algoritmo da combinação PoET-PoW. O algoritmo PoET sorteia um tempo aleatório (Linha 4), que deverá ser aguardado antes do processo de consenso ser realizado (Linha 5).

Código 4.11 – Combinação do *Proof of Elapsed Time* e o *Proof of Work*

```
1 Funcao IntegracaoPoET_PoW(transacao):
2     // Passo 1: Selecionar no candidato usando PoET
3     noCandidato ← SelecionarNoPoET()
4     tempo_espera ← SortearTempoAleatorioPoET(noCandidato)
5     Aguarde(tempo_espera)
6
7     // Passo 2: Verificar se o candidato no cumpriu o tempo de espera
8     Se NoElegivel(noCandidato):
9         // Passo 3: Criar o novo bloco utilizando PoW
```

```
10          bloco ← CriarBlocoPoW(transacao, noCandidato)
11
12          // Passo 4: Validar bloco
13          Se BlocoValido(bloco):
14              // Passo 5: Propagar o bloco na Blockchain
15              PropagarBloco(bloco)
16          Senao:
17              // Tratar possiveis falhas
18              TratarErroPoW(transacao)
19      Senao:
20           TratarErroSelecao(transacao)
```

Fonte: O autor (2025)

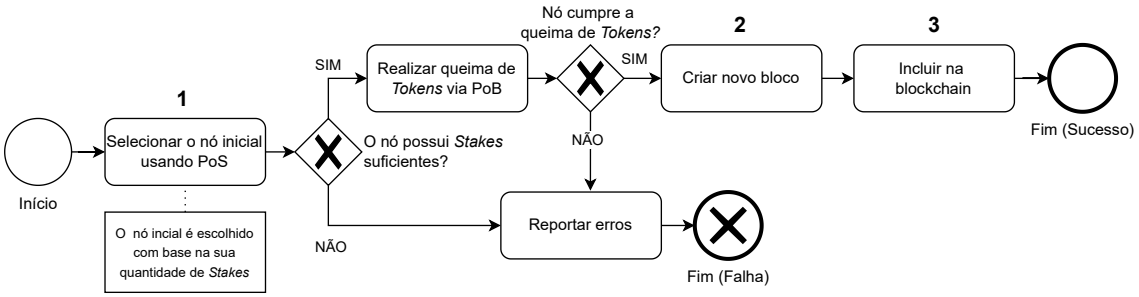
Apesar da combinação PoET-PoW buscar equilibrar a seleção justa e eficiente do nó de validação (PoET) com a robustez e segurança do PoW, ela apresenta algumas limitações. A dependência de hardwares específicos para a correta implementação do PoET pode restringir sua aplicabilidade em ambientes IoT com recursos limitados. Além disso, mesmo com a redução potencial no consumo de recursos por meio do PoET, o uso de PoW ainda impõe um uso elevado de recursos computacionais, o que pode afetar a escalabilidade e a eficiência operacional em redes com alta demanda.

4.2.12 *Proof of Stake* (PoS) e *Proof of Burn* (PoB)

A combinação entre os algoritmos PoS-PoB permite aproveitar os pontos fortes de cada algoritmo. O algoritmo PoS é usado por sua capacidade de selecionar validadores com base na quantidade de *stakes*, incentivando o comprometimento honesto dos participantes. Já o algoritmo PoB é utilizado na criação dos blocos, pois, ao exigir a queima de moedas, torna a produção de novos blocos dispendiosa para agentes maliciosos e assegura um processo de consenso baseado em prova de comprometimento. A Figura 4.14 descreve o funcionamento desta combinação.

Na primeira etapa do processo de consenso, o algoritmo PoS é empregado para escolher o nó que oferece a maior *stake*, que se torna assim, o validador responsável pela criação do novo bloco (1). Em seguida, o nó selecionado utiliza o algoritmo PoB para gerar um novo bloco (2), queimando uma quantidade pré-estabelecida de moedas como custo para a criação, o que reforça a segurança do processo de consenso. Finalmente, o bloco criado pode ser inserido na blockchain (3).

Figura 4.14 – Funcionamento da combinação entre o *Proof of Stake* e o *Proof of Burn*



Fonte: O autor (2025).

Esta combinação busca a integração entre o consumo eficiente de recursos computacionais e segurança: enquanto o algoritmo PoS reduz o consumo de recursos ao utilizar a participação financeira como critério de seleção de validadores, o algoritmo PoB adiciona uma camada de segurança ao processo de consenso, tornando a criação de blocos mais confiável e difícil de ser manipulada.

O Código 4.12 mostra o algoritmo da combinação entre PoS-PoB. Antes da criação de um novo bloco é preciso verificar se o nó possui *stakes* suficientes (Linhas 6) e se é realizada a queima de *tokens* (Linha 8).

Código 4.12 – Combinação do *Proof of Stake* e o *Proof of Burn*

```
1  Funcao IntegracaoPoS_PoB(transacao):
2      // Passo 1: Selecionar o no candidato com maior stake via PoS
3      noCandidato ← SelecionarNoPoS()
4
5      // Passo 2: Verificar se o no possui stakes suficiente
6      Se NoComStakeSuficiente(noCandidato):
7          // Passo 3: Realizar a queima de tokens (PoB)
8          Se RealizarQueimaDeTokens(noCandidato):
9              // Passo 4: Criar novo bloco utilizando PoB
10             bloco ← CriarBlocoPoB(transacao, nodoCandidato)
11
12             // Passo 5: Validar o bloco
13             Se BlocoValido(bloco):
14                 PropagarBloco(bloco)
15             Senao:
16                 TratarErroValidacao(transacao)
17         Senao:
18             TratarErroQueima(transacao)
19     Senao:
20          TratarErroStake(transacao)
```

Fonte: O autor (2025)

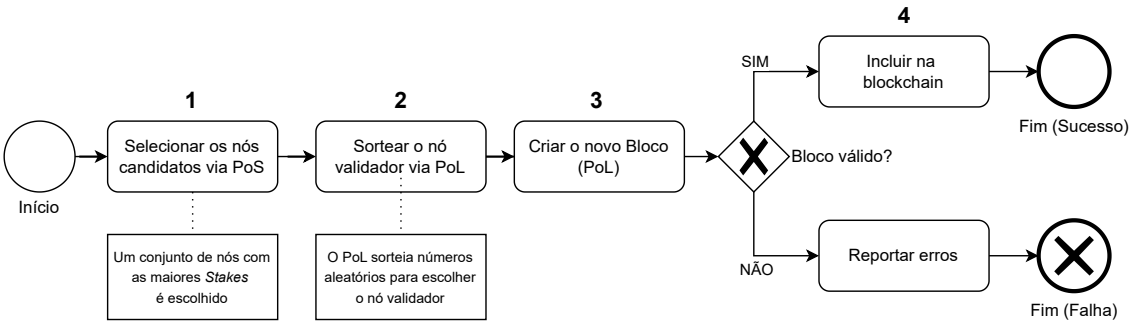
Esta combinação, embora eficiente em alguns quesitos, pode apresentar algumas desvantagens. A principal limitação do algoritmo PoS está na possibilidade de concentração de poder nos participantes com muitas *stakes*, o que pode comprometer a descentralização. Já o algoritmo PoB, por exigir a queima permanente de tokens, pode reduzir a liquidez e desestimular a participação contínua, principalmente em sistemas com oferta limitada de moedas. Além disso, o modelo de queima de moedas pode ser visto como economicamente ineficiente a longo prazo.

4.2.13 *Proof of Stake* (PoS) e *Proof of Luck* (PoL)

Esta combinação une os benefícios do PoS, que seleciona validadores com base nas suas *stakes*, à eficiência do PoL, que utiliza a sorte para determinar a criação de blocos. Enquanto o algoritmo PoS promove o comprometimento econômico dos participantes, o algoritmo PoL confere aleatoriedade e justiça ao processo de consenso. A Figura 4.15 descreve o funcionamento desta combinação.

No início do processo de consenso, o algoritmo PoS é usado para escolher um conjunto de nós com maiores *stakes*, tornando-os possíveis validadores (1). Em seguida, é utilizado o algoritmo PoL para sortear um nó dentre o conjunto de candidatos (2). O nó escolhido realiza a criação de um novo bloco (3), e o inclui na blockchain (4).

Figura 4.15 – Funcionamento da combinação entre o *Proof of Stake* e o *Proof of Luck*



Fonte: O autor (2025).

A principal vantagem dessa combinação é a redução do consumo de recursos computacionais, já que o PoL não demanda computação intensiva, e a promoção de um consenso mais justo e eficiente (PoS). Devido a essas características, esta combinação pode ser adequada

para ambientes IoT que exigem economia de recursos no gerenciamento de nós. O Código 4.13 mostra o algoritmo da combinação PoS-PoL. Inicialmente é selecionado um conjunto de nós que oferecem as maiores *stakes* (Linha 3); em seguida, o PoL sorteia o nó validador a partir desse conjunto (Linha 7).

Código 4.13 – Combinação do *Proof of Stake* e o *Proof of Luck*

```
1  Funcao IntegracaoPoS_PoL(transacao):
2      // Passo 1: Selecionar um conjunto de nos com maiores stakes (PoS)
3      conjuntoCandidatos ← SelecionarNosComMaiorStake()
4
5      Se Tamanho(conjuntoCandidatos) > 0:
6          // Passo 2: Aplicar PoL para sortear um no entre os candidatos
7          noSelecionado ← SortearNoPoL(conjuntoCandidatos)
8
9          // Passo 3: Criar bloco
10         bloco ← CriarBloco(transacao, noSelecionado)
11
12         // Passo 4: Validar bloco
13         Se BlocoValido(bloco):
14             // Passo 5: Propagar bloco
15             PropagarBloco(bloco)
16         Senao:
17             // Passo 6: Tratar erros
18             TratarErroValidacao(bloco)
19     Senao:
20         TratarErroSelecaoPoS(transacao)
```

Fonte: O autor (2025)

Esta combinação pode apresentar algumas desvantagens, devido ao fato de que o PoS tende a favorecer participantes com maior quantidade de moedas, o que pode levar à centralização de poder e dificultar a participação de novos nós. Além disso, embora o PoL introduza sorte na seleção, o elemento aleatório pode comprometer a previsibilidade e o controle sobre os tempos de criação de blocos, o que pode ser problemático em aplicações com requisitos estritos de latência ou sincronização.

4.2.14 *Proof of Stake* (PoS) e *Proof of Work* (PoW)

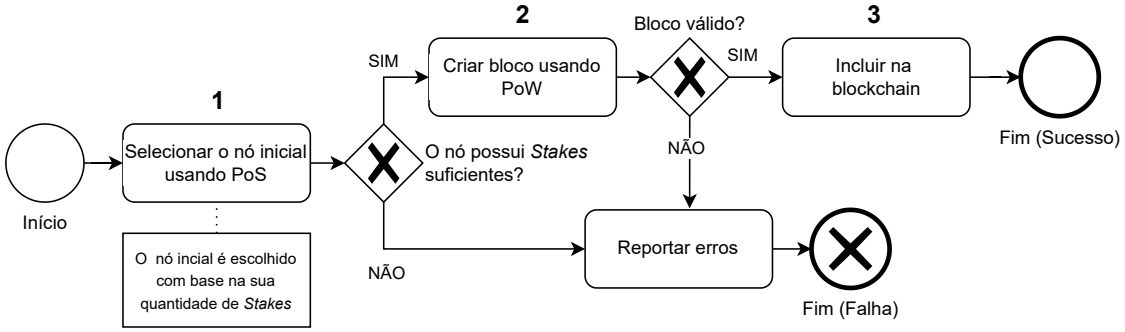
Nesta combinação, o algoritmo PoS foi utilizado pelo seu baixo consumo de energia e capacidade de criar novos blocos de acordo com a quantidade de moeda em poder dos participantes (KHAN; HARTOG; HU, 2022). Isto é relevante em ambientes IoT, onde a economia de energia

é crucial, e onde é desejada uma distribuição justa de recursos entre os nós participantes da blockchain.

O algoritmo de consenso PoW foi selecionado por sua resistência contra ataques e modificações, o que favorece a imutabilidade em ambientes blockchain desenvolvidos para a Internet das Coisas.

A Figura 4.16 mostra o diagrama de funcionamento desta combinação. O nó de validação inicial é selecionado entre aqueles que oferecem a maior quantidade de *stakes* (1). Em seguida, o nó selecionado executa o algoritmo PoW para criar um novo bloco (2). Por fim, o bloco criado é adicionado à blockchain (3).

Figura 4.16 – Funcionamento da combinação entre o *Proof of Stake* e o *Proof of Work*.



Fonte: O autor (2023).

Nesta combinação, o critério de seleção do nó validador é a participação financeira devido ao uso de PoS. Isso reduz a necessidade do algoritmo PoW de utilizar muitos recursos computacionais, tornando a criação de blocos mais eficiente e sustentável, o que contribui para a preservação dos recursos computacionais.

O Código 4.14 mostra o algoritmo da combinação entre os algoritmos PoS-PoW. Inicialmente o algoritmo verifica se o nó selecionado utilizou *stakes* suficientes (Linha 6), em seguida o PoW é utilizado para criar o novo bloco (Linha 8).

Código 4.14 – Combinação do *Proof of Stake* e o *Proof of Work*

```
1 Funcao IntegracaoPoSPoW(transacao):
2     // Passo 1: Selecionar no PoS
3     noPoS = SelecionarNoPoS()
4
5     // Passo 2: Verificar Stake PoS
6     Se NoComStakeSuficiente(noPoS):
7         // Passo 3: Criar bloco
```

```
8           blocoTemporario = CriarBlocoPoW(noPoS, transacao)
9
10          // Passo 4: Validar bloco
11          Se BlocoPoWPoSValido(blocoTemporario):
12              // Passo 5: Propagar bloco
13              PropagarBlocoPoWPoS(blocoTemporario)
14          Senao:
15              // Passo 7: Tratar de falhas
16              LidarComFalhas(transacao)
17      Senao:
18          LidarComFaltaDeStake(transacao)
```

Fonte: O autor (2023)

O ponto de atenção a ser considerado nesta combinação, é que o uso do algoritmo PoW pode consumir muita energia e recursos computacionais, e isto é crítico em ambientes IoT onde há a presença de dispositivos com recursos limitados.

4.2.15 *Proof of Stake* (PoS) e Raft

Nesta combinação, o algoritmo de consenso PoS foi utilizado devido à sua eficiência no consumo de recursos computacionais, o que o torna adequado para ambientes IoT. Ao mesmo tempo, o algoritmo Raft é especialmente projetado para lidar com falhas e melhorar a consistência e o desempenho do registro de dados na blockchain (HOWARD; MORTIER, 2020). A Figura 4.17 apresenta o funcionamento desta combinação.

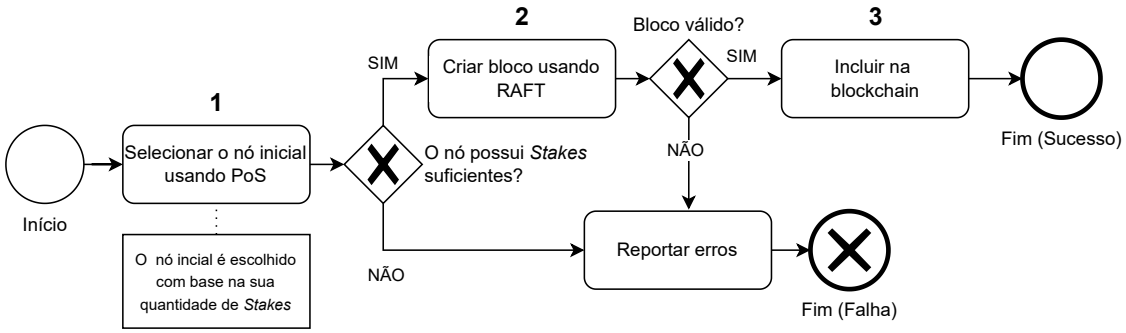
Para chegar ao consenso, o nó inicial é escolhido entre aqueles com maior quantidade de *stakes*, usando o PoS (1). Em seguida, o nó selecionado executa o algoritmo Raft para criar um novo bloco (2). No algoritmo Raft, também é necessário que outros nós validem o novo bloco, através de votação, antes de incluí-lo na blockchain (3).

O Raft é um algoritmo de consenso distribuído que prioriza consistência e disponibilidade (HOWARD; MORTIER, 2020). Este fato contribui para a estabilidade da blockchain, fazendo com que todos os nós estejam sincronizados, mesmo em condições adversas, como falhas ou possíveis nós maliciosos. O Código 4.15 mostra a combinação entre estes dois algoritmos de consenso. Se o nó validador escolhido possuir *stakes* suficientes (Linha 6), o Raft poderá ser utilizado para criar o novo bloco (Linha 8), e o propagar na blockchain (Linha 13).

Código 4.15 – Combinação do *Proof of Stake* e o *Raft*

```
1 Funcao IntegracaoPoS_RAFT(transacao):
2     // Passo 1: Selecionar no PoS
```

Figura 4.17 – Funcionamento da combinação entre o *Proof of Stake* e o *Raft*.



Fonte: O autor (2025).

```
    noPoS = SelecionarNoPoS()

    // Passo 2: Verificar Stake PoS
    Se NoComStakeSuficiente(noPoS):
        // Passo 3: Iniciar RAFT
        blocoTemporario = IniciarRAFT(transacao, noPoS)

        // Passo 4: Verificar consenso
        Se ConsensoAtingido(blocoTemporario):
            // Passo 5: Propagar do bloco
            PropagarBlocoRAFT(blocoTemporario)
        Senao:
            // Passo 6: Tratamento de falhas
            LidarComFalhasRAFT(transacao)
    Senao:
        LidarComFaltaDeParticipacao(transacao)
```

Fonte: O autor (2025)

O desempenho do algoritmo Raft pode ser afetado se o número de nós na rede for muito grande, o que pode ser uma preocupação em ambientes IoT que demandem alta escalabilidade. No processo de consenso do Raft o novo bloco precisa ser votado pelos demais participantes da rede antes de ser incluído na blockchain. Assim, caso o número de participantes seja muito grande o tempo de confirmação do bloco poderá ser alto.

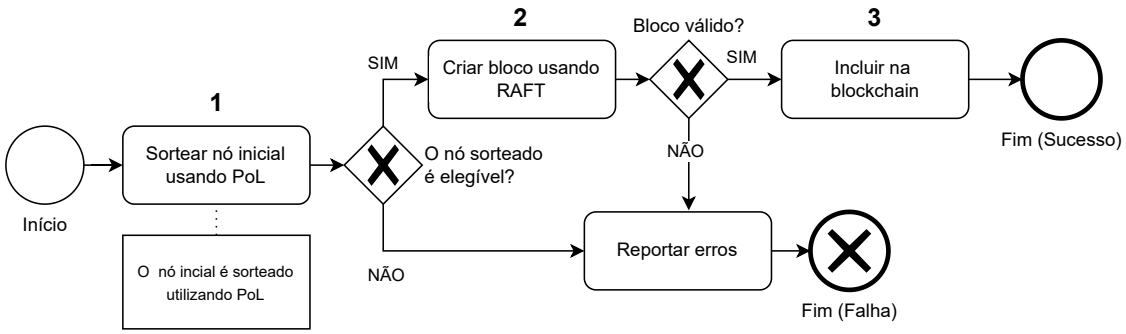
4.2.16 *Proof of Luck* (PoL) e Raft

A combinação entre os algoritmos de consenso PoL-Raft integra a escolha simples e justa de nós pelo PoL, com a eficiência e tolerância a adversários do algoritmo Raft. O algoritmo PoL

é adotado por sua capacidade de distribuir, de forma aleatória, a oportunidade de criação de blocos sem demandar muitos recursos computacionais, enquanto o algoritmo Raft é conhecido por alcançar consenso de maneira rápida e consistente em sistemas distribuídos.

A Figura 4.16 mostra o diagrama de funcionamento desta combinação. No processo de consenso, o PoL é utilizado inicialmente para determinar, por meio de um sorteio aleatório, qual nó se tornará elegível para propor o novo bloco (1). Após o processo de seleção, o nó escolhido utiliza o algoritmo Raft para iniciar o processo de votação e coordenação entre os nós (2), validando e incorporando o novo bloco criado à blockchain (3).

Figura 4.18 – Funcionamento da combinação entre o *Proof of Luck* e o Raft.



Fonte: O autor (2025).

A principal vantagem dessa combinação está na busca pela redução do consumo de recursos computacionais. A combinação entre esses dois algoritmos resulta em um sistema capaz de criar blocos de forma rápida, ideal para ambientes IoT que exigem escalabilidade e descentralização. O Código 4.16 mostra a implementação desta combinação. Inicialmente o nó validador é sorteado (Linha 3), em seguida o novo bloco será criado utilizando o Raft (Linha 8).

Código 4.16 – Combinação do *Proof of Luck* e o Raft

```
1  Funcao IntegracaoPoL_Raft(transacao):
2      // Passo 1: Sortear no lider (PoL)
3      noLider ← SortearNoPoL()
4
5      // Passo 2: Verificar elegibilidade do no sorteado
6      Se NoElegivel(noLider):
7          // Passo 3: Iniciar processo de consenso RAFT
8          bloco ← CriarBlocoRAFT(transacao, noLider)
9
10         // Passo 4: Validar Bloco
11         Se BlocoValido(bloco):
```

```
12              PropagarBloco(bloco)
13          Senao:
14              // Passo 6: Tratar erros
15              TratarErroRAFT(transacao)
16      Senao:
17          TratarErroSelecaoPoL(transacao)
```

Fonte: O autor (2025)

A combinação PoL-RAFT, embora traga simplicidade na seleção de líderes e confiabilidade no consenso, também apresenta algumas limitações importantes. A aleatoriedade do PoL pode compromete a previsibilidade do tempo de criação de novos blocos, especialmente em aplicações que requerem respostas rápidas.

Além disso, o algoritmo Raft foi projetado para grupos de nós pequenos e estáveis, e pode enfrentar desafios de desempenho e escalabilidade em redes maiores ou altamente dinâmicas, como é comum em aplicações IoT. A combinação deses dois algoritmos de consenso pode também exigir sincronização precisa entre os nós para evitar conflitos ou inconsistências, aumentando a complexidade operacional em ambientes distribuídos.

4.3 ESTRATÉGIA DE SELEÇÃO DE COMBINAÇÕES DE ALGORITMOS DE CONSENSO

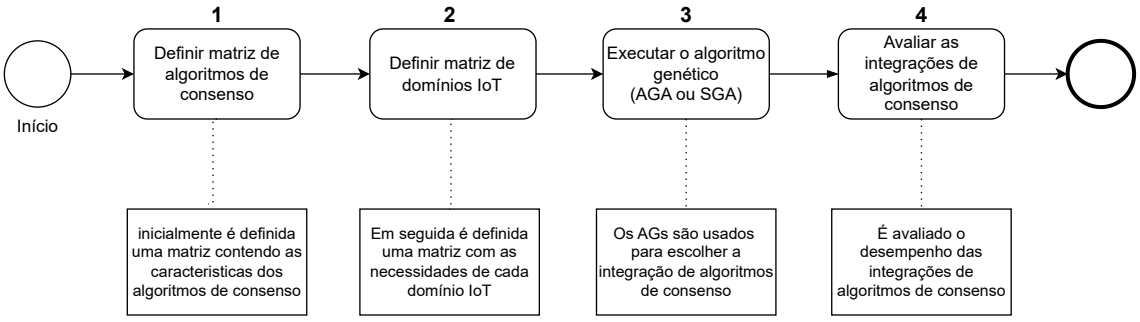
Após o desenvolvimento da blockchain *OmniBlock* e das combinações de algoritmos de consenso, foi necessário desenvolver uma estratégia para automatizar e otimizar a escolha dos algoritmos de consenso mais adequados a serem utilizados em cada um dos domínios da Internet das Coisas. A estratégia proposta nesta tese tem o objetivo de aprimorar a eficiência de escolha desses algoritmos para ambientes de IoT.

4.3.1 Visão Geral da Estratégia de Seleção

A estratégia desenvolvida consiste em quatro atividades principais. A Figura 4.19 apresenta uma visão geral da estratégia de escolha de algoritmos de consenso proposta nesta tese e das atividades que a compõem.

A primeira atividade da estratégia (1) é elaborar uma matriz de decisão contendo os algoritmos de consenso candidatos a serem utilizados no processo de consenso. Essa matriz contém 5 colunas, onde os valores de entrada são avaliados de acordo com as características de cada algoritmo, como: tempo para criação dos novos blocos, escalabilidade, latência, vazão

Figura 4.19 – Estratégia para escolher combinações de algoritmos de consenso



Fonte: O autor (2025).

e consumo de energia. Cada valor representa o quão eficiente o algoritmo é naquele quesito. Por exemplo, um algoritmo de consenso com valor alto no campo consumo de energia indica que ele é eficiente no consumo de energia durante suas execuções.

A segunda atividade da estratégia (2) consiste em elaborar uma matriz contendo os principais domínios de IoT. Essa matriz também contém 5 colunas, considerando as mesmas características anteriores: tempo para criação dos novos blocos, escalabilidade, latência, vazão e consumo de energia. Porém, nessa matriz os valores de entrada representam a necessidade que cada domínio de IoT tem de determinada característica. Por exemplo, se o domínio *Smart City* possuir um valor alto no campo de consumo de energia, significa que ele necessita de um algoritmo de consenso que seja altamente eficiente nesse quesito para satisfazer as suas necessidades.

A terceira atividade (3) consiste na execução dos algoritmos genéticos SGA e AGA. Esses algoritmos recebem, como entrada, as matrizes contendo os valores definidos nas atividades anteriores e, com base nos valores definidos em cada matriz, fornecem as sugestões de combinações de algoritmos de consenso mais apropriadas para cada domínio de IoT.

Finalmente, na quarta atividade (4), será avaliado o desempenho das combinações de algoritmos de consenso que foram sugeridas pelos algoritmos genéticos na atividade anterior. Nesta atividade, será analisado o custo computacional de criar um novo bloco e registrá-lo na blockchain *OmniBlock*.

4.3.2 Parametrização da Estratégia de Seleção

4.3.2.1 Definição dos pesos da matriz de algoritmos de consenso

De acordo com os passos da estratégia, mostrados na Figura 4.19, inicialmente é preciso definir uma matriz contendo os algoritmos de consenso implementados na blockchain *Omni-Block* (Tabela 4.1). Essa blockchain suporta 10 algoritmos de consenso, que foram escolhidos dentre aqueles que são mais frequentemente utilizados em blockchains para IoT e mencionados na literatura (KHAN; HARTOG; HU, 2022), (SALIMITARI; CHATTERJEE; FALLAH, 2020).

As colunas dessa matriz são compostas por pesos numerados de 1 a 10, representando as características de cada algoritmo. Cinco características foram consideradas para definir um blockchain para IoT: tempo para criação de novos blocos, escalabilidade, latência, vazão e consumo de energia.

Tabela 4.1 – Características analisadas em cada algoritmo de consenso.

	Tempo para criação de novos blocos	Escalabilidade	Latência	Vazão	Consumo de energia
PoW	2	8	4	5	6
PoS	3	7	5	6	9
PoA	5	8	4	2	4
PBFT	8	6	1	7	10
RAFT	9	6	4	3	9
PoET	1	5	2	5	2
PoL	3	2	5	9	7
PoB	2	7	4	6	9
PoI	5	6	2	1	4
dPoS	4	3	1	5	6

Fonte: O autor (2025).

Em blockchains para IoT, o tempo para criação de um novo bloco é o intervalo entre o envio de uma transação e sua inclusão efetiva na blockchain; escalabilidade é a capacidade da rede de manter o desempenho e a confiabilidade à medida que o número de nós ou transações aumenta; latência refere-se ao atraso entre o envio de uma transação e sua confirmação pelos validadores; taxa de transferência é o número de transações ou blocos processados por segundo pela rede; e consumo de energia abrange o uso de recursos computacionais (CPU, RAM e bateria) necessários para executar o protocolo de consenso em dispositivos com restrição de

energia.

Para definir os pesos da primeira coluna, *Tempo para criação de novos blocos*, foram considerados os valores obtidos na avaliação de desempenho realizada em Morais, Lins e Rosa (2023a), onde os tempos de cada algoritmo de consenso foram medidos. Pesos próximos a 10 foram definidos para os algoritmos que criaram blocos mais rapidamente, por exemplo, PBFT e Raft, e pesos próximos a 1 foram definidos para os algoritmos mais lentos, como o PoW e o PoET.

Para definir os pesos das demais colunas, utilizou-se como base a tabela do Apêndice D, que reproduz a Tabela 6 de Morais, Lins e Rosa (2023b), onde são analisados os principais algoritmos de consenso apresentados na literatura.

Na tabela do Apêndice D, *Escalabilidade*, *Latência*, *Vazão* e *Consumo de Energia* são classificados qualitativamente como *Alto*, *Médio* e *Baixo*. Para definir os pesos, foi realizada uma equivalência: os algoritmos com conceito *Alto* foram divididos em *Muito Alto* (10-8) e *Alto* (6-7). *Médio* (5). Aqueles com conceito *Baixo* foram divididos em *Baixo* (4-3) e *Muito Baixo* (2-1).

4.3.2.2 Definição dos pesos da matriz de domínios da IoT

A segunda atividade no desenvolvimento da estratégia para escolha das combinações foi criar uma matriz de decisão contendo os principais domínios da IoT, conforme mostrado em Tabela 4.2. Essa matriz possui as mesmas colunas da matriz de algoritmos de consenso apresentada na Tabela 4.1.

No entanto, aqui, os valores representam o quanto cada domínio necessita dessa característica presente no algoritmo de consenso que será utilizado. Por exemplo, um peso 10 na coluna *Escalabilidade* significa que esse domínio necessita de um algoritmo de consenso que suporte alta escalabilidade, um peso 1 representa que o aspecto da escalabilidade não é tão relevante nesse domínio e assim por diante para as demais colunas.

Para definir os pesos dessa matriz foi realizada uma análise empírica dos domínios IoT, que permitiu identificar como cada uma das dimensões presentes nas colunas da Tabela 4.2 é tratada na literatura (ZUBAYDI; VARGA; MOLNÁR, 2023; KHAN *et al.*, 2023; CHAKRABORTY *et al.*, 2023).

Para esta análise, oito domínios da IoT foram considerados (Tabela 4.2): *Smart Home* (LEE *et al.*, 2020); *Healthcare* (ZAABAR *et al.*, 2021); *Smart Agriculture* (LEI *et al.*, 2022); *Industrial*

Tabela 4.2 – Características analisadas em cada domínio IoT.

	Tempo para criação de blocos	Escalabilidade	Latência	Vazão	Consumo de energia
Smart Home	6	3	7	5	9
Healthcare	9	2	5	3	7
Smart Agriculture	3	9	3	6	8
Industrial IoT	8	8	4	7	1
Smart Cities	9	7	5	7	8
Smart Logistics	7	3	9	4	3
Smart Energy	8	8	3	6	2
Internet of Vehicles	9	1	4	3	9

Fonte: O autor (2025).

IoT (ZUBAYDI; VARGA; MOLNÁR, 2023); *Smart Cities* (PAUL *et al.*, 2021); *Smart Logistics* (UGOCHUKWU *et al.*, 2022); *Smart Energy* (KHAN *et al.*, 2023); e *Internet of Vehicles* (JAVED *et al.*, 2020).

Existem diversos outros domínios da IoT, que surgem conforme aplicações e setores evoluem (por exemplo, *wearables devices* - dispositivos vestíveis). No entanto, os oito que foram escolhidos para esta tese formam um conjunto representativo das maiores classes de aplicação da IoT em termos de impacto socioeconômico e volume de dispositivos. Os domínios escolhidos são suficientes para analisar um conjunto amplo e diverso de requisitos, como latência, consumo de energia, vazão e escalabilidade.

Os valores da Tabela 4.2 foram definidos em 10 níveis, que representam a necessidade que cada domínio possui de cada uma das características consideradas. Por exemplo, o valor 10 significa uma necessidade crítica, 8 ou 9 para necessidade muito alta, 6 ou 7 para uma necessidade alta, 5 para necessidade média, 3 ou 4 para necessidade baixa e 1 ou 2 para uma necessidade muito baixa.

As matrizes de decisão foram escolhidas em uma escala de 1 a 10 para fornecer granularidade suficiente para que o algoritmo genético otimize as integrações. Com dez níveis de peso, variações sutis nas características de cada combinação de consenso (como tempo de bloco, latência ou consumo de energia) podem ser representadas com mais precisão, enriquecendo o espaço de busca e facilitando a convergência do algoritmo para soluções balanceadas.

Além disso, essa granularidade adicional ajuda a evitar saltos repentinos na avaliação de aptidão, promovendo o ajuste fino e um processo de seleção mais sensível às diferenças de

desempenho entre as integrações.

4.4 EXECUÇÃO DOS ALGORITMOS GENÉTICOS PARA ESCOLHA DAS COMBINAÇÕES

Após a definição dos pesos da matriz dos algoritmos de consenso (Tabela 4.1) e da matriz dos domínios da IoT (Tabela 4.2), a terceira atividade da estratégia (ver Figura 4.19) é a execução dos algoritmos genéticos para calcular qual combinação é mais adequada para cada domínio.

Após as execuções dos algoritmos genéticos, foram obtidas as indicações das combinações de algoritmos de consenso mais adequadas para cada um dos domínios da IoT. A Tabela 4.3 apresenta os resultados obtidos nesta etapa da estratégia para a escolha das combinações de consenso.

Tabela 4.3 – Sugestão de combinações de algoritmos de consenso segundo os algoritmos genéticos

	SGA	AGA
Smart Home	PoS-PoW	PoS-PoL
Healthcare	PoS-RAFT	PoS-PoL
Smart Agriculture	PoS-PoB	PoS-PoW
Industrial IoT	PoA-PoW	PoET-PoA
Smart Cities	PoB-PBFT	PoI-RAFT
Smart Logistics	PoET-PoW	PoA-PoL
Smart Energy	PoA-PoW	PoI-RAFT
Internet of Vehicles	PoL-PBFT	PoI-PBFT

Fonte: O autor (2025).

Como é possível observar, as combinações sugeridas pelos SGA e pelo AGA são diferentes. Isso se deve ao fato de que no AGA as taxas de mutação e cruzamento variam e são ajustadas dinamicamente durante a execução do algoritmo. Esse ajuste dinâmico permite ao AGA explorar um número maior de possíveis combinações e apresentar resultados diferentes do SGA.

4.5 CONSIDERAÇÕES FINAIS

Este capítulo apresentou o projeto e implementação da blockchain *OmniBlock*, desenvolvida com base em uma estrutura de DAG e que oferece suporte a múltiplos algoritmos de consenso. Também foi descrita a estratégia para a seleção de combinações de algoritmos de consenso desenvolvida nesta tese. O capítulo apresentou a utilização de algoritmos genéticos para otimizar a seleção das combinações, permitindo ajustes dinâmicos que buscam melhorar o desempenho do processo de consenso. No próximo capítulo a solução proposta nesta tese será avaliada, considerando diversas métricas de desempenho.

5 AVALIAÇÃO EXPERIMENTAL

Este capítulo apresenta a avaliação da blockchain *OmniBlock* e das combinações de algoritmos de consenso sugeridas pelos algoritmos genéticos. Métricas importantes são consideradas nesta avaliação, como tempo para criação de novos blocos e consumo de recursos computacionais como CPU e memória. Por fim, é apresentado um comparativo entre os resultados fornecidos pelos algoritmos genéticos.

5.1 OBJETIVOS

A metodologia de avaliação utilizada nesta avaliação de desempenho foi inspirada na abordagem elaborada por Jain (1991). Os objetivos dessa avaliação são:

[*Objetivo 1*] Comparar o consumo de recursos computacionais da blockchain *OmniBlock*, com suas combinações de algoritmos de consenso, e uma blockchain linear sem combinações de algoritmos.

[*Objetivo 2*] Fazer uma avaliação comparativa entre os algoritmos genéticos.

Os principais componentes utilizados neste estudo são os algoritmos de consenso, e suas combinações, e os algoritmos genéticos. Os sistemas consistem em uma blockchain linear e uma blockchain baseada em DAG, ambas implementadas através da utilização de contêineres Docker (BOETTIGER, 2015).

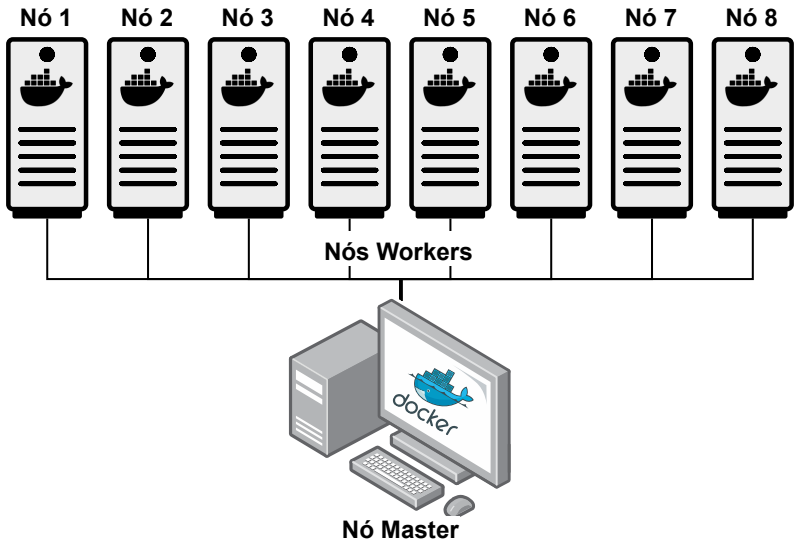
Os algoritmos de consenso foram implementados em *Node.js* (TILKOV; VINOSKI, 2010). Os experimentos foram executados em um *cluster* configurado utilizando Docker *Swarm* (SOPPELSA; KAEWKASI, 2016), uma ferramenta nativa do Docker que permite gerenciar e escalar aplicações distribuídas em *clusters* de máquinas. As demais ferramentas utilizadas para o desenvolvimento da *OmniBlock* são descritas no Apêndice E.

A Figura 5.1 mostra uma representação do *cluster* desenvolvido para este trabalho, que possui um nó principal chamado *master*, responsável por executar as combinações de algoritmos de consenso e 8 nós de trabalho, chamados *workers*, responsáveis por validar e replicar os dados registrados na *OmniBlock*.

O principal serviço oferecido pelo sistema consiste na criação e registro de novos blocos na blockchain. Os dados cadastrados foram cedidos pela *Startup* Semine[1], e consistem em dados reais, provenientes de sensores de umidade e temperatura do ar e do solo coletados em

[1] Mais informações podem ser obtidas em https://semine.ag/

Figura 5.1 – *Cluster* utilizado para executar as blockchains



Fonte: O autor (2025).

uma estação de monitoramento de uma propriedade agrícola. Foram escolhidos dados IoT do agronegócio porque este é um dos setores mais importantes para o Brasil, representando valor significativo do PIB (CNA BRASIL, 2025).

Devido ao escopo desta tese ser focado em desempenho, os possíveis erros e falhas de blocos não foram considerados. Portanto, o estudo se limitou à criação e inserção correta de blocos na blockchain.

5.2 MÉTRICAS

Para ambos os objetivos da avaliação, foram escolhidas quatro métricas:

1. *Tempo médio de criação de novos blocos*: é o intervalo de tempo entre o início da validação e a confirmação de um bloco na blockchain;

2. *Consumo médio de CPU*: consiste no uso do processador durante as operações de validação e criação de blocos;

3. *Consumo médio de memória*: quantidade de memória RAM ocupada pelas instâncias de nó durante o funcionamento da blockchain; e

4. *Tráfego médio de rede*: é o volume médio de dados (em MB/s) enviado e recebido pelos nós durante a sincronização e propagação de blocos.

A utilização dessas métricas avalia o desempenho do processo de consenso na blockchain. Um menor tempo de criação de blocos reduz janelas de ataque, enquanto que um consumo moderado de recursos computacionais e tráfego de rede dificultam, por exemplo, ataques de negações de serviço.

5.3 PARÂMETROS E FATORES

Os parâmetros do sistema que afetam o tempo de criação de novos blocos na blockchain são a velocidade das CPUs do *Cluster* que hospeda a implementação, o tamanho da mensagem armazenada no novo bloco e a largura de banda da rede do Docker Swarm.

A Tabela 5.1 mostra a configuração das máquinas presentes no *Cluster*. Foram utilizadas 9 máquinas, que juntas totalizaram 104 GB de RAM. Todas as máquinas utilizaram o sistema operacional Linux Ubuntu 24.04 LTS.

Tabela 5.1 – Configurações das máquinas presentes no *Cluster* Docker Swarm

Nº de máquinas	CPU	RAM de cada máquina
4	Intel Core i5-3470	16 GB
2	Intel Core i5-5200	8 GB
2	Intel Core i3-4160	8 GB
1	Intel Core i3-3217	8 GB

Fonte: O autor (2025).

Em uma avaliação de desempenho, existem fatores e níveis que, quando variados, influenciam no desempenho do sistema. Nesta avaliação, quatro fatores foram considerados. O primeiro fator, relacionado ao Objetivo 1 da avaliação de desempenho, foi a configuração da blockchain, que variou entre configuração linear e baseada em DAG.

O segundo fator, também relacionado ao Objetivo 1, foi a quantidade de nós presentes na blockchain, que variou em 100, 500 e 1000 nós. Esses números foram escolhidos para permitir estimar o comportamento da blockchain ao escalar o número de nós. O número máximo de nós suportados pelo *Custer* foi de mil nós.

Outros dois fatores considerados, ambos relacionados ao Objetivo 2 da avaliação, foram as taxas de mutação e cruzamento dos algoritmos genéticos, que variaram em níveis *baixo*,

médio e *alto*, com o objetivo de analisar qual variação resultaria em melhores combinações de algoritmos de consenso. A Tabela 5.2 apresenta os fatores e níveis considerados nesta avaliação.

Tabela 5.2 – Fatores e níveis da avaliação de desempenho

Fatores	Níveis
Configuração da blockchain	Linear, DAG
Número de nós da blockchain	100, 500, 1000
Taxa de mutação	0,01, 0,05, 0,1
Taxa de cruzamento	0,1, 0,5, 1,0

Fonte: O autor (2025).

5.4 CARGA DE TRABALHO E PROJETO DOS EXPERIMENTOS

A carga de trabalho é gerada por um programa enviando dados para serem registrados em um novo bloco e solicitando a execução das combinações de algoritmos de consenso. Este programa também monitora o tempo entre o envio da solicitação e a confirmação da inclusão do novo bloco. As métricas de consumo médio de CPU, memória e tráfego de rede são coletadas de cada uma das máquinas presentes no *Cluster* usando a ferramenta cAdvisor[2] e são monitoradas pelo Prometheus[3].

Foram realizados dois conjuntos de experimentos, o primeiro deles utilizando uma blockchain linear e o segundo conjunto utilizando a blockchain *OmniBlock*. Em cada conjunto de experimentos, novos blocos são criados utilizando os algoritmos de consenso e suas combinações.

Para calcular o tempo de criação de novos blocos, cada procedimento foi repetido 30 vezes (CAMPELO; TAKAHASHI, 2019), e em seguida foi calculado o tempo médio. Em cada execução também foram coletados valores médios de consumo de CPU, memória e tráfego de rede. A duração de cada experimento foi de 5 minutos, e as métricas foram coletadas a cada 3 segundos.

[2] cAdvisor: https://github.com/google/cadvisor
[3] Prometheus: https://prometheus.io/

5.5 AVALIAÇÃO COMPARATIVA

O objetivo deste experimento inicial é gerar valores de referência que nos permitam verificar se o desempenho da blockchain *OmniBlock*, baseada em DAG e com combinações de consensos escolhidas através de algoritmos genéticos, melhorou em comparação com o modelo tradicional (Objetivo 1 - seção 5.1).

No primeiro conjunto de experimentos, foi utilizada uma blockchain tradicional, com blocos lineares e 10 algoritmos de consenso. Porém, não são utilizadas combinações, apenas um algoritmo de consenso é escolhido a cada criação de blocos. A quantidade de nós dessa blockchain foi configurada para 100, 500 e 1000 nós. Os dados completos dessa avaliação de desempenho são apresentados no Apêndice B.

Os algoritmos de consenso escolhidos para esta avaliação foram o PoW, PoS, PoA, PBFT, Raft, PoL, PoB, PoI, dPoS e PoET. Esses algoritmos foram escolhidos por serem amplamente conhecidos e utilizados em cenários reais de blockchain, como Bitcoin (YAZDINEJAD *et al.*, 2020) e Ethereum (LONE; NAAZ, 2021).

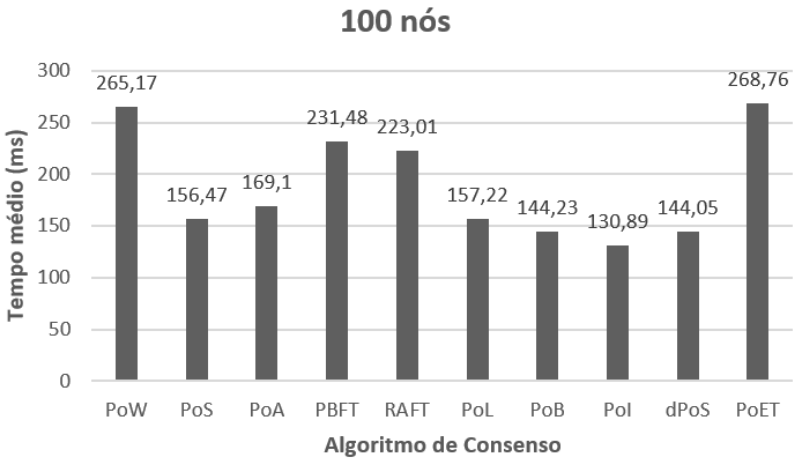
Cada um dos nós da blockchain é representado por um contêiner Docker executado no *cluster* da Figura 5.1. Os resultados obtidos na execução destes experimentos são apresentados nas seções a seguir.

5.5.1 Tempo médio de criação de novos blocos

A primeira métrica avaliada neste experimento foi o tempo necessário para a criação de um novo bloco por cada um dos algoritmos de consenso escolhidos. Essa é uma das métricas mais importantes em uma blockchain para IoT, devido ao fato de que a maioria das aplicações requer respostas rápidas. Por isso, um tempo de criação de blocos baixo pode evitar o comprometimento no desempenho da blockchain.

A Figura 5.2 mostra os tempos para a blockchain com 100 nós. Como é possível observar no gráfico, os algoritmos que apresentaram os menores tempos de criação de novos blocos foram o PoI e o dPoS.

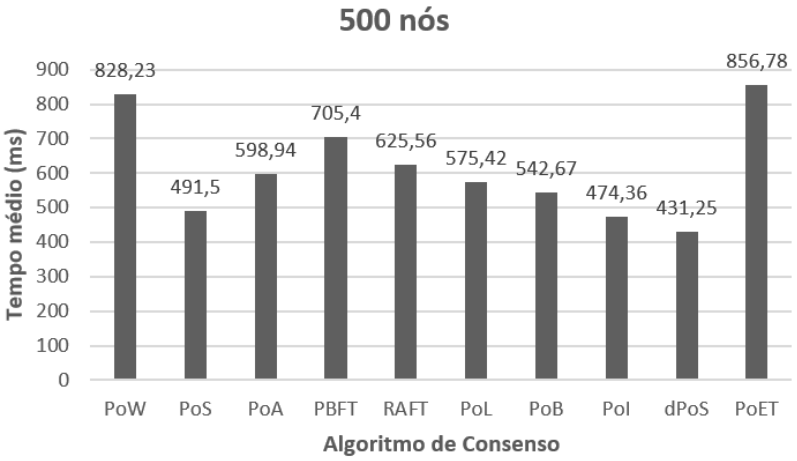
Figura 5.2 – Tempo médio para criar novos blocos usando 100 nós em uma blockchain linear



Fonte: O autor (2025).

Ao escalar o número de nós da blockchain de 100 para 500 nós, houve um aumento significativo nos tempos de criação de blocos em todos os algoritmos de consenso. O tempo médio de todos os algoritmos passou de 189,93 ms, com 100 nós, para 613,01 ms, um aumento de 222,7%. A Figura 5.3 mostra o resultado deste experimento. Nesse experimento, os algoritmos de consenso que apresentaram os melhores resultados foram o dPoS, seguido pelo PoI e o PoS.

Figura 5.3 – Tempo médio para criar novos blocos usando 500 nós em uma blockchain linear

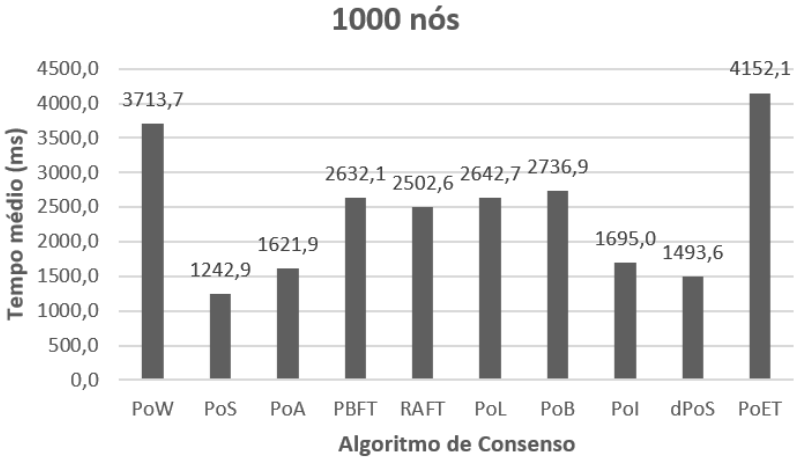


Fonte: O autor (2025).

Para a configuração da blockchain contendo 1000 nós, os resultados são apresentados na Figura 5.4. Ao aumentar o número de nós, o tempo necessário para criar novos blocos também aumenta, devido à necessidade de uma quantidade maior de nós para atingir o consenso. Neste

experimento, o tempo médio total passou de 613,01 ms para 2443,34 ms, o que representa um aumento de 298,6%.

Figura 5.4 – Tempo médio para criar novos blocos usando 1000 nós em uma blockchain linear

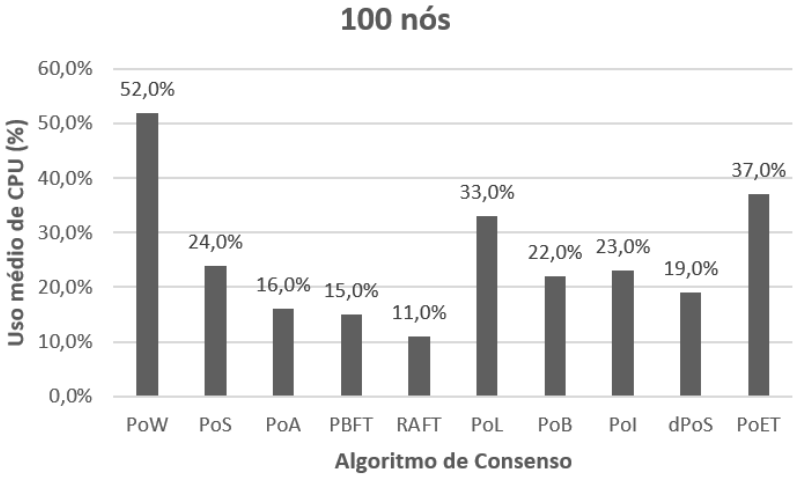


Fonte: O autor (2025).

5.5.2 Consumo médio de CPU

A segunda métrica considerada foi o consumo médio de CPU por cada algoritmo de consenso. A Figura 5.5 apresenta os resultados obtidos ao avaliar a blockchain com 100 nós. Os algoritmos de consenso que apresentaram menor consumo de CPU foram o Raft, seguido pelo PBFT e PoA.

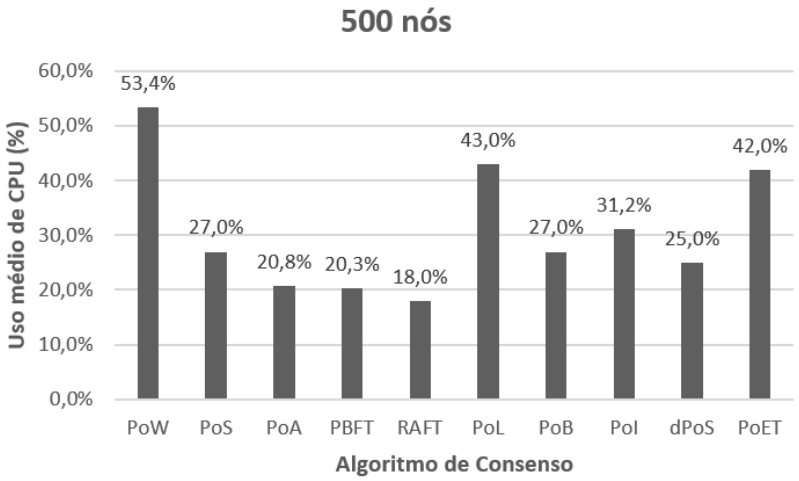
Figura 5.5 – Consumo médio de CPU usando 100 nós em uma blockchain linear.



Fonte: O autor (2025).

Para a blockchain contendo 500 nós, a variação no consumo médio de CPU provocada pelos algoritmos de consenso não sofreu grandes alterações. A média de uso de CPU de todos os algoritmos para 100 nós foi de 25,2%, para 500 nós a média foi de 29,9%, o que representa um aumento de 18,7%. A Figura 5.6 mostra os resultados deste experimento, os algoritmos de consenso que apresentaram as maiores médias de consumo de CPU foram PoW, PoL e PoET.

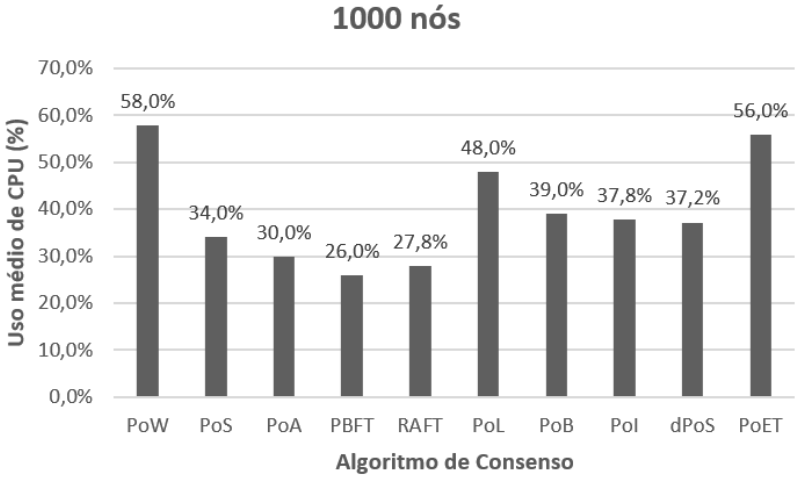
Figura 5.6 – Consumo médio de CPU usando 500 nós em uma blockchain linear.



Fonte: O autor (2025).

Para a blockchain contendo 1000 nós, mais uma vez as variações nas médias de consumo de CPU são pequenas. O consumo médio de todos os algoritmos foi de 34,5%, o que representa um aumento de 15,4% em relação ao resultado anterior. Neste experimento, o algoritmo de consenso mais eficiente foi o PBFT, conforme apresentado na Figura 5.7.

Figura 5.7 – Consumo médio de CPU usando 1000 nós em uma blockchain linear.



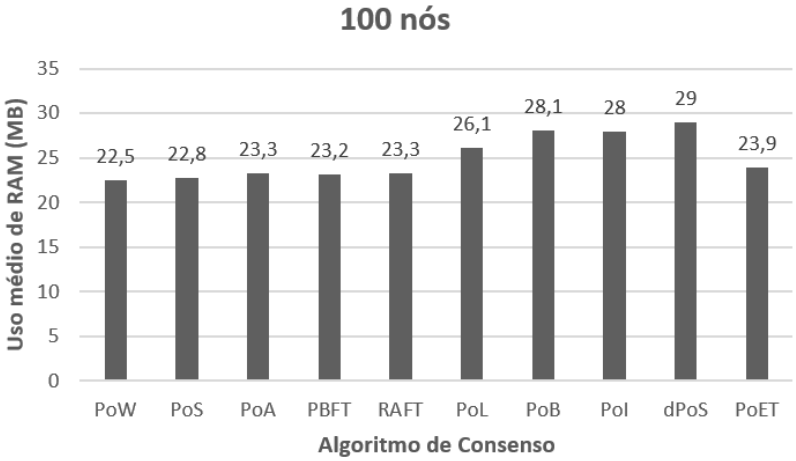
Fonte: O autor (2025).

5.5.3 Consumo médio de memória

A terceira métrica considerada nesta avaliação de desempenho foi o consumo médio de memória pelos nós presentes na blockchain linear ao executar cada um dos algoritmos de consenso.

A Figura 5.8 apresenta os resultados obtidos para a configuração da blockchain linear contendo 100 nós. Muitos algoritmos de consenso apresentam valores semelhantes de consumo de memória, porém os algoritmos que apresentaram os menores valores foram o PoW, o PoS e o PBFT.

Figura 5.8 – Consumo médio de memória usando 100 nós em uma blockchain linear.

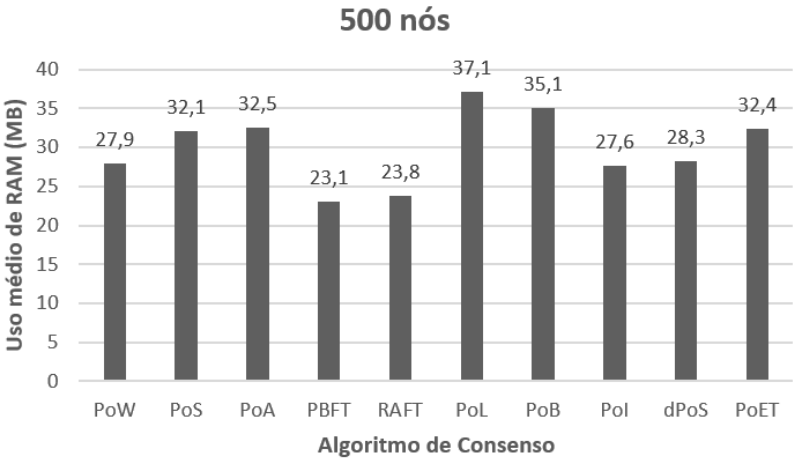


Fonte: O autor (2025).

Para a configuração da blockchain linear contendo 500 nós, foi possível observar uma maior variação entre os valores de consumo de memória durante a execução dos algoritmos de consenso.

A média geral dos algoritmos para a blockchain com 100 nós foi de 25,2 MB, enquanto que com 500 nós houve um aumento de 22,2%, passando para 30,8 MB. De acordo com a Figura 5.9, os algoritmos de consenso que apresentaram menor consumo de memória foi o PBFT seguido pelo Raft.

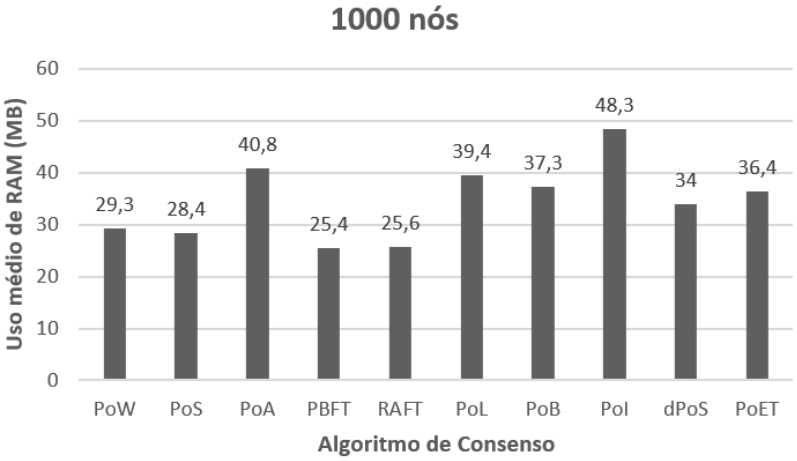
Figura 5.9 – Consumo médio de memória usando 500 nós em uma blockchain linear.



Fonte: O autor (2025).

Para a configuração da blockchain linear contendo 1000 nós, foi possível observar uma maior variação entre os valores das médias de consumo de memória, em comparação com os experimentos anteriores. A média geral teve um aumento de 27,9%, passando de 30,8 MB, com 500 nós, para 39,4 MB neste experimento. A Figura 5.10 mostra os resultados deste experimento. Aqui os algoritmos de consenso que obtiveram os piores resultados foram o PoI, PoA e PoL.

Figura 5.10 – Consumo médio de memória usando 1000 nós em uma blockchain linear.

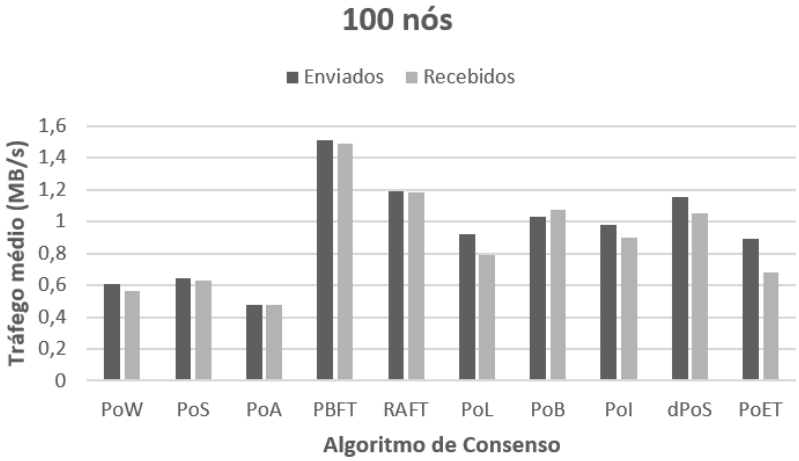


Fonte: O autor (2025).

5.5.4 Tráfego médio de rede

A quarta métrica avaliada foi o tráfego médio de rede entre os nós da blockchain. Esta métrica considera a quantidade de dados enviados e recebidos por cada nó. A Figura 5.11 mostra os resultados para a blockchain contendo 100 nós, considerando cada um dos algoritmos de consenso.

Figura 5.11 – Tráfego médio de rede usando 100 nós em uma blockchain linear.

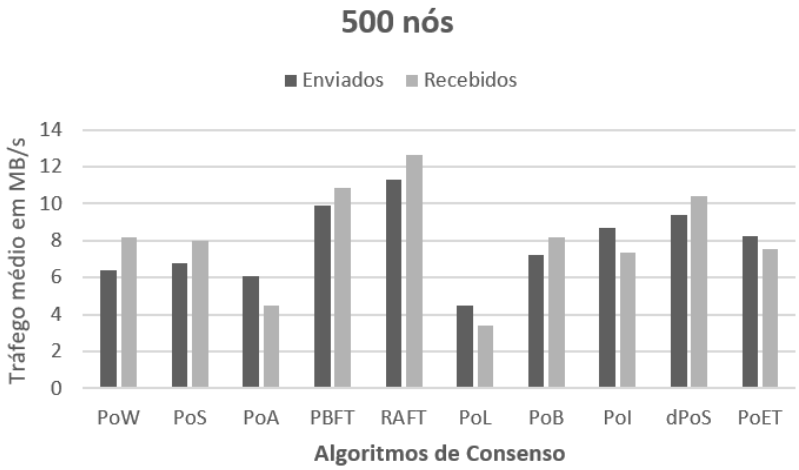


Fonte: O autor (2025).

Ao elevar o número de nós da blockchain linear de 100 para 500 nós, os valores médios de tráfego de rede obtidos aumentaram consideravelmente. A média total dos algoritmos para a blockchain com 100 nós passou de 0,91 MB/s para 7,98 MB/s neste experimento, um aumento de 776,9% .

Na Figura 5.12, os algoritmos que apresentaram os maiores valores de tráfego de rede (MB/s enviados e recebidos) foram o Raft e PBFT. E o algoritmo de consenso que gerou menos tráfego foi o PoL.

Figura 5.12 – Tráfego médio de rede usando 500 nós em uma blockchain linear.

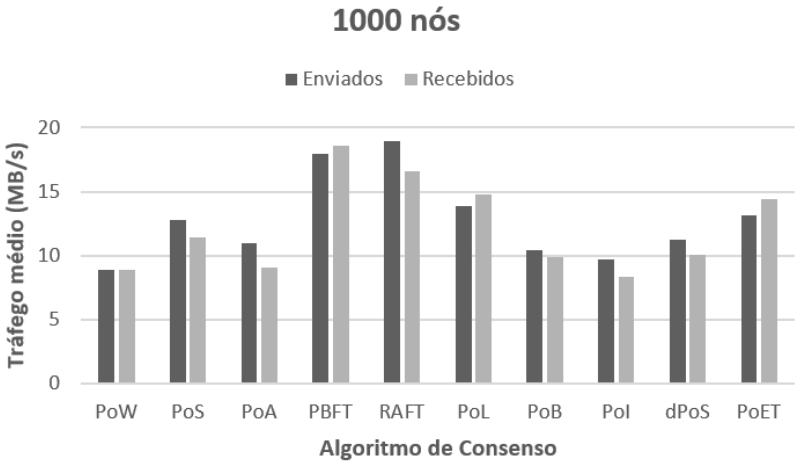


Fonte: O autor (2025).

Para a configuração da blockchain contendo 1000 nós, os valores de tráfego de rede obtidos aumentaram para todos os algoritmos. O valor médio total passou de 7,98 MB/s para 12,49 MB/s, o que representa um aumento de 56,5%.

A Figura 5.13 apresenta os resultados deste experimento. Como era esperado, os algoritmos que apresentam os maiores tráfegos de rede são o PBFT e o Raft, pois esses algoritmos realizam muitas trocas de mensagens antes de atingir o consenso.

Figura 5.13 – Tráfego médio de rede usando 1000 nós em uma blockchain linear.



Fonte: O autor (2025).

A Tabela 5.3 apresenta um sumário das quatro métricas consideradas na configuração de blockchain com 100 nós. Como é possível observar pela Tabela 5.3, os valores médios do tráfego de rede são baixos, devido à pequena quantidade de nós presentes.

Tabela 5.3 – Comparativo do desempenho dos algoritmos de consenso em uma blockchain linear com 100 nós.

	Tempo para criação de novos blocos (ms)	CPU (%)	RAM (MB)	Tráfego de rede (Enviados/recebidos - MB/s)
PoW	265,17	52%	**22,5**	0,611 / 0,57
PoS	156,47	24%	22,8	0,64 / 0,63
PoA	169,1	16%	23,3	**0,48 / 0,48**
PBFT	231,48	15%	23,2	1,51 / 1,49
Raft	223,01	**11%**	23,3	1,19 / 1,18
PoL	157,22	33%	26,1	0,9 / 0,8
PoB	144,23	22%	28,1	1,03 / 1,07
PoI	**130,89**	23%	28	0,98 / 0,9
dPoS	144,05	19%	29	1,15 / 1,05
PoET	268,76	37%	23,9	0,9 / 0,69

Fonte: O autor (2025).

De forma semelhante, para a configuração da blockchain com 500 nós, os dados foram sintetizados e apresentados na Tabela 5.4. Neste experimento, assim como no anterior, todos os valores de tempo de criação de blocos foram inferiores a 1000 milissegundos.

Tabela 5.4 – Comparativo do desempenho dos algoritmos de consenso em uma blockchain linear com 500 nós.

	Tempo para criação de novos blocos (ms)	CPU (%)	RAM (MB)	Tráfego de rede (Enviados/recebidos - MB/s)
PoW	828,23	53,4%	27,9	6,4 / 8,2
PoS	491,5	27%	32,1	16,8 / 7,9
PoA	598,94	20,8%	32,5	6,1 / 4,5
PBFT	705,4	20,3%	**23,1**	9,9 / 10,9
Raft	625,56	**18%**	23,8	11,3 / 12,7
PoL	575,42	43%	37,1	**4,7 / 3,4**
PoB	542,67	27%	35,1	7,2 / 8,2
PoI	474,36	31,2%	27,6	8,7 / 7,4
dPoS	**431,25**	25%	28,3	9,4 / 10,4
PoET	856,78	42%	32,4	8,2 / 7,7

Fonte: O autor (2025).

Por fim, a Tabela 5.5 apresenta o resumo dos dados coletados durante a execução da avaliação dos algoritmos de consenso para a blockchain com 1000 nós. Para esta configuração, todos os tempos de criação de blocos ficaram abaixo de 3000 milissegundos, exceto para os algoritmos PoW e PoET.

O tempo médio de criação de blocos abaixo de 3000 ms é consideravelmente inferior às blockchains públicas tradicionais, que normalmente operam em intervalos de tempo maiores: a Bitcoin gera blocos em média a cada 600 s (10 minutos) (JABŁCZYŃSKA *et al.*, 2023), e a Ethereum entre 12–15 s (KRANER *et al.*, 2023).

Tabela 5.5 – Comparativo do desempenho dos algoritmos de consenso em uma blockchain linear com 1000 nós.

	Tempo para criação de novos blocos (ms)	CPU (%)	RAM (MB)	Tráfego de rede (Enviados/recebidos - MB/s)
PoW	3713,7	58%	**23,3**	**8,88 / 8,87**
PoS	**1242,9**	34%	28,4	12,8 / 11,4
PoA	1621,9	30%	40,8	11 / 9,09
PBFT	2632,1	**26%**	25,4	17,95 / 18,59
Raft	2502,6	27,8%	25,6	18,98 / 16,58
PoL	2642,7	48%	32,4	13,9 / 14,8
PoB	2736,9	39%	33,3	10,4 / 9,92
PoI	1695	37,8%	48,3	9,72 / 8,3
dPoS	1493,6	37,2%	34	11,2 / 10,1
PoET	4152,1	56%	36,4	13,1 / 14,4

Fonte: O autor (2025).

5.6 AVALIAÇÃO DA OMNIBLOCK

O segundo conjunto de experimentos utilizou a blockchain *OmniBlock*. O objetivo dessa etapa da avaliação foi comparar o desempenho das combinações de algoritmos de consenso sugeridas pelos algoritmos genéticos simples (SGA) e o algoritmo genético adaptativo (AGA). Os dados completos obtidos durante a realização dessa avaliação de desempenho são apresentados no Apêndice C.

Inicialmente, o algoritmo SGA foi usado para definir combinações de algoritmos de consenso para cada domínio IoT. Em seguida, o algoritmo AGA foi executado, porém as taxas de mutação e cruzamento variaram segundo os parâmetros definidos na Tabela 5.2. Para cada variação da taxa, o AGA apresentou sugestões de combinações diferentes, conforme apresentado na Tabela 5.6.

Tabela 5.6 – Comparações do SGA e do AGA com diferentes configurações de TxMutate / TxCrossover.

		AGA		
	SGA	Tx. M(B) / Tx. C(B)	Tx. M(M) / Tx. C(M)	Tx. M(A) / Tx. C(A)
Smart Home	PoS-PoW	PoS-PoL	PoS-PoL	PoS-PoL
Healthcare	PoS-Raft	PoL-Raft	PoS-PoL	PoL-Raft
Smart Agriculture	PoS-PoB	PoS-PBFT	PoS-PoW	PoS-PBFT
Industrial IoT	PoA-PoW	PoA-PoW	PoA-PoET	PoA-PoS
Smart Cities	PoB-PBFT	PoS-PBFT	PoI-Raft	PoS-PBFT
Smart Logistics	PoET-PoW	PoB-PoL	PoA-PoL	PoS-Raft
Smart Energy	PoA-PoW	PoET-PoA	PoA-PoW	PoA-PoW
Internet of Vehicles	PoL-PBFT	PoS-Raft	PoI-PBFT	PoL-Raft

Legenda: Tx. M: Taxa de Mutação, Tx C: Taxa de Cruzamento; B: *Baixa*, M: *Média* e A: *Alta*.

Fonte: O autor (2025).

Como é possível observar, o algoritmo genético AGA apresenta o mesmo resultado apenas para o domínio *Smart Home*. Para os demais, as combinações sugeridas foram diferentes. Isso se deve ao fato de que, como no AGA as taxas variam durante a execução, o algoritmo tem a oportunidade de explorar mais possibilidades de cruzamento entre os algoritmos, com o objetivo de encontrar o melhor resultado possível. Diante disso, todas as combinações precisaram ser avaliadas.

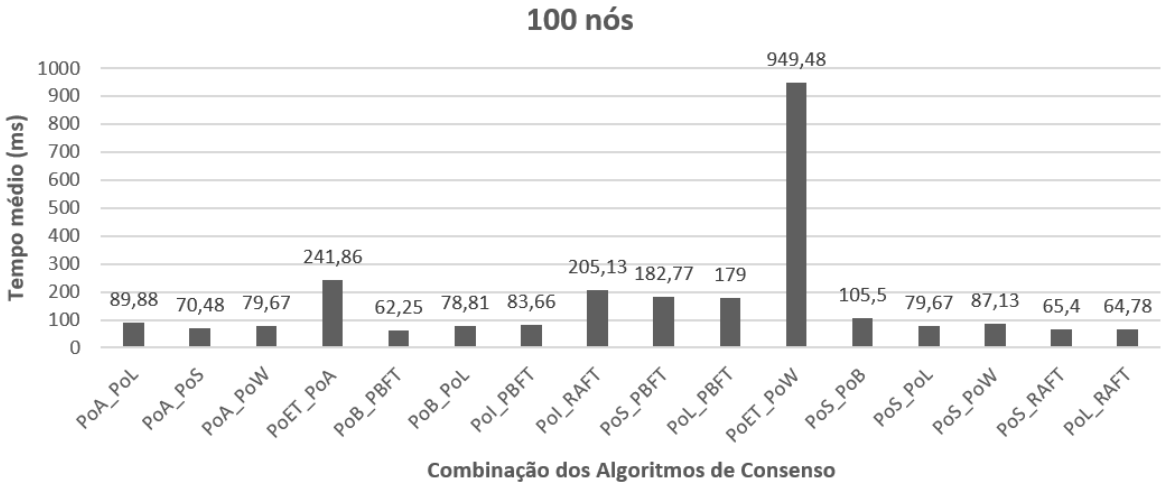
5.6.1 Tempo médio de criação de novos blocos

A primeira métrica analisada foi o tempo médio de criação de novos blocos. Esta é uma métrica importante em aplicações blockchain voltadas para IoT, pois a maioria das aplicações e dispositivos exigem respostas rápidas. Além disso, em aplicações que lidam com dados dos usuários, é desejável que os dados sejam registrados rapidamente para evitar possíveis ataques e falhas de segurança.

O tempo médio de criação de novos blocos foi comparado variando o número de nós da blockchain em 100, 500 e 1000. Realizar essas variações é importante para avaliar a escalabilidade da *OmniBlock*. A Figura 5.14 mostra a comparação do tempo de criação de bloco para a blockchain com 100 nós.

Neste experimento, as combinações de algoritmos de consenso que criaram blocos mais rapidamente foram PoB-PBFT, seguidas por PoS-Raft. Essas combinações integram o consumo eficiente de recursos computacionais do PoB e do PoS com a agilidade e tolerância a falhas do PBFT e Raft, respectivamente, o que pode ser eficiente para aplicações IoT.

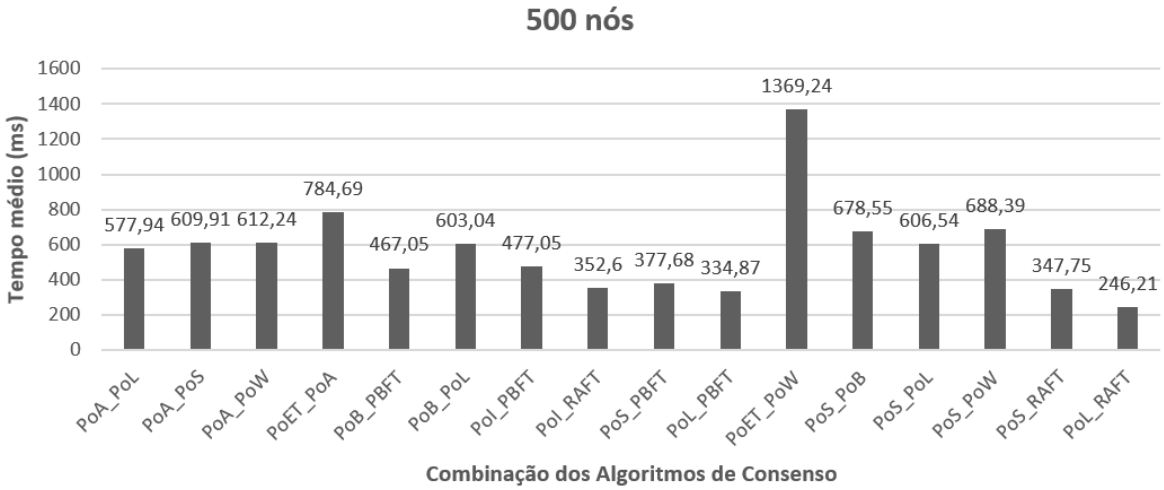
Figura 5.14 – Tempo médio para criar novos blocos usando 100 nós



Fonte: O autor (2025).

A Figura 5.15 mostra o resultado para a configuração com 500 nós. Nesta configuração, houve um aumento considerável no tempo médio para criação de novos nós, em comparação com o experimento anterior. Para a configuração anterior, com 100 nós, o tempo médio total foi de 164,09 ms, neste experimento houve um aumento de 247,8%, passando para 570,85 ms. No entanto, todas as combinações de algoritmos de consenso apresentam tempos de criação de blocos inferiores a 1000 milissegundos, exceto a combinação PoET-PoW, cujo tempo foi de 1369 milissegundos.

Figura 5.15 – Tempo médio para criar novos blocos usando 500 nós



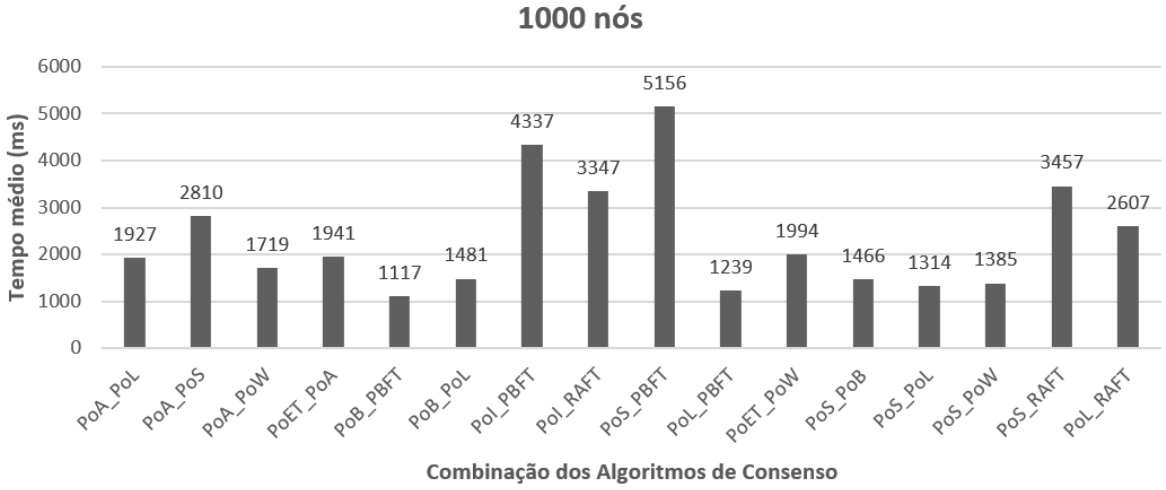
Fonte: O autor (2025).

Por fim, o número de nós da blockchain foi aumentado para 1000. Assim como no experi-

mento anterior, houve um aumento no tempo médio de criação de novos blocos em todas as combinações. O tempo médio total passou de 570,85 ms para 2331,08 ms, o que representa um aumento de 308,3%.

A combinação que apresentou o menor tempo para criação de blocos foi PoB-PBFT, seguida por PoL-PBFT, e PoS-PoL, onde o tempo médio dessas combinações foi de aproximadamente 1200 ms. As combinações de algoritmo de consenso que apresentaram os maiores tempos foram PoS-PBFT (5456 ms) e PoI-PBFT (4337 ms). Estes resultados são mostrados na Figura 5.16.

Figura 5.16 – Tempo médio para criar novos blocos usando 1000 nós



Fonte: O autor (2025).

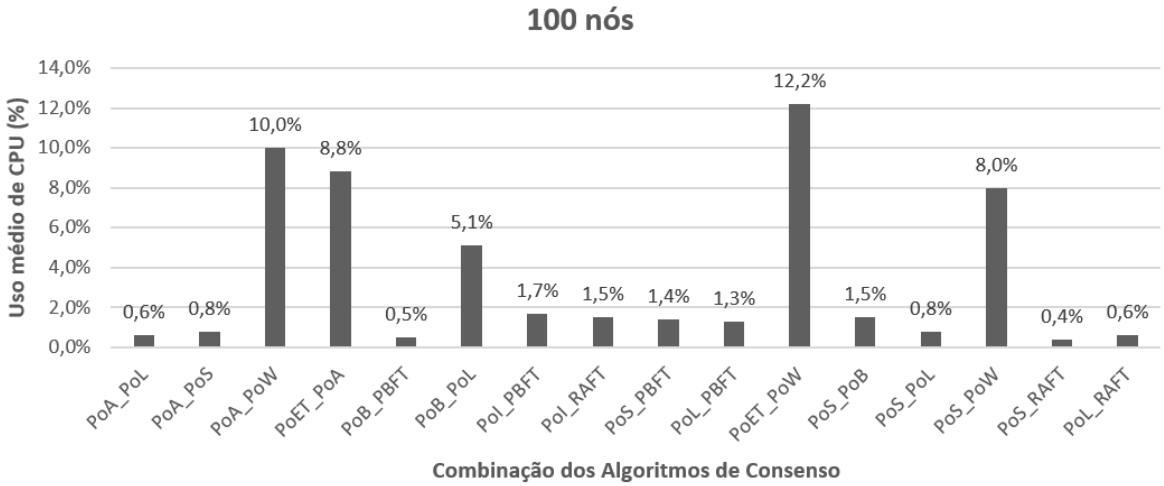
5.6.2 Consumo médio de CPU

A próxima métrica analisada foi o consumo médio de CPU durante a execução das combinações de algoritmos de consenso. Nas aplicações IoT, é comum que os dispositivos tenham poucos recursos computacionais ou dependam de baterias para funcionar. Portanto, ao implementar soluções baseadas em blockchain, é importante avaliar o consumo de CPU, para não sobrecarregar o processamento dos dispositivos.

Para analisar esta métrica, foi levado em consideração apenas o nó principal da blockchain. Ou seja, aquele utilizado para criar e validar o novo bloco no momento da execução do teste. Esta decisão foi motivada pelo fato de que apenas o nó validador apresentou valores significativos de consumo de CPU no momento da criação de novos blocos em experimentos preliminares.

A Figura 5.17 mostra o resultado deste experimento, considerando a blockchain *OmniBlock* com 100 nós. Nestes resultados, é possível observar que as combinações com maior consumo médio de CPU foram aquelas que utilizaram o algoritmo de consenso PoW. Isso ocorre porque o PoW é um algoritmo de consenso que demanda mais recursos computacionais. No entanto, a sua utilização proporciona um maior nível de segurança, pois ele exige muito poder computacional para violá-lo e alterar os dados, em comparação aos outros algoritmos (LASLA *et al.*, 2022).

Figura 5.17 – Consumo médio de CPU usando 100 nós.

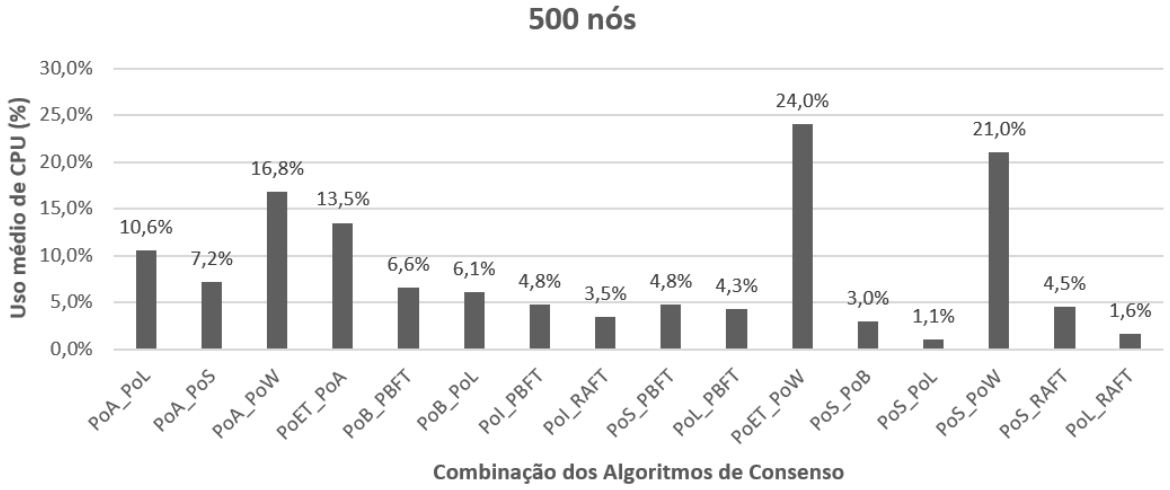


Fonte: O autor (2025).

No experimento seguinte, o consumo médio de CPU foi medido para a *OmniBlock* com 500 nós, conforme mostrado na Figura 5.18. Os valores obtidos apresentaram aumentos em relação ao experimento anterior. A média total passou de 3,5% (100 nós) para 8,3% (500 nós), o que significa um aumento de 137,1%.

As combinações que utilizam os algoritmos de consenso PoW e PoET continuam sendo as que apresentam o maior consumo médio de CPU. Porém, as demais combinações continuam apresentando baixo consumo de CPU (menos de 10%), o que é desejável em IoT (SAGIRLAR *et al.*, 2018).

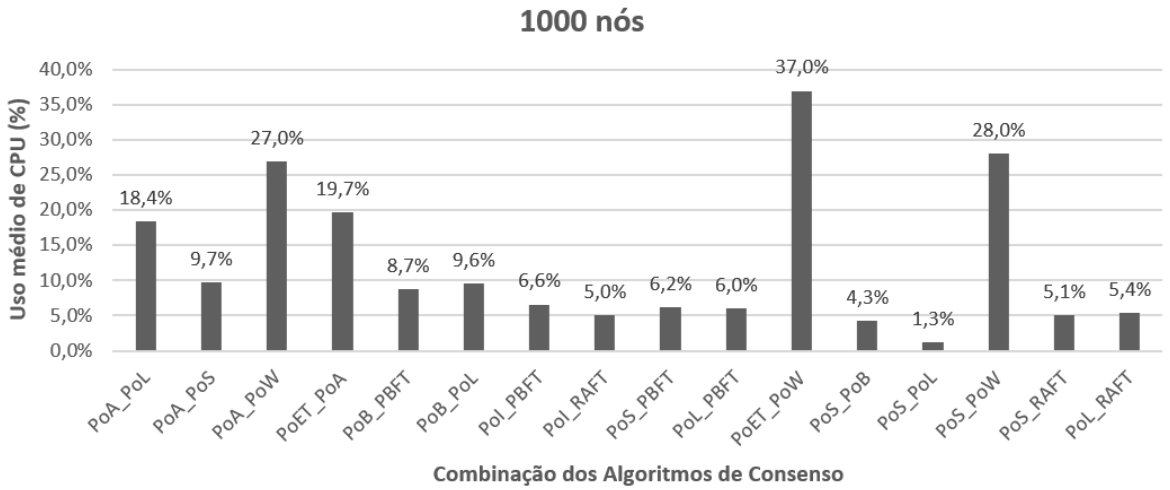
Figura 5.18 – Consumo médio de CPU usando 500 nós.



Fonte: O autor (2025).

A Figura 5.19 apresenta os resultados dos experimentos realizados para a configuração da blockchain com 1000 nós. Nestes experimentos, as combinações que apresentaram menores valores médios de consumo de CPU foram a PoS-PoL e PoS-PoB, que tiveram resultados abaixo de 5% do consumo de CPU.

Figura 5.19 – Consumo médio de CPU usando 1000 nós.



Fonte: O autor (2025).

5.6.3 Consumo médio de memória

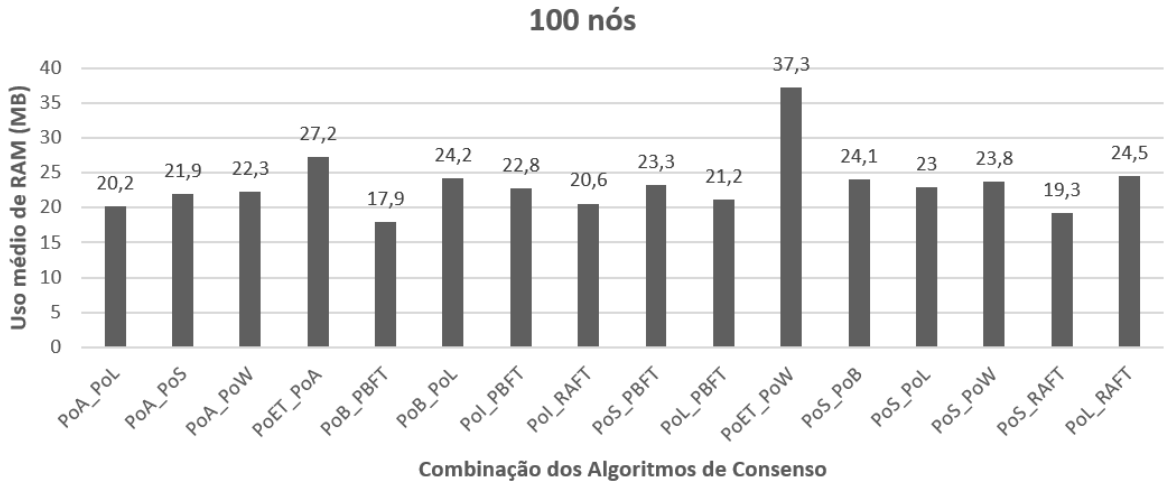
A próxima métrica analisada foi o consumo médio de memória pelos nós da blockchain. Em IoT, gerenciar os recursos de memória disponíveis é uma tarefa fundamental para que não

haja sobrecargas nos dispositivos.

Os experimentos foram realizados para todas as combinações dos algoritmos de consenso, e a média foi calculada levando em consideração todos os nós presentes na blockchain. Diferentemente da análise de CPU, onde o processamento ocorre apenas no nó validador, aqui há consumo de memória em todos os nós, pois eles precisam verificar e incluir o novo bloco criado em suas respectivas cópias da blockchain.

A Figura 5.20 apresenta os valores de consumo médio de memória para a blockchain *OmniBlock* contendo 100 nós. Neste experimento, a combinação que obteve o menor consumo de memória foi entre os algoritmos PoB-PBFT, enquanto que a combinação entre os algoritmos PoET-PoW teve o maior consumo.

Figura 5.20 – Consumo médio de memória usando 100 nós.

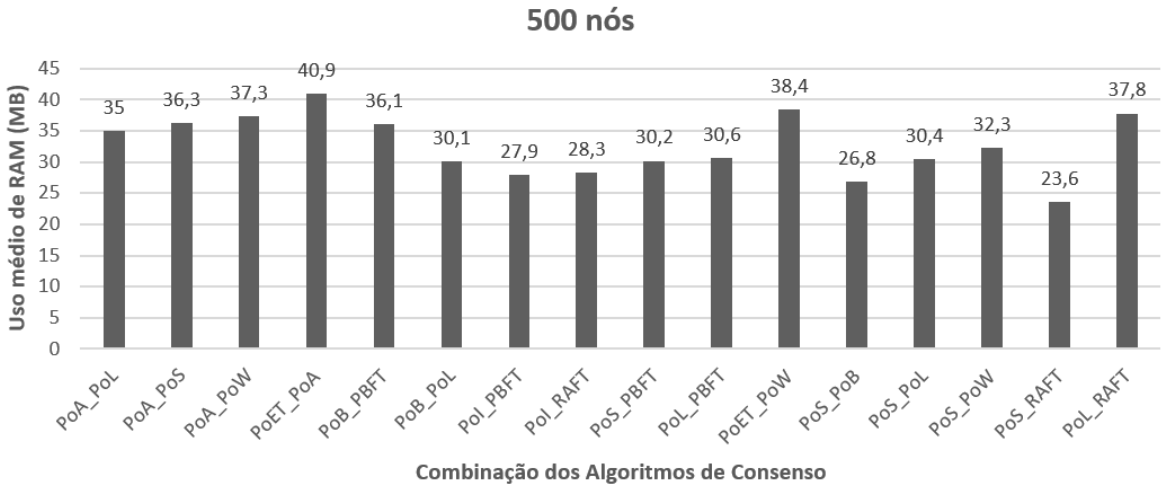


Fonte: O autor (2025).

O próximo experimento levou em consideração a configuração da blockchain com 500 nós. Os resultados são apresentados na Figura 5.21. Os valores obtidos foram mais altos que no experimento anterior. O consumo médio total de RAM passou de 23,35 MB (100 nós) para 32,65 MB neste experimento, totalizando um aumento de 39,8%.

A combinação entre os algoritmos PoS-Raft apresentou o menor consumo de memória nos experimentos. Isso ocorre porque ambos os algoritmos são projetados para serem eficientes em termos de consumo de recursos computacionais.

Figura 5.21 – Consumo médio de memória usando 500 nós.

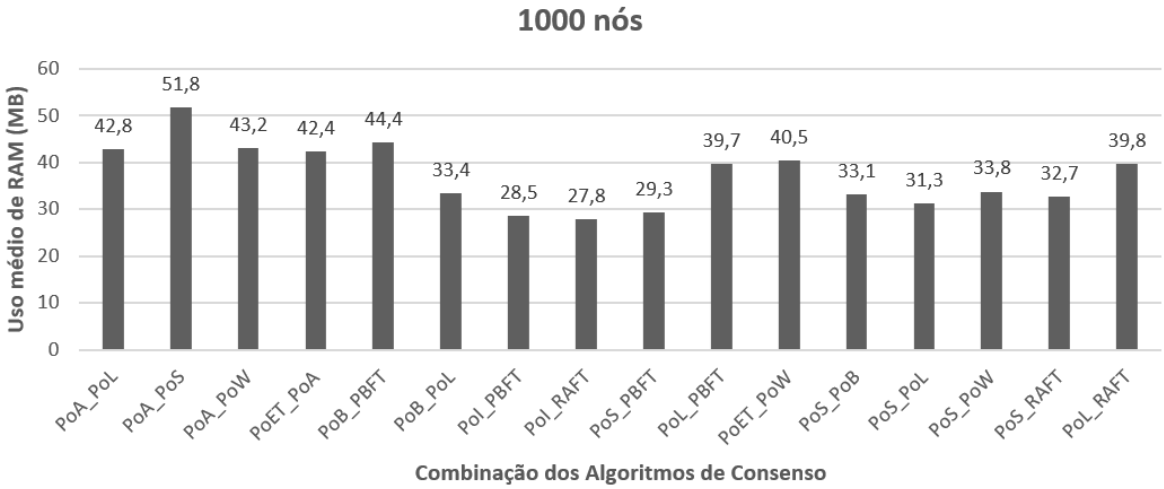


Fonte: O autor (2025).

Por fim, para a configuração da blockchain *OmniBlock* contendo 1000 nós, o consumo médio de memória sofreu variações, em relação aos experimentos anteriores, devido ao maior número de nós presentes na rede. A média total passou de 32,62 MB (100 nós) para 37,15 MB, aumentando 13,9%.

A Figura 5.22 mostra os resultados obtidos neste experimento. As combinações PoI-PBFT e PoI-Raft obtiveram os melhores resultados, já que o PBFT e o Raft são algoritmos que buscam otimizar o consumo de recursos.

Figura 5.22 – Consumo médio de memória usando 1000 nós.



Fonte: O autor (2025).

5.6.4 Tráfego médio de Rede

A última métrica considerada nesta avaliação de desempenho foi tráfego médio de rede. Esta também é uma métrica importante em blockchain para a IoT.

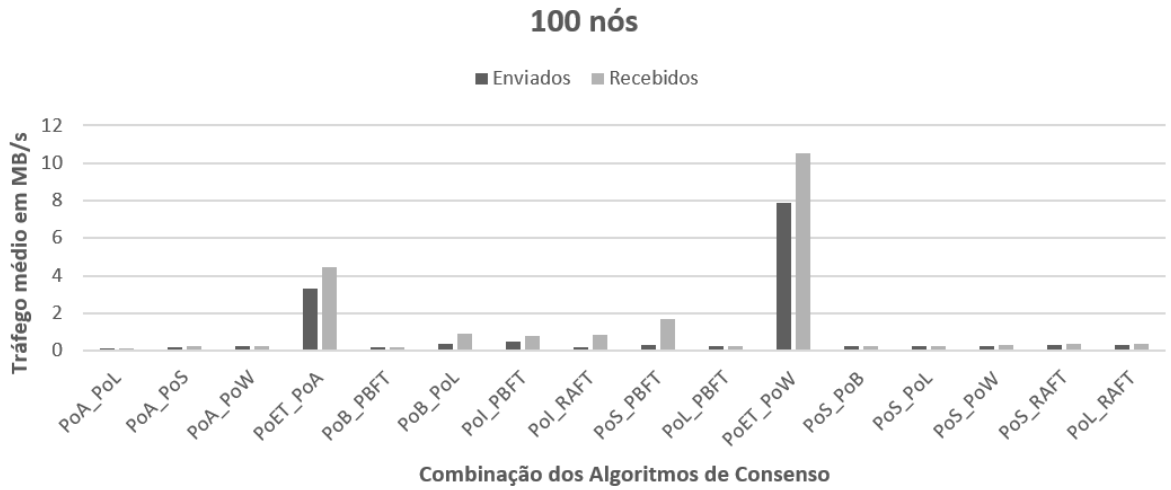
Em cenários onde há intensa geração e troca de dados entre os dispositivos, é necessário avaliar se há a possibilidade de sobrecarga da rede, o que poderia ocasionar perda de desempenho em toda a blockchain.

Assim como nos experimentos anteriores, todas as combinações de algoritmos de consenso foram avaliadas, considerando a blockchain com 100, 500 e 1000 nós. Para cada combinação, foram consideradas as quantidades de Bytes/s enviados e recebidos pelos nós.

A Figura 5.23 apresenta as médias de tráfego de rede obtidas para a blockchain com 100 nós. Neste experimento inicial, os valores obtidos foram consideravelmente baixos (média de 1,14 MB/s), devido à pequena quantidade de nós presentes.

As combinações que geraram mais tráfego foram aquelas que utilizaram o algoritmo PoET. Isso ocorre porque esse algoritmo realiza muitas trocas de mensagens entre os pares da rede, antes de validar um novo bloco e inseri-lo na blockchain.

Figura 5.23 – Tráfego médio de rede usando 100 nós.

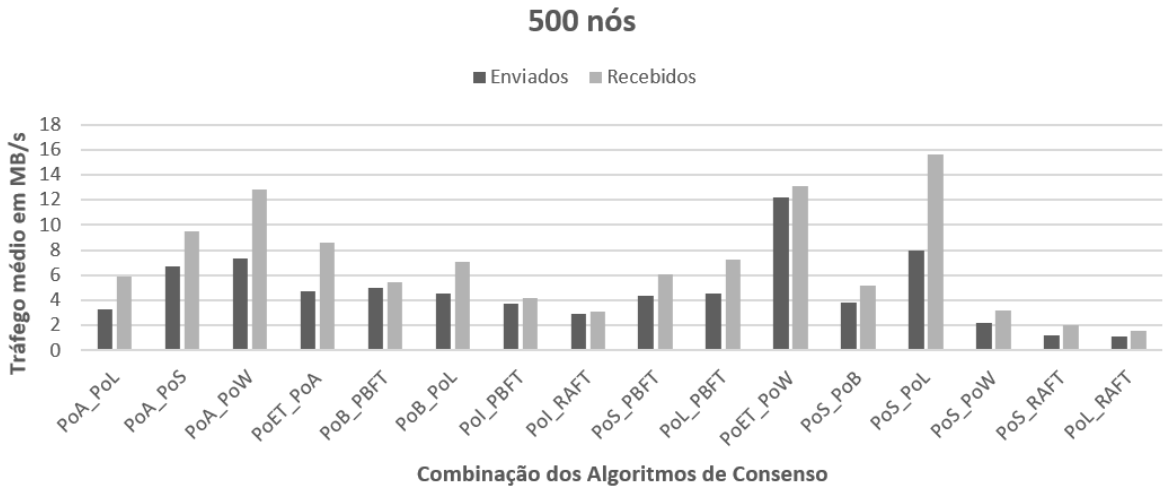


Fonte: O autor (2025).

O próximo experimento utilizou a configuração de blockchain com 500 nós. A Figura 5.24 detalha os resultados obtidos nesse experimento. Ao aumentar o número de nós da blockchain, é possível notar um aumento nos valores médios obtidos em relação ao experimento anterior. A média total passou de 1,14 MB/s para 5,82 MB/s, aumentando 410,5%. Neste caso, as

combinações que obtiveram a menor quantidade de dados enviados e recebidos foram aquelas que utilizaram o algoritmo Raft.

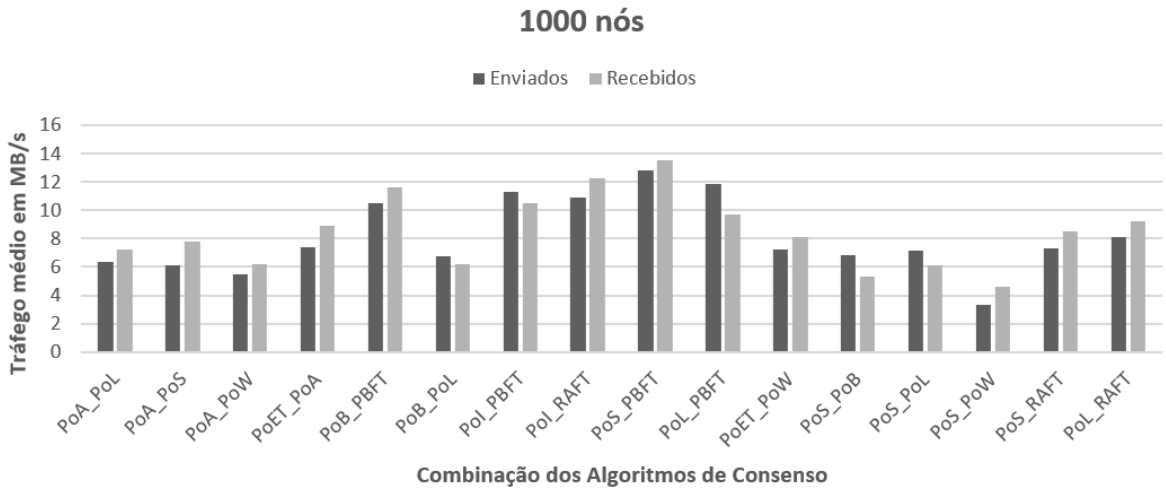
Figura 5.24 – Tráfego médio de rede usando 500 nós.



Fonte: O autor (2025).

Por fim, a blockchain foi configurada com 1000 nós. As médias de tráfego de rede foram novamente analisadas para todas as combinações, e os resultados são apresentados na Figura 5.25.

Figura 5.25 – Tráfego médio de rede usando 1000 nós.



Fonte: O autor (2025).

A partir dos gráficos, é possível observar que as combinações que apresentam menor tráfego de rede são aquelas que utilizam o algoritmo PoW, integrado com outros como PoA ou PoS. Isso acontece porque, no PoW, o novo bloco é criado pelo nó validador e apenas repassado

aos demais, sem a necessidade de muitas verificações pelos outros nós, o que reduz o tráfego de dados na rede.

Assim como na execução dos experimentos realizados anteriormente na blockchain linear (Seção 5.5), os dados obtidos foram organizados em tabelas, para facilitar a visualização e análise. A Tabela 5.7 apresenta um sumário das quatro métricas consideradas na avaliação da *OmniBlock* contendo 100 nós.

Como é possível observar, valores médios do tráfego de rede neste experimento são baixos, devido à pequena quantidade de nós presentes na blockchain. Os valores de consumo de médio de CPU ficaram abaixo de 10% para todas as combinações, exceto para PoET-PoW.

Tabela 5.7 – Comparativo do desempenho das combinações de algoritmos de consenso na blockchain *OmniBlock* com 100 nós.

	Tempo para criação de novos blocos (ms)	CPU (%)	RAM (MB)	Tráfego de rede (Enviados/recebidos - MB/s)
PoA-PoL	89,88	0,6 %	20,2	**0,11 / 0,11**
PoA-PoS	70,48	0,8 %	21,9	0,21 / 0,23
PoA-PoW	79,67	10,0 %	22,3	0,23 / 0,24
PoET-PoA	241,86	8,8 %	27,2	3,3 / 4,4
PoB-PBFT	**62,25**	0,5 %	**17,9**	0,19 / 0,2
PoB-PoL	78,81	5,1 %	24,2	0,35 / 0,9
PoI-PBFT	83,66	1,7 %	22,8	0,49 / 0,82
PoI-Raft	205,13	1,5 %	20,6	0,19 / 0,85
PoS-PBFT	182,77	1,4 %	23,3	0,28 / 1,7
PoL-PBFT	179,0	1,3 %	21,2	0,25 / 0,27
PoET-PoW	949,48	12,2 %	37,3	7,88 / 10,5
PoS-PoB	105,5	1,5 %	24,1	0,25 / 0,27
PoS-PoL	79,67	0,8 %	23,0	0,27 / 0,27
PoS-PoW	87,13	8,0 %	23,8	0,28 / 0,28
PoS-Raft	65,4	**0,4 %**	19,3	0,33 / 0,37
PoL-Raft	64,78	0,6 %	24,5	0,33 / 0,35

Fonte: O autor (2025).

De forma semelhante, para a configuração da blockchain *OmniBlock* contendo 500 nós, os dados foram sintetizados e apresentados na Tabela 5.8.

Neste experimento, assim como no experimento anterior, todas as combinações de algoritmos de consenso obtiveram os valores de tempo médio para criação de novos blocos inferior a 1000 milissegundos, exceto a combinação entre os algoritmos PoET-PoW, cujo valor foi de 1369 milissegundos.

Quanto ao consumo médio de CPU, as combinações que obtiveram os menores valores foram PoS-PoL (1,1%), PoL-Raft (1,6%). O maior consumo de CPU foi da combinação PoET-PoW.

Para este experimento, de forma semelhante ao anterior, as combinações que tiveram os menores valores médios de tráfego de rede foram PoS-Raft, e PoL-Raft.

Tabela 5.8 – Comparativo do desempenho das combinações de algoritmos de consenso na blockchain *OmniBlock* com 500 nós.

	Tempo para criação de novos blocos (ms)	CPU (%)	RAM (MB)	Tráfego de rede (Enviados/recebidos - MB/s)
PoA-PoL	577,94	10,6 %	35,0	3,28 / 5,93
PoA-PoS	609,91	7,2 %	36,3	6,72 / 9,5
PoA-PoW	612,24	16,8 %	37,3	7,32 / 12,8
PoET-PoA	784,69	13,5 %	40,9	4,72 / 8,64
PoB-PBFT	467,05	6,6 %	36,1	4,97 / 5,47
PoB-PoL	603,04	6,1 %	30,1	4,57 / 7,02
PoI-PBFT	477,05	4,8 %	27,9	3,71 / 4,14
PoI-Raft	352,6	3,5 %	28,3	2,93 / 3,1
PoS-PBFT	377,68	4,8 %	30,2	4,37 / 6,09
PoL-PBFT	334,87	4,3 %	30,6	4,58 / 7,24
PoET-PoW	1369,24	24,0 %	38,4	12,2 / 13,1
PoS-PoB	678,55	3,0 %	26,8	3,83 / 5,17
PoS-PoL	606,54	**1,1 %**	30,4	7,97 / 15,6
PoS-PoW	688,39	21,0 %	32,3	2,2 / 3,19
PoS-Raft	347,75	4,5 %	**23,6**	1,17 / 2,03
PoL-Raft	**246,21**	1,6 %	37,8	**1,11 / 1,6**

Fonte: O autor (2025).

Por fim, a Tabela 5.9 apresenta o resumo dos dados coletados durante a execução da avaliação de desempenho das combinações de algoritmos de consenso para a configuração da *OmniBlock* com 1000 nós.

Esta foi a configuração com maiores variações nos valores das métricas, devido ao aumento no número de nós. Nestes experimentos, os valores dos tempos de criação de blocos tiveram aumento, porém todos ficaram abaixo de 5000 milissegundos, exceto a combinação PoS-PBFT (5156 ms). Esses valores de tempo de criação de blocos podem ser considerados baixos e satisfatórios, uma vez que outras blockchains privadas, como a Hyperledger Fabric (ANDROULAKI *et al.*, 2018), podem apresentar atrasos de até 134 s para a propagação de um novo bloco (NGUYEN *et al.*, 2019).

Quanto ao consumo médio de CPU, a combinação que obteve o maior valor foi a PoET-PoW (37%), seguida pelas combinações PoS-PoW (28%) e PoA-PoW (27%). Isso se deve ao fato de o algoritmo de consenso PoW exigir um esforço computacional maior para ser executado. Todas as demais combinações de algoritmos de consenso obtiveram valores médios de consumo de CPU abaixo de 20%.

Esses valores são considerados aceitáveis e semelhantes a outras propostas de blockchains privadas (YANG *et al.*, 2022), onde são encontrados valores médios de consumo de CPU em torno de 23%. De acordo com a documentação oficial da blockchain Hyperledger Fabric (Hyperledger Foundation, 2025), é recomendado manter o uso de CPU abaixo de 70–80%.

Tabela 5.9 – Comparativo do desempenho das combinações de algoritmos de consenso na blockchain *Omni-Block* com 1000 nós.

	Tempo para criação de novos blocos (ms)	CPU (%)	RAM (MB)	Tráfego de rede (Enviados/recebidos - MB/s)
PoA-PoL	1927	18,4 %	42,8	6,39 / 7,2
PoA-PoS	2810	9,7 %	51,8	6,1 / 7,8
PoA-PoW	1719	27,0 %	43,2	**5,5 / 6,2**
PoET-PoA	1941	19,7 %	42,4	7,39 / 8,9
PoB-PBFT	**1117**	8,7 %	44,4	10,53 / 11,6
PoB-PoL	1481	9,6 %	33,4	6,79 / 6,19
PoI-PBFT	4337	6,6 %	28,5	11,32 / 10,5
PoI-Raft	3347	5,0 %	**27,8**	10,9 / 12,24
PoS-PBFT	5156	6,2 %	29,3	12,84 / 13,57
PoL-PBFT	1239	6,0 %	39,7	11,86 / 9,71
PoET-PoW	1994	37,0 %	40,5	7,2 / 8,1
PoS-PoB	1466	4,3 %	33,1	6,83 / 5,3
PoS-PoL	1314	**1,3 %**	31,3	7,12 / 6,1
PoS-PoW	1385	28,0 %	33,8	3,34 / 4,63
PoS-Raft	3457	5,1 %	32,7	7,28 / 8,52
PoL-Raft	2607	5,4 %	39,8	8,1 / 9,2

Fonte: O autor (2025).

5.7 COMPARATIVO DOS ALGORITMOS GENÉTICOS

Devido às variações nas taxas de mutação e cruzamento, os algoritmos genéticos SGA e AGA apresentam diferentes sugestões de combinações, conforme mostrado na Tabela 5.6. Portanto, com base na avaliação de desempenho realizada em cada uma das combinações, é possível fazer uma comparação e analisar qual algoritmo genético apresentou os melhores

resultados.

A Tabela 5.10 apresenta as melhores sugestões de combinações apresentadas pelo SGA e pelo AGA, considerando as variações das taxas de mutação e cruzamento em *Baixo*, *Médio* e *Alto*, respectivamente.

Tabela 5.10 – Melhores combinações de algoritmos de consenso sugeridas pelos algoritmos genéticos

		AGA		
	SGA	Tx. M(B) / Tx. C(B)	Tx. M(M) / Tx. C(M)	Tx. M(A) / Tx. C(A)
Smart Home		PoS-PoL ✓	PoS-PoL ✓	PoS-PoL ✓
Healthcare			PoS-PoL ✓	
Smart Agriculture			PoS-PoW ✓	
Industrial IoT	PoA-PoW ✓			
Smart Cities			PoI-Raft ✓	
Smart Logistics		PoB-PoL ✓		
Smart Energy		PoET-PoA ✓		
Internet of Vehicles		PoS-Raft ✓		

Legenda: Tx. M: Taxa de Mutação, Tx C: Taxa de Cruzamento; B: *Baixa*, M: *Média* e A: *Alta*.

Fonte: O autor (2025).

Para esta comparação, as integrações foram ranqueadas de acordo com os valores de cada métrica. Por exemplo, para o domínio de *Smart Cities*, a combinação sugerida pelo SGA foi PoB-PBFT (Ver Tabela 5.6). As sugestões apresentadas pelo AGA foram: PoS-PBFT, para taxas de mutação e cruzamento *Baixa*; PoI-Raft, para taxas de mutação e cruzamento *Média*; e PoS-PBFT para taxas de mutação e cruzamento *Alta*.

Com base nos resultados dos experimentos, a combinação PoI-Raft foi a mais eficiente em duas das quatro métricas consideradas: consumo de CPU e memória. Por isso, essa integração foi marcada (✓) como a mais eficiente para este domínio na Tabela 5.10. O mesmo foi realizado para todos os demais domínios IoT considerados.

O SGA apresentou melhor resultado apenas em um dos domínios: *Industrial IoT*. Para este domínio, a combinação sugerida foi PoA-PoW, que se mostrou mais eficiente nas métricas de tempo para criação de novos blocos e tráfego de rede em comparação com as outras sugestões apresentadas pelo AGA.

Para o domínio de *Smart Home*, o AGA sugeriu a mesma combinação para as variações de taxas de mutação e cruzamento (PoS-PoL). Ainda assim, essa combinação foi mais eficiente nas métricas de tempo, CPU e memória do que a sugestão do algoritmo SGA (PoS-PoW).

Com base nestes resultados, o *Adaptive Genetic Algorithm* (AGA) apresenta resultados mais adequados na grande maioria dos casos. Isso se deve ao fato de que as variações realizadas

nas taxas de mutação e cruzamento contribuem para a busca por melhores soluções.

Ainda de acordo com os experimentos realizados, onde as taxas de mutação e cruzamento foram variadas em três níveis, os valores *Baixo* ou *Médio* para essas taxas produzem melhores resultados. Ao definir essas taxas para valores *Alto*, apenas uma das integrações sugeridas obteve melhor resultado (PoS-PoL).

Manter as taxas iniciais de mutação e cruzamento *Alta* faz com que o algoritmo descarte rapidamente boas soluções, comportando-se quase como uma busca puramente aleatória e atrasando ou até impedindo a convergência para soluções de boa qualidade.

Assim, a utilização de valores *Baixo* ou *Médio* permite equilibrar a exploração de novas regiões do espaço de busca com a exploração refinada de soluções já identificadas, otimizando o desempenho do algoritmo genético.

5.8 ANÁLISES E DISCUSSÕES

Esta seção apresenta uma análise detalhada da comparação da *OmniBlock*com uma blockchain linear, e dos algoritmos genéticos utilizados.

5.8.1 Comparação de desempenho entre a blockchain linear e a *OmniBlock*

Nesta comparação, os valores médios das métricas (ver Seção 5.2) foram calculados para cada configuração das blockchains, i.e., 100, 500 e 1000 nós. A Tabela 5.11 apresenta um resumo dos dados utilizados neste comparativo.

Tabela 5.11 – Comparativo de desempenho entre a blockchain linear e a *OmniBlock*

	Tempo para criação de novos blocos (ms)	CPU (%)	RAM (MB)	Tráfego de rede (Enviados/recebidos - MB/s)
	B.L. / OmniBlock	B.L. / *OmniBlock*	B.L. / *OmniBlock*	B.L. / *OmniBlock*
100 nós	189,0 / **134,1**	25,2% / **3,5%**	25,0 / **23,4**	**0,9/0,9** / 0,9/1,4
500 nós	613,0 / **470,9**	30,8% / **8,3%**	**30,0** / 32,6	7,8/8,1 / **4,7/6,9**
100 nós	2443,3 / **2131,1**	39,4% / **12,4%**	**34,5** / 37,2	12,8/12,2 / **8,1/8,5**

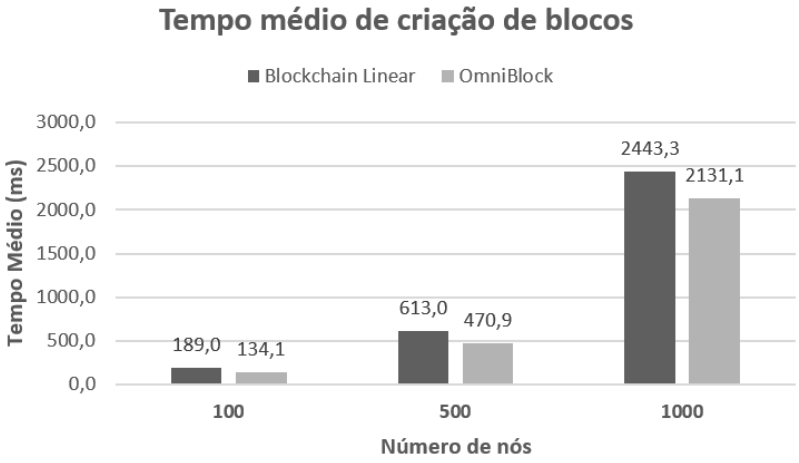
Legenda: B.L.: Blockchain Linear.

Fonte: O autor (2025).

A Figura 5.26 mostra a comparação do tempo médio para criação de blocos pelas combinações de algoritmos de consenso da *OmniBlock* e da blockchain linear.

De acordo com os resultados, é possível observar que o tempo médio de criação de blocos na *OmniBlock* é menor em todas as configurações de nós da blockchain.

Figura 5.26 – Comparação do tempo médio para criação de novos blocos na blockchain linear e na *OmniBlock*

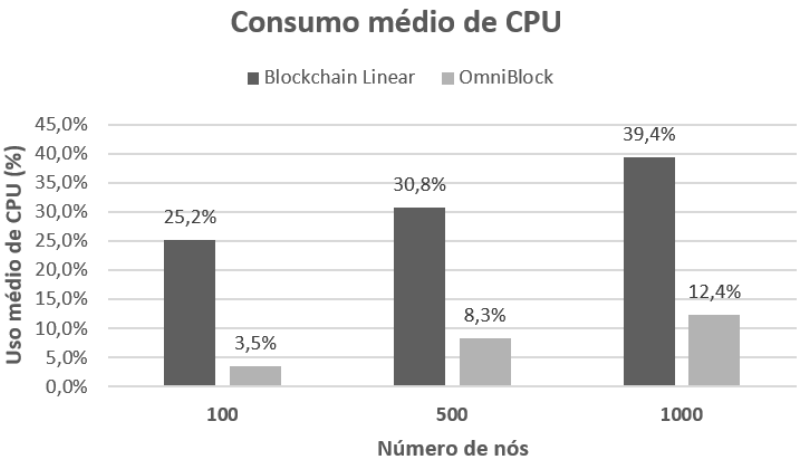


Fonte: O autor (2025).

Em seguida, calculou-se o consumo médio de CPU de todas as combinações de algoritmos da *OmniBlock* e da blockchain linear. Os resultados são apresentados na Figura 5.27.

O consumo médio de CPU na *OmniBlock* foi consideravelmente menor do que na blockchain linear. Por exemplo, para mil nós, a redução do consumo de CPU passou de 39,4% para 12,4%, uma redução de de 86%. Isso ocorre porque, na estrutura em DAG, a validação é feita em paralelo por vários nós, reduzindo o consumo médio de CPU por nó. Já no modelo linear, todo o processamento é sequencial, concentrando e elevando o uso de CPU em cada nó.

Figura 5.27 – Comparação do uso médio de CPU na blockchain linear e na *OmniBlock*



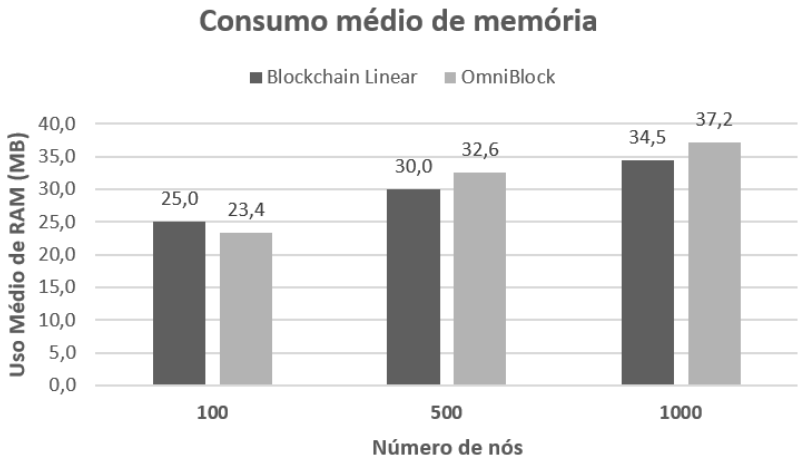
Fonte: O autor (2025).

A próxima análise compara o consumo de memória de ambas as blockchains. Os resultados são apresentados na Figura 5.28. Para esta avaliação, o consumo de memória da *OmniBlock* foi menor apenas na configuração com 100 nós. Nas outras configurações, a blockchain linear possui um consumo de memória ligeiramente menor (e.g. 7,8% para a configuração com mil nós).

Este era um resultado esperado, pois em uma blockchain baseada em DAG, cada nó precisa armazenar não apenas uma sequência linear de blocos, mas também múltiplos ramos de transações em paralelo, mantendo um grafo completo que cresce conforme novas transações são adicionadas.

Isso aumenta o consumo de memória em comparação a uma estrutura de blockchain tradicional, na qual cada nó aponta apenas para o bloco anterior e não requer o armazenamento do grafo.

Figura 5.28 – Comparação do uso médio de memória na blockchain linear e na *OmniBlock*

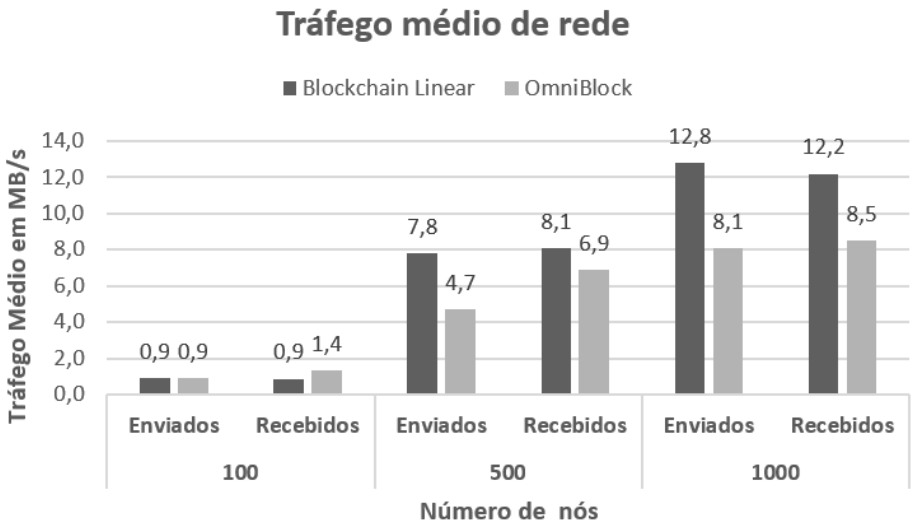


Fonte: O autor (2025).

A última análise comparou o tráfego de rede da blockchain tradicional e da *OmniBlock*. Os resultados são apresentados na Figura 5.29. Também nesta análise, a *OmniBlock* apresentou melhores resultados do que a blockchain linear.

O desempenho da *OmniBlock* vem do fato de que, na DAG, cada transação aponta para duas transações-pais, evitando a propagação de blocos inteiros e a reescrita do histórico linear. Com isso, ocorrem menos trocas de dados entre os nós do que em uma blockchain linear, o que reduz o tráfego de rede.

Figura 5.29 – Comparativo do tráfego médio de rede na blockchain linear e na *OmniBlock*



Fonte: O autor (2025).

Com base nestes resultados, é possível constatar que o uso de uma blockchain baseada em DAG, com múltiplas combinações de algoritmos de consenso, que foram sugeridas pelos algoritmos genéticos, apresenta um desempenho melhor do que o de uma blockchain linear, contendo apenas algoritmos de consenso isolados. De acordo com estes resultados, a *OmniBlock* melhorou o tempo de criação de blocos, consumo de CPU e tráfego de rede, em comparação a uma blockchain linear.

5.8.2 Combinações mais indicadas para Internet das Coisas

Com base nos resultados dos experimentos e nas métricas consideradas, algumas combinações de algoritmos de consenso são mais promissoras para melhorar o desempenho no registro de dados da Internet das Coisas. Esta análise foi realizada observando os resultados da configuração com 1000 nós, pois foi nesta configuração que houve variações mais significativas nas métricas. Além disto, os resultados podem se aproximar mais de um cenário real, onde há um grande número de nós conectados na blockchain.

Se a aplicação requer respostas rápidas e um tempo de confirmação de transação curto, as combinações entre os algoritmos de consenso PoB-PBFT e PoL-PBFT são as mais adequadas, pois, de acordo com os experimentos realizados, apresentam os menores tempos médios de criação de novos blocos (aproximadamente 1200 ms).

Algumas aplicações IoT utilizam dispositivos com baixa capacidade de processamento e,

por isso, requerem baixo consumo de CPU. Para esta métrica, a maioria das combinações sugeridas pelos algoritmos genéticos apresentou baixo consumo de CPU (inferior a 10%).

As combinações que apresentaram menor consumo de CPU foram entre os algoritmos de consenso PoS-PoL (1,3%) e PoS-PoB (4,3%). Apenas cinco combinações apresentaram consumo de CPU acima de 10%: PoET-PoW, PoS-PoW, PoA-PoW, PoET-PoA e PoA-PoL. Isso ocorre devido ao fato de que essas combinações utilizam algoritmos conhecidos por seu alto consumo de recursos computacionais, como PoW. Portanto, podem não ser combinações viáveis para cenários de IoT onde os recursos computacionais são limitados.

Em muitos casos, também é necessário manter o consumo de memória baixo para evitar a sobrecarga dos nós da blockchain. As combinações de algoritmos de consenso que apresentaram o menor consumo médio de memória foram PoI-Raft (27,8 MB) e PoI-PBFT (28,5 MB). Isso ocorre porque essas combinações utilizam algoritmos conhecidos por seu consumo eficiente de recursos computacionais, como o PBFT e o Raft.

Uma solução de blockchain para Internet das Coisas também deve apresentar um tráfego moderado de rede. Caso contrário, há a possibilidade de sobrecarga da rede e aumento da latência. Dentre as combinações dos algoritmos de consenso avaliados, as que apresentaram o menor tráfego de rede foram PoS-PoW e PoA-PoW. Os maiores tráfegos de rede ocorrem em combinações que utilizam os algoritmos PBFT e Raft.

A escalabilidade das combinações de algoritmos de consenso é crucial para atender à crescente demanda por transações em ambientes de IoT. Combinações como PoI-Raft e PoB-PBFT podem oferecer uma boa escalabilidade. A combinação PoI-Raft proporciona uma escalabilidade horizontal, permitindo que a rede cresça com mais nós devido à eficiência do Raft em alcançar consenso distribuído. A combinação PoB-PBFT oferece uma escalabilidade com melhor consumo de energia e menor uso de recursos computacionais, algo desejável para cenários com grandes volumes de dados e recursos limitados.

No entanto, as combinações que utilizam algoritmos como o PoW tendem a ser menos escaláveis devido à necessidade de um alto consumo de recursos computacionais, tornando-as mais adequadas para cenários onde a segurança é prioridade em relação ao desempenho e ao crescimento.

5.9 CONSIDERAÇÕES FINAIS

Este capítulo apresentou a avaliação experimental da *OmniBlock*. Inicialmente, foram definidos os objetivos, métricas, parâmetros e fatores. Foi realizada uma avaliação comparativa entre o desempenho da *OmniBlock* e uma blockchain linear. As métricas consideradas nesta avaliação foram tempo para criação de novos blocos, consumo de CPU e memória e tráfego de rede. Por fim, foi realizado um comparativo entre os algoritmos genéticos variando as taxas de mutação e cruzamento. No próximo capítulo serão apresentadas as conclusões, as contribuições e os trabalhos futuros desta tese.

6 CONCLUSÕES E TRABALHOS FUTUROS

Este capítulo revisita as questões de pesquisa consideradas no desenvolvimento desta tese, destaca as principais contribuições e apresenta as oportunidades de trabalhos futuros relacionados ao uso de algoritmos de consenso em soluções de blockchain baseadas em DAG para a Internet das Coisas.

6.1 CONCLUSÕES

A Internet das Coisas está se tornando cada vez mais presente na sociedade e na vida das pessoas. No entanto, existem desafios em manter o desempenho durante o registro de dados dos dispositivos da IoT. O registro distribuído oferecido pela tecnologia Blockchain tem se apresentado como uma alternativa viável neste contexto.

Porém os modelos tradicionais de blockchain, com blocos lineares e algoritmos de consenso clássicos, na maioria das vezes, não são adequados aos requisitos dos domínios da IoT, que exigem respostas rápidas e consumo eficiente de recursos computacionais. Para superar estes desafios, muitas soluções na literatura têm feito uso de estruturas baseadas em *Directed Acyclic Graph* (DAG) e combinações de mais de um algoritmo de consenso na mesma blockchain. Além disso, existe a necessidade de escolher de forma eficiente qual combinação de algoritmos de consenso é mais viável para cada domínio da IoT.

Diante disso, a hipótese desta tese é de que o uso de uma blockchain baseada em DAG, que combine múltiplos algoritmos de consenso, melhora o desempenho do registro de dados, através da redução do tempo de criação de novos blocos e do consumo de recursos computacionais em ambientes IoT.

Desta forma, esta tese apresenta o projeto e o desenvolvimento da blockchain *OmniBlock*, que teve como principal objetivo melhorar o desempenho do registro de dados da Internet das Coisas. Para isso, foi utilizada uma estrutura de blockchain baseada em DAG, com suporte a múltiplos algoritmos de consenso.

Uma estratégia de escolha de combinações de algoritmos de consenso foi elaborada através da utilização dos algoritmos genéticos *Simple Genetic Algorithm* (SGA) e o *Adaptive Genetic Algorithm* (AGA). Cada um desses algoritmos recebe como entrada uma matriz, contendo as principais características de cada algoritmo de consenso, e fornece como resultado sugestões

de combinações entre esses algoritmos, para cada um dos principais domínios da Internet das Coisas.

Por fim, foram realizadas avaliações comparando o desempenho da *OmniBlock* com uma blockchain linear, que usa algoritmos de consenso tradicionais. Nas avaliações realizadas, foram consideradas as métricas de tempo médio para criação de novos blocos, consumo médio de memória e CPU e tráfego de rede.

Os resultados mostraram que a blockchain *OmniBlock* apresentou melhores resultados para as métricas consideradas do que a blockchain linear; foi observada uma redução no tempo médio de criação de blocos e no consumo de recursos computacionais. Por fim, foi realizada ainda uma comparação entre os dois algoritmos genéticos utilizados, onde os resultados mostraram que o AGA apresentou melhores sugestões de combinações de algoritmos de consenso do que o SGA.

6.2 CONTRIBUIÇÕES

A principal contribuição apresentada por esta tese consiste no desenvolvimento da Blockchain *OmniBlock*, elaborada com base em uma estrutura de DAG e com suporte a combinações de algoritmos de consenso. Outras contribuições apresentadas por esta tese, e que formam um conjunto conjunto coeso com a contribuição principal, foram:

- *Combinações de algoritmos de consenso:* dezesseis combinações de algoritmos de consenso foram consideradas, cada uma com suas vantagens e desvantagens, que poderão ser utilizadas em diferentes cenários da IoT;
- *Estratégia de escolha das combinações de algoritmos de consenso:* a elaboração da estratégia de escolha resultou em sugestões de combinações de algoritmos de consenso que, de acordo com a avaliação de desempenho, se mostraram eficientes para o uso em diferentes domínios da IoT;
- *Implementação dos algoritmos genéticos:* esta tese adaptou dois algoritmos genéticos (SGA e AGA) especificamente para a escolha de algoritmos de consenso em uma blockchain para IoT. Esses algoritmos poderão ser aplicados também para outros cenários de uso da tecnologia blockchain, onde seja necessário escolher algoritmos de consenso; e

- *Comparativo dos algoritmos genéticos:* um comparativo entre os dois algoritmos genéticos utilizados permite indicar qual deles consegue apresentar resultados mais eficientes. Também são indicadas quais valores de taxas de mutação e cruzamento são mais adequadas para uso no AGA.

Todas essas contribuições são parte fundamental da Blockchain *OmniBlock*, e colaboram com o seu objetivo principal, que é o de melhorar o desempenho do processo de registro de dados da IoT.

6.3 LIMITAÇÕES

Ao longo do desenvolvimento desta pesquisa, houveram algumas dificuldades e limitações, que são descritas a seguir:

- *Escopo experimental restrito:*os experimentos realizados desconsideraram a ocorrência de erros ou falhas na inclusão de blocos, concentrando-se apenas nos casos de sucesso. Além disso, os experimentos foram feitos exclusivamente em um *cluster*, sem experimentos em cenários reais de IoT, onde outras variáveis poderiam afetar os resultados;
- *Generalização dos resultados:* o *cluster* utilizado para as avaliações suportou uma quantidade máxima de 1000 nós. Com isso, não foi possível avaliar os resultados para topologias maiores de IoT; e
- *Limitações de parâmetros iniciais:* tanto o SGA quanto o AGA dependem fortemente da população inicial (algoritmos de consenso). Populações pouco diversificadas podem levar a explorações insuficientes do espaço de busca.

Essas limitações impediram a realização de conjuntos de testes mais elaborados, que permitissem avaliar melhor o desempenho da Blockchain *OmniBlock*.

6.4 PUBLICAÇÕES

As contribuições e resultados obtidos ao longo desta pesquisa encontram-se publicados em:

- Survey on Integration of Consensus Mechanisms in IoT-based Blockchains. **JUCS - Journal of Universal Computer Science**, Journal of Universal Computer Science, v. 29, n. 10, p. 1139–1160, 2023 (MORAIS; LINS; ROSA, 2023b).

- Integration and Evaluation of Blockchain Consensus Algorithms for IoT Environments. In: BAROLLI, L. (Ed.). **Advanced Information Networking and Applications**. Cham: Springer International Publishing, 2023. p. 1–13. ISBN 978-3-031-28451-9 (MORAIS; LINS; ROSA, 2023a).

- Selecting Blockchain Consensus Algorithms Integrations for IoT-based Enviroments. In: **Proceedings of the 13th Latin-American Symposium on Dependable and Secure Computing**. [S.l.: s.n.], 2024. p. 116–125. (MORAIS; LINS; ROSA, 2024).

6.5 TRABALHOS FUTUROS

Existem ainda muitos pontos a serem explorados em relação à temática desta tese. A seguir, são descritas algumas possibilidades de pesquisas futuras:

- *Avaliar falhas e ataques:* é possível testar a *OmniBlock* em cenários com nós maliciosos, perdas de mensagens e ataques de negação de serviço, para validar a resiliência das combinações de consenso;

- *Avaliar a OmniBlock em ambientes reais de IoT:* é desejável implantar a *OmniBlock* em redes de sensores e atuadores do mundo real (por exemplo, usando NodeMCU e Raspberry Pi), para avaliar o impacto de variáveis como instabilidade de conexão e mobilidade de dispositivos;

- *Avaliar outras métricas:* é possível considerar outras métricas na avaliação de desempenho, como o consumo de energia em dispositivos de baixa capacidade e taxa de leitura/gravação em disco;

- *Explorar novas meta-heurísticas:* outras técnicas de otimização poderiam ser testadas na *OmniBlock*, como Algoritmos de Colônia de Formigas (CAPRIO *et al.*, 2022) ou Otimização por Enxame de Partículas (SHAMI *et al.*, 2022), para selecionar combinações de consenso e comparar os resultados com o SGA e o AGA;

- *Realizar modelagem formal e verificação:* seria possível também utilizar técnicas de provas formais (como CSP ou redes de Petri) para verificar propriedades de *deadlock* e *livelock* das combinações entre algoritmos de consenso; e

- *Estudar custos e incentivos econômicos:* é desejável analisar modelos de incentivo econômico (*tokens*) para cada combinação de consenso, avaliando a viabilidade financeira em redes IoT públicas. Também poderia ser proposto um mecanismos de penalização para nós que descumpram as regras do consenso, reforçando a segurança contra comportamentos maliciosos.

- *Integrar IA generativa na seleção de consensos:* explorar o uso de Modelos de Linguagem de Grande Escala (LLMs) para auxiliar na definição das matrizes de requisitos de domínios IoT e na escolha de combinações de algoritmos de consenso, comparando o desempenho dessa abordagem com o uso exclusivo de AGs

REFERÊNCIAS

ABBAS, S.; JAVAID, N.; ALMOGREN, A.; GULFAM, S. M.; AHMED, A.; RADWAN, A. Securing genetic algorithm enabled sdn routing for blockchain based internet of things. **IEEE Access**, IEEE, v. 9, p. 139739–139754, 2021.

ABIDALLAH, N.; AILANE, A.; BOUREKKACHE, S.; KAHLOUL, L.; KHALGUI, M.; AZZAOUI, N. Blockchain and genetic algorithms interactions: A brief review. In: IEEE. **2025 International Symposium on iNnovative Informatics of Biskra (ISNIB)**. [S.l.], 2025. p. 1–6.

ACHARYA, B.; GARIKAPATI, K.; YARLAGADDA, A.; DASH, S. Internet of things (iot) and data analytics in smart agriculture: benefits and challenges. In: **AI, Edge and IoT-based Smart Agriculture**. [S.l.]: Elsevier, 2022. p. 3–16.

AHMETOGLU, S.; COB, Z. C.; ALI, N. A systematic review of internet of things adoption in organizations: Taxonomy, benefits, challenges and critical factors. **applied sciences**, MDPI, v. 12, n. 9, p. 4117, 2022.

ALI, M. S.; VECCHIO, M.; PINCHEIRA, M.; DOLUI, K.; ANTONELLI, F.; REHMANI, M. H. Applications of blockchains in the internet of things: A comprehensive survey. **IEEE Communications Surveys & Tutorials**, IEEE, v. 21, n. 2, p. 1676–1717, 2018.

ALI, O.; ISHAK, M. K.; BHATTI, M. K. L.; KHAN, I.; KIM, K.-I. A comprehensive review of internet of things: Technology stack, middlewares, and fog/edge computing interface. **Sensors**, MDPI, v. 22, n. 3, p. 995, 2022.

ALOFI, A.; BOKHARI, M. A.; BAHSOON, R.; HENDLEY, R. Optimizing the energy consumption of blockchain-based systems using evolutionary algorithms: A new problem formulation. **IEEE Transactions on Sustainable Computing**, IEEE, v. 7, n. 4, p. 910–922, 2022.

ALRUBEI, S.; BALL, E.; RIGELSFORD, J. Securing iot-blockchain applications through honesty-based distributed proof of authority consensus algorithm. In: IEEE. **2021 International Conference on Cyber Situational Awareness, Data Analytics and Assessment (CyberSA)**. [S.l.], 2021. p. 1–7.

ALRUBEI, S.; BALL, E.; RIGELSFORD, J. Hdpoa: Honesty-based distributed proof of authority via scalable work consensus protocol for iot-blockchain applications. **Computer Networks**, Elsevier, v. 217, p. 109337, 2022.

ÁLVAREZ, I. A.; GRAMLICH, V.; SEDLMEIR, J. Unsealing the secrets of blockchain consensus: A systematic comparison of the formal security of proof-of-work and proof-of-stake. In: **Proceedings of the 39th ACM/SIGAPP Symposium on Applied Computing**. [S.l.: s.n.], 2024. p. 278–287.

ANDOLA, N.; VENKATESAN, S.; VERMA, S. *et al.* Poewal: A lightweight consensus mechanism for blockchain in iot. **Pervasive and Mobile Computing**, Elsevier, v. 69, p. 101291, 2020.

ANDROULAKI, E.; BARGER, A.; BORTNIKOV, V.; CACHIN, C.; CHRISTIDIS, K.; CARO, A. D.; ENYEART, D.; FERRIS, C.; LAVENTMAN, G.; MANEVICH, Y. *et al.* Hyperledger fabric: a distributed operating system for permissioned blockchains. In: **Proceedings of the thirteenth EuroSys conference**. [S.l.: s.n.], 2018. p. 1–15.

ANGELIS, S. D.; ANIELLO, L.; BALDONI, R.; LOMBARDI, F.; MARGHERI, A.; SASSONE, V. *et al.* Pbft vs proof-of-authority: Applying the cap theorem to permissioned blockchain. In: CEUR-WS. **CEUR workshop proceedings**. [S.l.], 2018. v. 2058.

ARVIND, S.; NARAYANAN, V. A. An overview of security in coap: attack and analysis. In: IEEE. **2019 5th international conference on advanced computing & communication systems (ICACCS)**. [S.l.], 2019. p. 655–660.

ATZORI, M. Blockchain-based architectures for the internet of things: A survey. **Available at SSRN 2846810**, 2017.

AUHL, Z.; CHILAMKURTI, N.; ALHADAD, R.; HEYNE, W. A comparative study of consensus mechanisms in blockchain for iot networks. **Electronics**, MDPI, v. 11, n. 17, p. 2694, 2022.

BABU, R. M.; SATAMRAJU, K. P.; GANGOTHRI, B. N.; MALARKODI, B.; SURESH, C. V. A hybrid model using genetic algorithm for energy optimization in heterogeneous internet of blockchain things. **Telecommunications and Radio Engineering**, Begel House Inc., v. 83, n. 3, 2024.

BACH, L. M.; MIHALJEVIC, B.; ZAGAR, M. Comparative analysis of blockchain consensus algorithms. In: IEEE. **2018 41st international convention on information and communication technology, electronics and microelectronics (MIPRO)**. [S.l.], 2018. p. 1545–1550.

BAI, H.; XIA, G.; FU, S. A two-layer-consensus based blockchain architecture for iot. In: IEEE. **2019 IEEE 9th International Conference on Electronics Information and Emergency Communication (ICEIEC)**. [S.l.], 2019. p. 1–6.

BAIRD, L. Overview of swirlds hashgraph. **White Paper, May**, 2016.

BENHAMAID, S.; BOUABDALLAH, A.; LAKHLEF, H. Recent advances in energy management for green-iot: An up-to-date and comprehensive survey. **Journal of Network and Computer Applications**, Elsevier, v. 198, p. 103257, 2022.

Bitcoin Core Developers. **Block Chain – Bitcoin Developer Reference**. 2025. <https://developer.bitcoin.org/reference/block_chain.html>. Acesso em 02 jul. 2025; especifica o uso do campo ‘time‘ no bloco Bitcoin.

BOETTIGER, C. An introduction to docker for reproducible research. **ACM SIGOPS Operating Systems Review**, ACM New York, NY, USA, v. 49, n. 1, p. 71–79, 2015.

BREWER, E. A. Towards robust distributed systems. In: PORTLAND, OR. **PODC**. [S.l.], 2000. v. 7, n. 10.1145, p. 343–477.

BUTERIN, V. *et al.* A next-generation smart contract and decentralized application platform. **white paper**, v. 3, n. 37, p. 2–1, 2014.

CAMPELO, F.; TAKAHASHI, F. Sample size estimation for power and accuracy in the experimental comparison of algorithms. **Journal of Heuristics**, Springer, v. 25, n. 2, p. 305–338, 2019.

CAPRIO, D. D.; EBRAHIMNEJAD, A.; ALREZAAMIRI, H.; SANTOS-ARTEAGA, F. J. A novel ant colony algorithm for solving shortest path problems with fuzzy arc weights. **Alexandria Engineering Journal**, Elsevier, v. 61, n. 5, p. 3403–3415, 2022.

CASTRO, M.; LISKOV, B. *et al.* Practical byzantine fault tolerance. In: **OsDI**. [S.l.: s.n.], 1999. v. 99, n. 1999, p. 173–186.

CHAKRABORTY, A.; ISLAM, M.; SHAHRIYAR, F.; ISLAM, S.; ZAMAN, H. U.; HASAN, M. Smart home system: a comprehensive review. **Journal of Electrical and Computer Engineering**, Wiley Online Library, v. 2023, n. 1, p. 7616683, 2023.

CHEN, Y.; GUO, Y.; WANG, Y.; BIE, R. Toward prevention of parasite chain attack in iota blockchain networks by using evolutionary game model. **Mathematics**, MDPI, v. 10, n. 7, p. 1108, 2022.

CHEN, Y.; ZHANG, Y.; ZHUANG, Y.; MIAO, K.; POURIYEH, S.; HAN, M. Efficient and secure blockchain consensus algorithm for heterogeneous industrial internet of things nodes based on double-dag. **IEEE Transactions on Industrial Informatics**, IEEE, v. 20, n. 4, p. 6300–6312, 2024.

CNA BRASIL. **Sumário Executivo do PIB do Agronegócio – 4º trimestre de 2024**. [S.l.], 2025. Setor representou 23,2% do PIB nacional em 2024. Disponível em: <https://cnabrasil.org.br/publicacoes/sumario-executivo-pib-do-agronegocio-4o-trimestre-de-2024/>.

CULLEN, A.; FERRARO, P.; KING, C.; SHORTEN, R. On the resilience of dag-based distributed ledgers in iot applications. **IEEE Internet of Things Journal**, IEEE, v. 7, n. 8, p. 7112–7122, 2020.

DIGITALE, J. C.; MARTIN, J. N.; GLYMOUR, M. M. Tutorial on directed acyclic graphs. **Journal of Clinical Epidemiology**, Elsevier, v. 142, p. 264–267, 2022.

DZIEMBOWSKI, S.; FAUST, S.; KOLMOGOROV, V.; PIETRZAK, K. Proofs of space. In: SPRINGER. **Annual Cryptology Conference**. [S.l.], 2015. p. 585–605.

Ethereum Foundation. **Blocks – Ethereum Developer Documentation**. 2025. <https://ethereum.org/en/developers/docs/blocks/>. Acesso em 02 jul. 2025; descreve que o tempo de bloco na Ethereum é de 12 segundos.

FAN, C.; GHAEMI, S.; KHAZAEI, H.; CHEN, Y.; MUSILEK, P. Performance analysis of the iota dag-based distributed ledger. **ACM Transactions on Modeling and Performance Evaluation of Computing Systems**, ACM New York, NY, v. 6, n. 3, p. 1–20, 2021.

FEI, T.; KUN, D.; ZHANGTAO, Y.; GUOWEI, L. Authentication scheme for industrial internet of things based on dag blockchain. **Journal of China University of Posts and Telecommunications**, v. 28, n. 6, p. 1, 2021.

FERDOUS, M. S.; CHOWDHURY, M. J. M.; HOQUE, M. A. A survey of consensus algorithms in public blockchain systems for crypto-currencies. **Journal of Network and Computer Applications**, Elsevier, v. 182, p. 103035, 2021.

FU, J.; ZHANG, L.; WANG, L.; LI, F. Bct: An efficient and fault tolerance blockchain consensus transform mechanism for iot. **IEEE Internet of Things Journal**, IEEE, v. 10, n. 14, p. 12055–12065, 2021.

FU, X.; WANG, H.; SHI, P.; ZHANG, X. Teegraph: A blockchain consensus algorithm based on tee and dag for data sharing in iot. **Journal of Systems Architecture**, Elsevier, v. 122, p. 102344, 2022.

GIL, A. C. **Métodos e técnicas de pesquisa social**. [S.l.]: 6. ed. Ediitora Atlas SA, 2008.

GOLDBERG, D. E. The genetic algorithm approach: why, how, and what next? In: **Adaptive and learning systems: Theory and applications**. [S.l.]: Springer, 1986. p. 247–253.

GURUPRAKASH, J.; KOPPU, S. Ec-elgamal and genetic algorithm-based enhancement for lightweight scalable blockchain in iot domain. **IEEE Access**, IEEE, v. 8, p. 141269–141281, 2020.

HASSIJA, V.; CHAMOLA, V.; SAXENA, V.; JAIN, D.; GOYAL, P.; SIKDAR, B. A survey on iot security: application areas, security threats, and solution architectures. **IEEE Access**, IEEE, v. 7, p. 82721–82743, 2019.

HELLANI, H.; SLIMAN, L.; SAMHAT, A. E.; EXPOSITO, E. Tangle the blockchain: towards connecting blockchain and dag. In: IEEE. **2021 IEEE 30th International Conference on Enabling Technologies: Infrastructure for Collaborative Enterprises (WETICE)**. [S.l.], 2021. p. 63–68.

HOLLAND, J. H. Adaptation in natural and artificial systems. **The University of Michigan Press**, 1975.

HOWARD, H.; MORTIER, R. Paxos vs raft: Have we reached consensus on distributed consensus? In: **Proceedings of the 7th Workshop on Principles and Practice of Consistency for Distributed Data**. [S.l.: s.n.], 2020. p. 1–9.

HUANG, J.; KONG, L.; CHEN, G.; WU, M.-Y.; LIU, X.; ZENG, P. Towards secure industrial iot: Blockchain system with credit-based consensus mechanism. **IEEE Transactions on Industrial Informatics**, IEEE, v. 15, n. 6, p. 3680–3689, 2019.

HUANG, R.; YANG, X.; AJAY, P. Consensus mechanism for software-defined blockchain in internet of things. **Internet of Things and Cyber-Physical Systems**, Elsevier, v. 3, p. 52–60, 2023.

HUSSEIN, Z.; SALAMA, M. A.; EL-RAHMAN, S. A. Evolution of blockchain consensus algorithms: a review on the latest milestones of blockchain consensus algorithms. **Cybersecurity**, Springer, v. 6, n. 1, p. 30, 2023.

Hyperledger Foundation. **Hyperledger Fabric: Performance Considerations**. 2025. <https://hyperledger-fabric.readthedocs.io/en/latest/performance.html>. Acesso em 02 jul. 2025. "a threshold of 70–80for maximum workload".

ILAKKIYA, N.; RAJARAM, A. A secured trusted routing using the structure of a novel directed acyclic graph-blockchain in mobile ad hoc network internet of things environment. **Multimedia Tools and Applications**, Springer, p. 1–26, 2024.

IoT Security Foundation. **IoT Security Assurance Framework, Release 3.0**. 2021. Online; Version 3.0. Acesso em: 25 jun. 2025. Disponível em: <https://www.iotsecurityfoundation.org/best-practice-guidelines/>.

ISLAM, M. J.; ISLAM, S.; HOSSAIN, M.; NOOR, S.; ISLAM, S. R. Securing blockchain systems: A layer-oriented survey of threats, vulnerability taxonomy, and detection methods. **Future Internet**, MDPI, v. 17, n. 5, p. 205, 2025.

JABŁCZYŃSKA, M.; KOSC, K.; RYŚ, P.; SAKOWSKI, P.; ŚLEPACZUK, R.; ZAKRZEWSKI, G. Energy and cost efficiency of bitcoin mining endeavor. **PloS one**, Public Library of Science San Francisco, CA USA, v. 18, n. 3, p. e0283687, 2023.

JAIN, R. **The art of computer systems performance analysis: techniques for experimental design, measurement, simulation, and modeling**. [S.l.]: Wiley New York, 1991. v. 1.

JAKOBOVIĆ, D.; GOLUB, M. Adaptive genetic algorithm. **Journal of computing and information technology**, Sveučilište u Zagrebu Sveučilišni računski centar, v. 7, n. 3, p. 229–235, 1999.

JAVED, M. U.; REHMAN, M.; JAVAID, N.; ALDEGHEISHEM, A.; ALRAJEH, N.; TAHIR, M. Blockchain-based secure data storage for distributed vehicular networks. **Applied Sciences**, MDPI, v. 10, n. 6, p. 2011, 2020.

KASHANI, M. H.; MADANIPOUR, M.; NIKRAVAN, M.; ASGHARI, P.; MAHDIPOUR, E. A systematic review of iot in healthcare: Applications, techniques, and trends. **Journal of Network and Computer Applications**, Elsevier, v. 192, p. 103164, 2021.

KAUR, S.; CHATURVEDI, S.; SHARMA, A.; KAR, J. A research survey on applications of consensus protocols in blockchain. **Security and Communication Networks**, Wiley Online Library, v. 2021, n. 1, p. 6693731, 2021.

KHAN, A. A.; LAGHARI, A. A.; RASHID, M.; LI, H.; JAVED, A. R.; GADEKALLU, T. R. Artificial intelligence and blockchain technology for secure smart grid and power distribution automation: A state-of-the-art review. **Sustainable Energy Technologies and Assessments**, Elsevier, v. 57, p. 103282, 2023.

KHAN, M.; HARTOG, F. den; HU, J. A survey and ontology of blockchain consensus algorithms for resource-constrained iot systems. **Sensors**, MDPI, v. 22, n. 21, p. 8188, 2022.

KIAYIAS, A.; RUSSELL, A.; DAVID, B.; OLIYNYKOV, R. Ouroboros: A provably secure proof-of-stake blockchain protocol. In: SPRINGER. **Annual international cryptology conference**. [S.l.], 2017. p. 357–388.

KIM, H.; KIM, W.; KANG, Y.; KIM, H.; SEO, H. Post-quantum delegated proof of luck for blockchain consensus algorithm. **Applied Sciences**, MDPI, v. 14, n. 18, p. 8394, 2024.

KIZZA, J. M. Internet of things (iot): growth, challenges, and security. In: **Guide to Computer Network Security**. [S.l.]: Springer, 2024. p. 557–573.

KRANER, B.; VALLARANO, N.; SCHWARZ-SCHILLING, C.; TESSONE, C. J. Agent-based modelling of ethereum consensus. In: IEEE. **2023 IEEE International Conference on Blockchain and Cryptocurrency (ICBC)**. [S.l.], 2023. p. 1–8.

KUMAR, A.; SAH, B. K.; MEHROTRA, T.; RAJPUT, G. K. A review on double spending problem in blockchain. In: IEEE. **2023 International Conference on Computational Intelligence and Sustainable Engineering Solutions (CISES)**. [S.l.], 2023. p. 881–889.

KWON, J. Tendermint: Consensus without mining. **Draft v. 0.6, fall**, v. 1, n. 11, p. 1–11, 2014.

LAO, L.; LI, Z.; HOU, S.; XIAO, B.; GUO, S.; YANG, Y. A survey of iot applications in blockchain systems: Architecture, consensus, and traffic modeling. **ACM Computing Surveys (CSUR)**, ACM New York, NY, USA, v. 53, n. 1, p. 1–32, 2020.

LARIMER, D. Delegated proof-of-stake (dpos). **Bitshare whitepaper**, v. 81, p. 85, 2014.

LASLA, N.; AL-SAHAN, L.; ABDALLAH, M.; YOUNIS, M. Green-pow: An energy-efficient blockchain proof-of-work consensus algorithm. **Computer Networks**, Elsevier, v. 214, p. 109118, 2022.

LEE, Y.; RATHORE, S.; PARK, J. H.; PARK, J. H. A blockchain-based smart home gateway architecture for preventing data forgery. **Human-centric Computing and Information Sciences**, Springer, v. 10, n. 1, p. 9, 2020.

LEI, M.; XU, L.; LIU, T.; LIU, S.; SUN, C. Integration of privacy protection and blockchain-based food safety traceability: Potential and challenges. **Foods**, MDPI, v. 11, n. 15, p. 2262, 2022.

LI, L.; HUANG, D.; ZHANG, C. An efficient dag blockchain architecture for iot. **IEEE Internet of Things Journal**, IEEE, v. 10, n. 2, p. 1286–1296, 2022.

LIAO, G.; DING, H.; ZHONG, C.; LEI, Y. Rt-dag: A dag-based blockchain supporting real-time transactions. **IEEE Internet of Things Journal**, IEEE, 2024.

LIAO, Z.; CHENG, S. Rvc: A reputation and voting based blockchain consensus mechanism for edge computing-enabled iot systems. **Journal of Network and Computer Applications**, Elsevier, v. 209, p. 103510, 2023.

LOBO, F.; LIMA, C. F.; MICHALEWICZ, Z. **Parameter setting in evolutionary algorithms**. [S.l.]: Springer Science & Business Media, 2007. v. 54.

LONE, A. H.; NAAZ, R. Applicability of blockchain smart contracts in securing internet and iot: A systematic literature review. **Computer Science Review**, Elsevier, v. 39, p. 100360, 2021.

LUNARDI, R. C.; MICHELIN, R. A.; NEU, C. V.; NUNES, H. C.; ZORZO, A. F.; KANHERE, S. S. Impact of consensus on appendable-block blockchain for iot. In: **Proceedings of the 16th EAI International Conference on Mobile and Ubiquitous Systems: Computing, Networking and Services**. [S.l.: s.n.], 2019. p. 228–237.

LUO, H.; YU, H.; LUO, J. Praft and rpbft: A class of blockchain consensus algorithm and their applications in electric vehicles charging scenarios for v2g networks. **Internet of Things and Cyber-Physical Systems**, Elsevier, v. 3, p. 61–70, 2023.

MATHUR, A.; BARATI, M.; AUJLA, G. S.; RANA, O. Towards scalable and secure blockchain in internet of things: A preference-driven committee member auction consensus approach. **Distributed Ledger Technologies: Research and Practice**, ACM New York, NY, 2024.

MATHUR, S.; KALLA, A.; GÜR, G.; BOHRA, M. K.; LIYANAGE, M. A survey on role of blockchain for iot: Applications and technical aspects. **Computer Networks**, Elsevier, v. 227, p. 109726, 2023.

MAZIERES, D. The stellar consensus protocol: A federated model for internet-level consensus. **Stellar Development Foundation**, v. 32, p. 1–45, 2015.

MENDEZ, M.; PAPAPANAGIOTOU, I.; YANG, B. **Internet of things: survey on security, information security journal. Glob Perspect 27 (3): 162–182**. 2018.

MENON, A. A.; SARANYA, T.; SURESHBABU, S.; MAHESH, A. A comparatıve analysis on three consensus algorithms: proof of burn, proof of elapsed time, proof of authority. In: **Computer Networks and Inventive Communication Technologies: Proceedings of Fourth ICCNCT 2021**. [S.l.]: Springer, 2021. p. 369–383.

MILLAR, J.; HADDADI, H.; MADHAVAPEDDY, A. Energy-aware deep learning on resource-constrained hardware. **arXiv preprint arXiv:2505.12523**, 2025.

MILUTINOVIC, M.; HE, W.; WU, H.; KANWAL, M. Proof of luck: An efficient blockchain consensus protocol. In: **proceedings of the 1st Workshop on System Software for Trusted Execution**. [S.l.: s.n.], 2016. p. 1–6.

MISHRA, B.; KERTESZ, A. The use of mqtt in m2m and iot systems: A survey. **Ieee Access**, Ieee, v. 8, p. 201071–201086, 2020.

MOHANTA, B. K.; DEHURY, M. K.; SOLLETI, P. K.; PANDA, S. S. Decentralized consensus protocol for securing iot networks using blockchain technology. In: IEEE. **2024 8th International Conference on I-SMAC (IoT in Social, Mobile, Analytics and Cloud)(I-SMAC)**. [S.l.], 2024. p. 81–87.

MONRAT, A. A.; SCHELÉN, O.; ANDERSSON, K. A survey of blockchain from the perspectives of applications, challenges, and opportunities. **IEEE Access**, IEEE, v. 7, p. 117134–117151, 2019.

MORAIS, A. M. de; LINS, F. A. A.; ROSA, N. S. Integration and evaluation of blockchain consensus algorithms for iot environments. In: BAROLLI, L. (Ed.). **Advanced Information Networking and Applications**. Cham: Springer International Publishing, 2023. p. 1–13. ISBN 978-3-031-28451-9.

MORAIS, A. M. de; LINS, F. A. A.; ROSA, N. S. Survey on integration of consensus mechanisms in iot-based blockchains. **JUCS - Journal of Universal Computer Science**, Journal of Universal Computer Science, v. 29, n. 10, p. 1139–1160, 2023. ISSN 0948-695X. Disponível em: <https://doi.org/10.3897/jucs.94929>.

MORAIS, A. M. de; LINS, F. A. A.; ROSA, N. S. Selecting blockchain consensus algorithms integrations for iot-based enviroments. In: **Proceedings of the 13th Latin-American Symposium on Dependable and Secure Computing**. [S.l.: s.n.], 2024. p. 116–125.

MORRIS, C.; CONWAY, A. A.; BECRAFT, J. L.; FERRUCCI, B. J. Toward an understanding of data collection integrity. **Behavior analysis in practice**, Springer, v. 15, n. 4, p. 1361–1372, 2022.

NAIK, N. Choice of effective messaging protocols for iot systems: Mqtt, coap, amqp and http. In: IEEE. **2017 IEEE international systems engineering symposium (ISSE)**. [S.l.], 2017. p. 1–7.

NAKAMOTO, S. A peer-to-peer electronic cash system. **Bitcoin.–URL: https://bitcoin.org/bitcoin. pdf**, v. 4, 2008.

NGUYEN, T. S. L.; JOURJON, G.; POTOP-BUTUCARU, M.; THAI, K. L. Impact of network delays on hyperledger fabric. In: IEEE. **IEEE INFOCOM 2019-IEEE conference on computer communications workshops (INFOCOM WKSHPS)**. [S.l.], 2019. p. 222–227.

ONGARO, D.; OUSTERHOUT, J. The raft consensus algorithm. **Lecture Notes CS**, v. 190, p. 2022, 2015.

P4TITAN. A peer-to-peer crypto-currency with proof-of-burn "mining without powerful hardware". 2014. Acesso em 22/03/2024. Disponível em: <https://slimcoin.info/whitepaperSLM.pdf>.

PAUL, R.; GHOSH, N.; SAU, S.; CHAKRABARTI, A.; MOHAPATRA, P. Blockchain based secure smart city architecture using low resource iots. **Computer Networks**, Elsevier, v. 196, p. 108234, 2021.

PERWEJ, Y.; HAQ, K.; PARWEJ, F.; MUMDOUH, M.; HASSAN, M. The internet of things (iot) and its application domains. **International Journal of Computer Applications**, v. 975, n. 8887, p. 182, 2019.

PRODANOV, C. C.; FREITAS, E. C. D. **Metodologia do trabalho científico: métodos e técnicas da pesquisa e do trabalho acadêmico-2ª Edição**. [S.l.]: Editora Feevale, 2013.

QI, J.; GUAN, Y. Practical byzantine fault tolerance consensus based on comprehensive reputation. **Peer-to-Peer Networking and Applications**, Springer, v. 16, n. 1, p. 420–430, 2023.

QU, X.; WANG, S.; LI, K.; HUANG, J.; CHENG, X. Tidyblock: A novel consensus mechanism for dag-based blockchain in iot. **IEEE Transactions on Mobile Computing**, IEEE, 2024.

RASOLROVEICY, M.; FOKAEFS, M. Dynamic reconfiguration of consensus protocol for iot data registry on blockchain. In: **Proceedings of the 30th Annual International Conference on Computer Science and Software Engineering**. [S.l.: s.n.], 2020. p. 227–236.

REN, Y.; CHEN, Z. A parallel directed acyclic graph blockchain ledger and consensus algorithm for industrial internet of things. In: IEEE. **2023 3rd International Conference on Electronic Information Engineering and Computer Science (EIECS)**. [S.l.], 2023. p. 1049–1053.

RESHI, I. A.; SHOLLA, S. Ibf network: enhancing network privacy with iot, blockchain, and fog computing on different consensus mechanisms. **Cluster Computing**, Springer, v. 28, n. 3, p. 208, 2025.

REYNA, A.; MARTÍN, C.; CHEN, J.; SOLER, E.; DÍAZ, M. On blockchain and its integration with iot. challenges and opportunities. **Future generation computer systems**, Elsevier, v. 88, p. 173–190, 2018.

SAAD, S. M. S.; RADZI, R. Z. R. M.; OTHMAN, S. H. Comparative analysis of the blockchain consensus algorithm between proof of stake and delegated proof of stake. In: IEEE. **2021 International Conference on Data Science and Its Applications (ICoDSA)**. [S.l.], 2021. p. 175–180.

SADAWI, A. A.; HASSAN, M. S.; NDIAYE, M. A review on the integration of blockchain and iot. In: IEEE. **2020 International conference on communications, signal processing, and their applications (ICCSPA)**. [S.l.], 2021. p. 1–6.

SAGIRLAR, G.; CARMINATI, B.; FERRARI, E.; SHEEHAN, J. D.; RAGNOLI, E. Hybrid-iot: Hybrid blockchain architecture for internet of things-pow sub-blockchains. In: IEEE. **2018 IEEE International Conference on Internet of Things (iThings) and IEEE Green Computing and Communications (GreenCom) and IEEE Cyber, Physical and Social Computing (CPSCom) and IEEE Smart Data (SmartData)**. [S.l.], 2018. p. 1007–1016.

SALIMITARI, M.; CHATTERJEE, M.; FALLAH, Y. P. A survey on consensus methods in blockchain for resource-constrained iot networks. **Internet of Things**, Elsevier, v. 11, p. 100212, 2020.

SHAMI, T. M.; EL-SALEH, A. A.; ALSWAITTI, M.; AL-TASHI, Q.; SUMMAKIEH, M. A.; MIRJALILI, S. Particle swarm optimization: A comprehensive survey. **Ieee Access**, IEEE, v. 10, p. 10031–10061, 2022.

SILVANO, W. F.; MARCELINO, R. Iota tangle: A cryptocurrency to communicate internet-of-things data. **Future generation computer systems**, Elsevier, v. 112, p. 307–319, 2020.

SON, B.; LEE, J.; JANG, H. A scalable iot protocol via an efficient dag-based distributed ledger consensus. **Sustainability**, MDPI, v. 12, n. 4, p. 1529, 2020.

SOORI, M.; AREZOO, B.; DASTRES, R. Internet of things for smart factories in industry 4.0, a review. **Internet of Things and Cyber-Physical Systems**, Elsevier, v. 3, p. 192–204, 2023.

SOPPELSA, F.; KAEWKASI, C. **Native docker clustering with swarm**. [S.l.]: Packt Publishing Ltd, 2016.

SUTIKNO, S.; SURYA, A.; EFFENDI, R. An implementation of elgamal elliptic curves cryptosystems. In: IEEE. **IEEE. APCCAS 1998. 1998 IEEE Asia-Pacific Conference on Circuits and Systems. Microelectronics and Integrating Systems. Proceedings (Cat. No. 98EX242)**. [S.l.], 1998. p. 483–486.

TAPWAL, R.; DEB, P. K.; MISRA, S.; PAL, S. K. Amaurotic-entity-based consensus selection in blockchain-enabled industrial iot. **IEEE Internet of Things Journal**, IEEE, v. 9, n. 14, p. 11648–11655, 2021.

THULASIRAMAN, K.; SWAMY, M. **5.7 acyclic directed graphs**. [S.l.]: John Wiley and Son New York, NY; Chichester; Brisbane; Toronto; Singapore, 1992. v. 118. ISSN 978-0-471-51356-8.

TILKOV, S.; VINOSKI, S. Node. js: Using javascript to build high-performance network programs. **IEEE Internet Computing**, IEEE, v. 14, n. 6, p. 80–83, 2010.

TODD, P. Ripple protocol consensus algorithm review. **May 11th**, 2015.

TSANG, Y. P.; CHOY, K. L.; WU, C. H.; HO, G. T. S.; LAM, H. Y. Blockchain-driven iot for food traceability with an integrated consensus mechanism. **IEEE access**, IEEE, v. 7, p. 129000–129017, 2019.

UGOCHUKWU, N. A.; GOYAL, S.; RAJAWAT, A. S.; ISLAM, S. M.; HE, J.; ASLAM, M. An innovative blockchain-based secured logistics management architecture: utilizing an rsa asymmetric encryption method. **Mathematics**, MDPI, v. 10, n. 24, p. 4670, 2022.

ULLO, S. L.; SINHA, G. R. Advances in smart environment monitoring systems using iot and sensors. **Sensors**, MDPI, v. 20, n. 11, p. 3113, 2020.

UVEISE, S. M.; FATHIMA, S. S. S. Efficient lightweight blockchain with hybridized consensus algorithm for iot networks. **IETE Journal of Research**, Taylor & Francis, v. 70, n. 12, p. 8527–8537, 2024.

VOSE, M. D. **The simple genetic algorithm: foundations and theory**. [S.l.]: MIT press, 1999.

WANG, B.; LI, Z.; LI, H. Hybrid consensus algorithm based on modified proof-of-probability and dpos. **Future Internet**, MDPI, v. 12, n. 8, p. 122, 2020.

WANG, S.; LI, H.; CHEN, J.; WANG, J.; DENG, Y. Dag blockchain-based lightweight authentication and authorization scheme for iot devices. **Journal of information security and applications**, Elsevier, v. 66, p. 103134, 2022.

WANG, T.; WANG, Q.; SHEN, Z.; JIA, Z.; SHAO, Z. Understanding characteristics and system implications of dag-based blockchain in iot environments. **IEEE Internet of Things Journal**, IEEE, v. 9, n. 16, p. 14478–14489, 2021.

WERTH, J.; BERENJESTANAKI, M. H.; BARZEGAR, H. R.; IOINI, N. E.; PAHL, C. A review of blockchain platforms based on the scalability, security and decentralization trilemma. **ICEIS (1)**, p. 146–155, 2023.

WINTER, T.; THUBERT, P.; BRANDT, A.; HUI, J.; KELSEY, R.; LEVIS, P.; PISTER, K.; STRUIK, R.; VASSEUR, J.-P.; ALEXANDER, R. **RPL: IPv6 routing protocol for low-power and lossy networks**. [S.l.], 2012.

WU, W.; SHEN, L.; ZHAO, Z.; LI, M.; HUANG, G. Q. Industrial iot and long short-term memory network-enabled genetic indoor-tracking for factory logistics. **IEEE Transactions on Industrial Informatics**, IEEE, v. 18, n. 11, p. 7537–7548, 2022.

XU, R.; CHEN, Y.; BLASCH, E.; CHEN, G. Microchain: A hybrid consensus mechanism for lightweight distributed ledger for iot. **arXiv preprint arXiv:1909.10948**, 2019.

YANG, C.; WANG, T.; WANG, K. A consensus mechanism based on an improved genetic algorithm. **Open Access Library Journal**, Scientific Research Publishing, v. 7, n. 9, p. 1–6, 2020.

YANG, G.; LEE, K.; LEE, K.; YOO, Y.; LEE, H.; YOO, C. Resource analysis of blockchain consensus algorithms in hyperledger fabric. **IEEE Access**, IEEE, v. 10, p. 74902–74920, 2022.

YANG, R.; WAKEFIELD, R.; LYU, S.; JAYASURIYA, S.; HAN, F.; YI, X.; YANG, X.; AMARASINGHE, G.; CHEN, S. Public and private blockchain in construction business process and information integration. **Automation in construction**, Elsevier, v. 118, p. 103276, 2020.

YANG, W.; DAI, X.; XIAO, J.; JIN, H. Ldv: A lightweight dag-based blockchain for vehicular social networks. **IEEE Transactions on Vehicular Technology**, IEEE, v. 69, n. 6, p. 5749–5759, 2020.

YAZDINEJAD, A.; SRIVASTAVA, G.; PARIZI, R. M.; DEHGHANTANHA, A.; KARIMIPOUR, H.; KARIZNO, S. R. Slpow: Secure and low latency proof of work protocol for blockchain in green iot networks. In: IEEE. **2020 IEEE 91st Vehicular Technology Conference (VTC2020-Spring)**. [S.l.], 2020. p. 1–5.

YOUSUF, R.; JEELANI, Z.; KHAN, D. A.; BHAT, O.; TELI, T. A. Consensus algorithms in blockchain-based cryptocurrencies. In: IEEE. **2021 international conference on advances in electrical, computing, communication and sustainable technologies (ICAECT)**. [S.l.], 2021. p. 1–6.

YUSOF, S.; ZAHILAH, R.; OTHMAN, S. Blockchain consensus for resources constraint devices: A hybrid approach using poa, dpos and threshold cryptography. **Journal of Engineering and Technology (JET)**, v. 15, n. 2, 2024.

ZAABAR, B.; CHEIKHROUHOU, O.; JAMIL, F.; AMMI, M.; ABID, M. Healthblock: A secure blockchain-based healthcare data management system. **Computer Networks**, Elsevier, v. 200, p. 108500, 2021.

ZHANG, Z.; ZHANG, M.; LI, Y.; FAN, B.; JIANG, L. Directed acyclic graph blockchain for secure spectrum sharing and energy trading in power iot. **China Communications**, IEEE, v. 20, n. 5, p. 182–197, 2023.

ZHENG, W.; WANG, X.; XIE, Z.; LI, Y.; YE, X.; WANG, J.; XIONG, X. Data management method for building internet of things based on blockchain sharding and dag. **Internet of Things and Cyber-Physical Systems**, Elsevier, v. 4, p. 217–234, 2024.

ZHIPENG, F. Research on blockchain hybrid consensus algorithm based on internet of things. **Int. J. Internet Things Big Data**, 2019.

ZHUANG, Y.; CHEN, Y.; ZHANG, X.; REN, T.; HAN, M.; ALAM, M.; HONG, Z. A large-scale node lightweight consensus algorithm of blockchain for internet of things. **IEEE Internet of Things Journal**, IEEE, 2024.

ZILNIEKS, V. Choosing a consensus mechanism for a blockchain based p2p instant transaction system integrated with iot. In: IEEE. **2021 International Conference on Communication, Control and Information Sciences (ICCISc)**. [S.l.], 2021. v. 1, p. 1–6.

ZUBAYDI, H. D.; VARGA, P.; MOLNÁR, S. Leveraging blockchain technology for ensuring security and privacy aspects in internet of things: a systematic literature review. **Sensors**, MDPI, v. 23, n. 2, p. 788, 2023.

APÊNDICE A – METODOLOGIA

A metodologia utilizada nesta fase da pesquisa foi a revisão bibliográfica, que permite a identificação de informações em conteúdos já publicados na literatura para posterior análise desses resultados. A execução deste trabalho deu-se da seguinte forma:

1. **Definição dos temas de estudo:** dado o crescimento e popularização da IoT e as inúmeras propostas de utilização de Blockchain, inicialmente foram observadas as possíveis lacunas na área. Com isso percebeu-se que o a combinação de algoritmos de consenso pode ser muito vantajosa para IoT, mas ainda é um tema pouco explorado na literatura.
2. **Definição dos objetivos da pesquisa:** nesta etapa o objetivo foi analisar as propostas de combinação dos algoritmos de consenso atualmente existentes na literatura.
3. **Revisão de literatura sobre combinações entre algoritmos de consenso para IoT:** após definição do objetivo do estudo, foram realizadas buscas nas principais bases científicas por trabalhos relacionados.
4. **Análise dos artigos:** em seguida, os artigos foram analisados para verificar a possibilidade de realizar novas combinações entre os algoritmos de consenso para torná-los adequados para IoT.

Para a realização do estudo, os critérios de pesquisa foram definidos por meio da elaboração de um protocolo. Informações essenciais sobre este protocolo são apresentadas a seguir:

- **Pergunta de pesquisa:** Quais algoritmos de consenso podem ser integrados para usar Blockchain em IoT de forma eficiente?
- **Bases científicas consultadas:** IEEE Xplore; Biblioteca Digital ACM; Springer; ScienceDirect; Google Acadêmico;
- **Palavras-chave:** "Blockchain", "IoT", "Consensus algorithms";
- **Critérios de inclusão:** foram escolhidos artigos publicados nos últimos cinco anos e que apresentem conteúdo relacionado à questão de pesquisa;
- **Critérios de exclusão:** trabalhos publicados há mais de cinco anos; trabalhos escritos em idiomas diferentes do inglês; artigos que apresentem conteúdo não relacionado ao tema da pesquisa;

- ***Strings* de busca:** "Blockchain AND IoT"; "Integration AND Consensus AND IoT"; "Integrating AND Consensus AND IoT"; "Hybrid AND Consensus AND IoT"; "Multiple AND Consensus AND IoT".

A segunda etapa da Revisão Bibliográfica teve como objetivo atualizar o conteúdo estudado até então, incluindo novos trabalhos publicados e também direcionando melhor o interesse dos temas pesquisados.

Foram consultadas as mesmas bases cientificas do inicio, por serem estas as mais relevantes do meio acadêmico. Foram acrescentadas as seguintes *Strings* de busca:

- **IEEE**: ("Document Title":Blockchain) AND (("Document Title":IoT) OR ("Document Title":Internet of Things)) AND ("Document Title":Consensus)
- **Science Direct:** Blockchain AND (IoT OR Internet of Things) AND Consensus
- **Springer:** Blockchain AND (IoT OR Internet of Things) AND Consensus

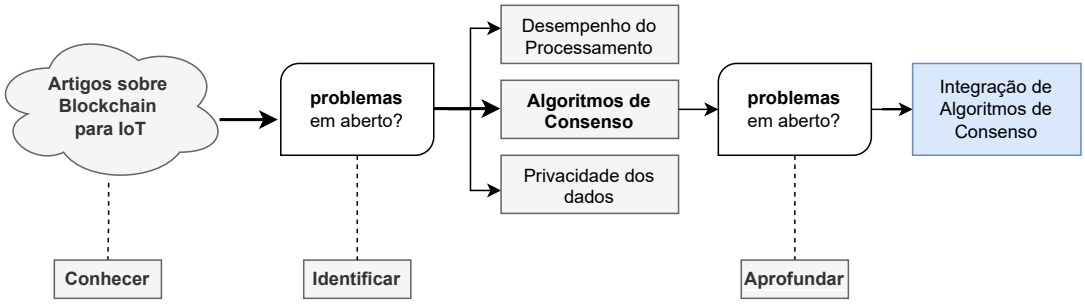
Ainda na segunda e posteriormente na terceira etapa da Revisão Bibliográfica, foram incluídas *Strings* de buscas relacionadas aos demais temas de interesse desta tese: algoritmo genéticos e Blockchain baseadas em DAG:

- **IEEE:**("Document Title":"blockchain") AND (("Document Title":"DAG") OR ("Document Title":"Directed Acyclic Graph")) AND ("Document Title":"consensus") AND ("Document Title":"IoT"OR "Document Title":"Internet of Things")
- **Springer:** blockchain AND (DAG OR "Directed Acyclic Graph") AND ("genetic algorithm"OR "evolutionary algorithm") AND (IoT OR "Internet of Things")
- **ACM:** (Title:blockchain) AND (Title:DAG OR Title:"Directed Acyclic Graph") AND (Title:"genetic algorithm"OR Title:"evolutionary algorithm") AND (Title:IoT OR Title:"Internet of Things")
- **Science Direct:** (blockchain) AND (DAG OR "Directed Acyclic Graph") AND ("genetic algorithm"OR "evolutionary algorithm") AND (IoT OR "Internet of Things")

A.1 PRIMEIRA FASE DA REVISÃO DE LITERATURA

A primeira fase da pesquisa bibliográfica teve como objetivo explorar o uso da tecnologia Blockchain para segurança de dispositivos IoT. A Figura A.1 apresenta a sequencia de passos adotados nesta etapa da revisão bibliográfica.

Figura A.1 – Etapas da primeira fase da revisão bibliográfica de literatura.



Fonte: O autor (2023).

No inicio da pesquisa, as *strings* de busca foram aplicadas às bases científicas listadas na Tabela A.1. Esta etapa, nomeada de 'conhecer' resultou em 197 artigos, com enfoques diversos sobre o uso da tecnologia Blockchain para IoT. A Tabela A.1 mostra o total de artigos que foram coletados em cada base científica.

Tabela A.1 – Quantitativo de trabalhos sobre o uso de blockchain em ambientes IoT.

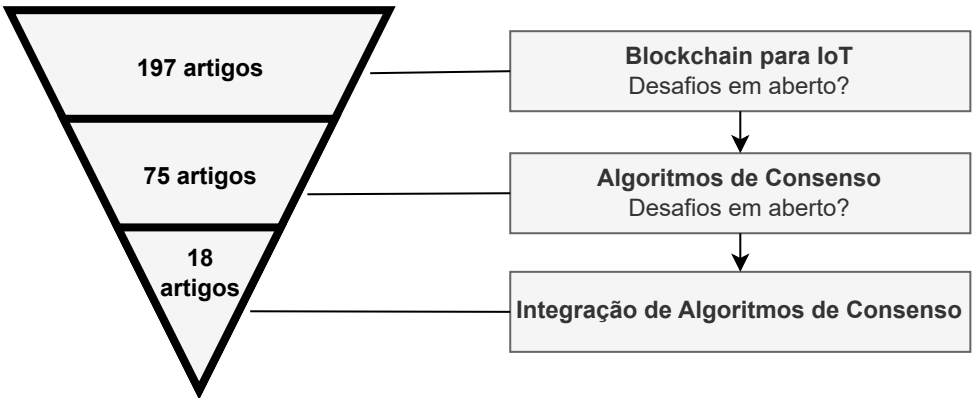
Bases de Pesquisa	**Nº. de Artigos**
IEEE Xplore	92
ACM Digital Library	8
Springer Link	12
Science Direct	35
Google Scholar	50
Total	**197**

Fonte: O autor (2023).

Após esta primeira etapa, os artigos foram examinados. em uma etapa nomeada 'identificar', para descobrir problemas em aberto. Esta analise resultou em alguns temas relevantes como: preocupações relacionadas ao desempenho de blockchains conectadas a sistemas IoT; também há na literatura frequentes discussões sobre a privacidade dos dados gerados por dispositivos IoT; e também sobre a escolha de qual algoritmos de consenso utilizar de forma eficiente em blockchains para IoT.

A Figura A.2 apresenta a sequência de atividades adotadas nesta revisão bibliográfica. Entre os desafios em aberto inicialmente identificados, o mais relevantes e frequentemente citado, foram os relacionados a algoritmos de consenso. Este tema é fundamental para manter a integridade dos dados registrados na blockchain e o desempenho das aplicações. Dentre os 197 trabalhos identificados inicialmente, 75 deles apresentavam o tema 'algoritmos de consenso' como o seu foco principal.

Figura A.2 – Sequência de atividades adotadas na revisão bibliográfica.



Fonte: O autor (2023).

A próxima etapa da pesquisa, nomeada de 'aprofundar', teve como objetivo avaliar a existência de problemas em aberto entre os 75 artigos que tratam de algoritmos de consenso para IoT. O problema mais recorrente citado pelos autores foi que a maioria dos dispositivos IoT possuem limitações de recursos, como memória e processamento, o que torna um desafio a implementação de estratégias robustas de segurança baseadas em criptografia (ANDOLA *et al.*, 2020). Como consequência disto, os dispositivos podem apresentar vulnerabilidades de segurança. A Tabela A.2 mostra o número de artigos recuperados de cada base cientifica nesta fase da pesquisa.

Em seguida, optou-se por aprofundar os estudos sobre casos de combinação entre algoritmos de consenso já propostos na literatura. A Tabela A.3 mostra as bases de dados nas quais os artigos foram selecionados para análise nesta etapa da pesquisa. Este estudo resultou em 18 artigos.

Tabela A.2 – Quantitativo de trabalhos sobre ao uso de algoritmos de consenso em blockchain para IoT.

Dase de Pesquisa	Nº. de Artigos
IEEE Xplore	40
ACM Digital Library	5
Springer Link	3
Science Direct	10
Google Scholar	17
Total	**75**

Fonte: O autor (2023).

Tabela A.3 – Quantitativo de artigos sobre o uso de combinações de algoritmos de consenso em blockchains para IoT

Bases de Pesquisa	Nº. de Artigos
IEEE Xplore	5
ACM Digital Library	2
Springer Link	1
Science Direct	4
Google Scholar	6
Total	**18**

Fonte: O autor (2023).

A.2 SEGUNDA FASE DA REVISÃO DE LITERATURA

A segunda fase da pesquisa bibliográfica teve como objetivo explorar o uso da blockchain baseadas em DAG para ambientes IoT. No inicio dessa fase, as strings de busca (apresentadas no Apêndice A) foram aplicadas às bases científicas listadas na Tabela A.4.

Tabela A.4 – Quantitativo de artigos sobre o uso de blockchain baseadas em DAG para IoT

Bases de Pesquisa	Nº. de Artigos
IEEE Xplore	13
ACM Digital Library	1
Science Direct	3
Google Scholar	5
Total	**22**

Fonte: O autor (2025).

A.3 TERCEIRA FASE DA REVISÃO DE LITERATURA

A terceira fase da pesquisa bibliográfica teve como objetivo analisar o uso de algoritmos genéticos em blockchain para IoT. Nessa fase, as strings de busca (Apêndice A) retornaram resultados apenas nas bases científicas listadas na Tabela A.5. Esta pesquisa retornou 7 artigos relacionados a algoritmos genéticos para blockchain, no entanto apenas 3 são específicos para o uso em blockchain para Internet das Coisas.

Tabela A.5 – Quantitativo de artigos sobre uso de algoritmos genéticos em blockchain para IoT

Bases de Pesquisa	**Nº. de Artigos**
IEEE Xplore	4
Google Scholar	3
Total	**7**

Fonte: O autor (2025).

APÊNDICE B – DADOS DA AVALIAÇÃO DE DESEMPENHO DA BLOCKCHAIN LINEAR

PoW	**CPU**		
	100 nós	500 nós	1000 nós
	52%	53,40%	58%
	RAM		
	100 nós	500 nós	1000 nós
	22,5 MB	27,9 MB	29,3 MB
	REDE (Sent / Received)		
	100 nós	500 nós	1000 nós
	611 KB/s / 566 KB/s	6,39 MB/s / 8,2 MB/s	8,88 MB/s / 8,87 MB/s
	Tempo de Criação de Novos Blocos		
	100 nós	500 nós	1000 nós
	265,17 ms	828,23 ms	3713,71 ms

PoS	**CPU**		
	100 nós	500 nós	1000 nós
	24%	27,00%	34%
	RAM		
	100 nós	500 nós	1000 nós
	22,8 MB	32,1 MB	28,4 MB
	REDE (Sent / Received)		
	100 nós	500 nós	1000 nós
	641 Kb/s / 630 Kb/s	6,8 MB/s / 7,98 MB/s	12,8 MB/s / 11,4 MB/s
	Tempo de Criação de Novos Blocos		
	100 nós	500 nós	1000 nós
	156,47 ms	491,5 ms	1242,92 ms

PoA	**CPU**		
	100 nós	500 nós	1000 nós
	16%	20,00%	30%
	RAM		
	100 nós	500 nós	1000 nós
	23,3 MB	32,5 MB	40,8 MB
	REDE (Sent / Received)		
	100 nós	500 nós	1000 nós
	474 Kb/s / 475 Kb/s	6,06 MB/s / 4,49 MB/s	11 MB/s / 9,09 MB/s
	Tempo de Criação de Novos Blocos		
	100 nós	500 nós	1000 nós
	169,10 ms	598,94 ms	1621,91 ms

PBFT	**CPU**		
	100 nós	500 nós	1000 nós
	15%	20,30%	26%
	RAM		
	100 nós	500 nós	1000 nós
	23,2 MB	23,1 MB	25,4 MB
	REDE (Sent / Received)		
	100 nós	500 nós	1000 nós
	1,513 Mb/s / 1,489 Mb/s	9,930 MB/s / 10,88 MB/s	17,95 MB/s / 18,59 MB/s

	Tempo de Criação de Novos Blocos		
	100 nós	500 nós	1000 nós
	231,48 ms	705,4 ms	2632,1 ms
	CPU		
	100 nós	500 nós	1000 nós
	11%	18,00%	27,80%
	RAM		
	100 nós	500 nós	1000 nós
	23,3 MB	23,8 MB	25,4 MB
RAFT	**REDE (Sent / Received)**		
	100 nós	500 nós	1000 nós
	1,188 Mb/s / 1,184 Mb/s	11,30 MB/s / 12,66 MB/s	18,98 MB/s / 16,58 MB/s
	Tempo de Criação de Novos Blocos		
	100 nós	500 nós	1000 nós
	233,01 ms	625,56 ms	2502,6 ms
	CPU		
	100 nós	500 nós	1000 nós
	33%	43,00%	48,00%
	RAM		
	100 nós	500 nós	1000 nós
	26,1 MB (27,2)	37,1 MB	32,4 MB
PoL	**REDE (Sent / Received)**		
	100 nós	500 nós	1000 nós
	923 Kb/s / 789 Kb/s	4,46 MB/s / 3,4 B/s	13 MB/s / 14,8 MB/s
	Tempo de Criação de Novos Blocos		
	100 nós	500 nós	1000 nós
	157,22 ms	575,42 ms	2642,65 ms
	CPU		
	100 nós	500 nós	1000 nós
	22%	27,00%	39%
	RAM		
	100 nós	500 nós	1000 nós
	28,1 MB	35,1 MB	33,3 MB
PoB	**REDE (Sent / Received)**		
	100 nós	500 nós	1000 nós
	1,03 Mb/s / 1,07 Mb/s	7,2 MB/s / 8,2 MB/s	10,4 MB/s / 9,92 MB/s
	Tempo de Criação de Novos Blocos		
	100 nós	500 nós	1000 nós
	144,23 ms	542,67 ms	2736,88 ms
	CPU		
	100 nós	500 nós	1000 nós
	23%	31,20%	37,80%
	RAM		
	100 nós	500 nós	1000 nós

	28 MB	27,6 MB	48,3 MB
PoI			
	REDE (Sent / Received)		
	100 nós	500 nós	1000 nós
	980 Kb/s / 897 Kb/s	8,7 MB/s / 7,37 MB/s	9,72 MB/s / 8,3 MB/s
	Tempo de Criação de Novos Blocos		
	100 nós	500 nós	1000 nós
	130,89 ms	474,36 ms	1695,03 ms
	CPU		
	100 nós	500 nós	1000 nós
	19%	25,00%	37,20%
	RAM		
	100 nós	500 nós	1000 nós
	29 MB	28,3 MB	34 MB
dPoS			
	REDE (Sent / Received)		
	100 nós	500 nós	1000 nós
	1,15 Mb/s / 1,05 MB/s	9,4 MB/s / 10,4 MB/s	11,2 MB/s / 10,1 MB/s
	Tempo de Criação de Novos Blocos		
	100 nós	500 nós	1000 nós
	144,05 ms	431,25 ms	1493,55 ms
	CPU		
	100 nós	500 nós	1000 nós
	37,00%	42,00%	56,00%
	RAM		
	100 nós	500 nós	1000 nós
	23,9 MB	32,4 MB	36,4 MB
PoET			
	REDE (Sent / Received)		
	100 nós	500 nós	1000 nós
	890 Kb/s / 678 Kb/s	8,24 MB/s / 7,56 MB/s	13,1 MB/s / 14,4 MB/s
	Tempo de Criação de Novos Blocos		
	100 nós	500 nós	1000 nós
	268,76 ms	856,78 ms	4152,1 ms

APÊNDICE C – DADOS DA AVALIAÇÃO DE DESEMPENHO DAS COMBINAÇÕES DA *OMNIBLOCK*

	CPU		
	100 nós	500 nós	1000 nós
	0,60%	10,60%	18,40%
	RAM		
	100 nós	500 nós	1000 nós
	20,2	35	42,8
PoA + PoL	**REDE (Sent / Received)**		
	100 nós	500 nós	1000 nós
	0,111	3,28	6,39
	0,116	5,93	13,2
	Tempo de Criação de Novos Blocos		
	100 nós	500 nós	1000 nós
	89,88	577,94	1927,26

	CPU		
	100 nós	500 nós	1000 nós
	0,80%	7,20%	9,70%
	RAM		
	100 nós	500 nós	1000 nós
	21,9	36,3	51,8
PoA + PoS	**REDE (Sent / Received)**		
	100 nós	500 nós	1000 nós
	0,213	6,72	12,1
	0,229	9,5	16,8
	Tempo de Criação de Novos Blocos		
	100 nós	500 nós	1000 nós
	70,48	609,91	2810,47

	CPU		
	100 nós	500 nós	1000 nós
	1,00%	6,80%	15,70%
	RAM		
	100 nós	500 nós	1000 nós
	22,3	37,3	43,2
PoA + PoW	**REDE (Sent / Received)**		
	100 nós	500 nós	1000 nós
	0,229	7,32	11,5
	0,245	12,8	17
	Tempo de Criação de Novos Blocos		
	100 nós	500 nós	1000 nós
	79,67	612,24	1718,68

	CPU		
	100 nós	500 nós	1000 nós
	8,80%	13,50%	19,70%
	RAM		
	100 nós	500 nós	1000 nós
	27,2	40,9	43,4
PoET + PoA	**REDE (Sent / Received)**		
	100 nós	500 nós	1000 nós
	3,31	4,72	9,39
	4,44	8,64	14,9
	Tempo de Criação de Novos Blocos		
	100 nós	500 nós	1000 nós

	241,86	784,69	941,35

	CPU		
	100 nós	500 nós	1000 nós
	0,50%	6,60%	8,70%
	RAM		
	100 nós	500 nós	1000 nós
	17,9	36,1	44,4
PoB + PBFT	**REDE (Sent / Received)**		
	100 nós	500 nós	1000 nós
	0,187	4,97	9,53
	0,201	5,47	15,6
	Tempo de Criação de Novos Blocos		
	100 nós	500 nós	1000 nós
	62,25	467,05	1116,73

	CPU		
	100 nós	500 nós	1000 nós
	5,10%	6,10%	9,60%
	RAM		
	100 nós	500 nós	1000 nós
	24,2	30,1	33,4
PoB + PoL	**REDE (Sent / Received)**		
	100 nós	500 nós	1000 nós
	0,355	4,57	8,79
	0,89	7,02	15,1
	Tempo de Criação de Novos Blocos		
	100 nós	500 nós	1000 nós
	78,81	603,04	1381,28

	CPU		
	100 nós	500 nós	1000 nós
	1,70%	4,80%	6,60%
	RAM		
	100 nós	500 nós	1000 nós
	22,8	27,9	28,5
PoI + PBFT	**REDE (Sent / Received)**		
	100 nós	500 nós	1000 nós
	0,488	3,71	5,32
	0,822	4,14	13,5
	Tempo de Criação de Novos Blocos		
	100 nós	500 nós	1000 nós
	83,66	477,05	4337,34

	CPU		
	100 nós	500 nós	1000 nós
	1,50%	3,50%	5,00%
	RAM		
	100 nós	500 nós	1000 nós
	20,6	28,3	30,5
PoI + RAFT	**REDE (Sent / Received)**		
	100 nós	500 nós	1000 nós
	0,193	2,93	4,2

	0,846	3,1	8,24
	Tempo de Criação de Novos Blocos		
	100 nós	500 nós	1000 nós
	205,13	352,6	3347,11

	CPU		
	100 nós	500 nós	1000 nós
	1,40%	4,80%	6,20%
	RAM		
	100 nós	500 nós	1000 nós
	23,3	30,2	32
PoS + PBFT	**REDE (Sent / Received)**		
	100 nós	500 nós	1000 nós
	0,285	4,37	5,84
	1,7	6,09	9,57
	Tempo de Criação de Novos Blocos		
	100 nós	500 nós	1000 nós
	182,77	377,68	5155,7

	CPU		
	100 nós	500 nós	1000 nós
	1,30%	4,30%	6,00%
	RAM		
	100 nós	500 nós	1000 nós
	21,2	30,6	39,7
PoL + PBFT	**REDE (Sent / Received)**		
	100 nós	500 nós	1000 nós
	0,253	4,58	7,86
	0,275	7,24	9,71
	Tempo de Criação de Novos Blocos		
	100 nós	500 nós	1000 nós
	179	334,87	1338,73

	CPU		
	100 nós	500 nós	1000 nós
	12,20%	24,00%	37%
	RAM		
	100 nós	500 nós	1000 nós
	37,3	38,4	40,5
PoET + PoW	**REDE (Sent / Received)**		
	100 nós	500 nós	1000 nós
	7,88	12,2	15,2
	10,53	13,1	17,1
	Tempo de Criação de Novos Blocos		
	100 nós	500 nós	1000 nós
	949,48	1369,24	1993,62

	CPU		
	100 nós	500 nós	1000 nós
	1,50%	3,00%	4,30%
	RAM		
	100 nós	500 nós	1000 nós
	24,1	26,8	33,7
PoS + PoB			

PoS + PoB	**REDE (Sent / Received)**		
	100 nós	500 nós	1000 nós
	0,249	3,83	8,83
	0,244	5,17	11,3
	Tempo de Criação de Novos Blocos		
	100 nós	500 nós	1000 nós
	105,5	678,55	1466,16

PoS + PoL	**CPU**		
	100 nós	500 nós	1000 nós
	0,80%	1,08%	1,28%
	RAM		
	100 nós	500 nós	1000 nós
	23	30,4	31,3
	REDE (Sent / Received)		
	100 nós	500 nós	1000 nós
	0,269	7,97	8,12
	0,274	15,6	16
	Tempo de Criação de Novos Blocos		
	100 nós	500 nós	1000 nós
	79,67	606,54	1394,1

PoS + PoW	**CPU**		
	100 nós	500 nós	1000 nós
	0,80%	2,10%	8,80%
	RAM		
	100 nós	500 nós	1000 nós
	23,8	32,3	33,8
	REDE (Sent / Received)		
	100 nós	500 nós	1000 nós
	0,279	2,2	3,34
	0,288	3,19	6,63
	Tempo de Criação de Novos Blocos		
	100 nós	500 nós	1000 nós
	87,13	688,39	1385,4

PoS + RAFT	**CPU**		
	100 nós	500 nós	1000 nós
	0,40%	4,50%	5,10%
	RAM		
	100 nós	500 nós	1000 nós
	19,3	23,6	32,7
	REDE (Sent / Received)		
	100 nós	500 nós	1000 nós
	0,329	1,17	4,28
	0,366	2,03	8,52
	Tempo de Criação de Novos Blocos		
	100 nós	500 nós	1000 nós
	65,4	347,75	3456,72

	CPU		
	100 nós	500 nós	1000 nós
	0,60%	1,60%	5,40%
	RAM		

	100 nós	500 nós	1000 nós
	24,5	37,8	39,8
PoL+ RAFT			
	REDE (Sent / Received)		
	100 nós	500 nós	1000 nós
	0,326	1,11	5,1
	0,351	1,6	11
	Tempo de Criação de Novos Blocos		
	100 nós	500 nós	1000 nós
	64,78	246,21	2606,64

APÊNDICE D – COMPARATIVO DOS ALGORITMOS DE CONSENSO

Consensus Mechanisms for Blockchains Applied to IoT							
Family of Consensus Algorithms	**Scalability**	**Latency**	**Throughput**	**Power Consumption**	**Integrated Algorithms**	**Integration Complexity**	**Blockchain Configuration**	**Adversary Tolerance**
Proof of Work (PoW)	High	High	Low	High	(PoW, PoA), (PoW, PoS), (PoW, PBFT), (PoW, PoA)	High	1	<50%
Proof of Stake (PoS)	High	Medium	Low	Medium	(PoS, PoW)	High	1	<51%
Delegated Proof of Stake (dPoS)	High	Medium	High	Medium	(dPoS, PoP)	Low	1	<51%
Proof of Luck (PoL)	High	Low	High	Low	-	-	1	<50%
Proof of Capacity/ Space (PoC)	High	High	Low	Low	(PoC, VCF)	Low	1	<50%
Proof of Activity (PoA)	High	Medium	Low	Medium	(PoA, PoW)	High	1	51%
Proof of Burn (PoB)	High	High	Low	Medium	-	-	1	<25%
Proof of Importance (PoI)	High	Medium	High	Low	-	-	1	<51%
Proof of Elapsed Time (PoET)	High	Low	High	Low	(PoET, Raft), (PoET, PBFT)	Low	1	N/A
Practical Byzantine Fault Tolerance (PBFT)	Low	Low	High	Low	(PBFT, PoW), (PBFT, WBC), (PBFT, Raft), (PBFT, PoET)	Medium	1	<33%
dPBFT	High	Medium	High	Low	-	-	1	<33%
Tendermint	High	Low	High	Low	-	-	1	<33%
Raft	High	Low	High	Low	(Raft, PoET), (Raft, PBFT)	Low	1	<50%
Ripple	High	Medium	High	Low	-	-	2	20%
Stellar	High	Medium	High	Low	-	-	2	Variable

APÊNDICE E – TECNOLOGIAS UTILIZADAS

A seguir são descritas as principais tecnologias utilizadas para o desenvolvimento da solução proposta nesta tese:

- **Ubuntu 24.04 LTS**: sistema Operacional usado como SO base das máquinas/contêineres que executaram a plataforma e os testes;
- **Docker Engine (v28.x)**: docker foi utilizado para encapsular cada nó da OmniBlock em um contêiner isolado, facilitando a replicação do ambiente, a gestão de múltiplas instâncias (nós) em um *cluster* Docker Swarm e a automação dos experimentos. Cada contêiner representa um nó DAG que armazena uma cópia do *ledger* e executa a pilha de software necessária (rede, consenso, armazenamento). O uso de contêineres permitiu experimentar diferentes topologias e escalas com baixo custo de provisão;
- **Node.js (v24 runtime)**: as implementações dos diversos algoritmos de consenso, das integrações entre eles, e dos algoritmos genéticos foram desenvolvidas em Node.js. A escolha por Node.js deveu-se à sua rapidez de prototipação, apoio a I/O assíncrono (importante para simular troca de mensagens) e facilidade de integração com bibliotecas criptográficas e ferramentas de rede;
- **Prometheus (v3.5.0)**: responsável pela coleta e armazenamento das métricas experimentais (CPU, memória, vazão, etc.) e pela exportação para análise posterior;
- **cAdvisor (v0.53.0)**: ferramenta complementar para coleta detalhada de métricas de contêiner (uso por contêiner, estatísticas de I/O) integrada ao pipeline de monitoramento;
- **Grafana (v12.x)**: utilizada para visualização das métricas (*dashboards* interativos) e geração dos gráficos apresentados nos experimentos; e
- **Hardware de execução**: os experimentos foram executados em máquinas com processador Intel Core i5-10210U e 8 GB de RAM, além de laboratórios com vários *hosts* para simulação distribuída via Docker.