Mateus Cavalcanti Alves Teixeira Silva

Votação Eletrônica Descentralizada com Blockchain e Proof of Authority

Natal – RN

Junho de 2026

Mateus Cavalcanti Alves Teixeira Silva

Votação Eletrônica Descentralizada com Blockchain e Proof of Authority

Trabalho de Conclusão de Curso de Engenharia de Computação da Universidade Federal do Rio Grande do Norte, apresentado como requisito parcial para a obtenção do grau de Bacharel em Engenharia de Computação

Orientador: Prof. Dr. Eduardo de Lucena Falcão

Universidade Federal do Rio Grande do Norte – UFRN

Departamento de Engenharia de Computação e Automação – DCA

Curso de Engenharia de Computação

Natal – RN

Junho de 2026

Mateus Cavalcanti Alves Teixeira Silva

Votação Eletrônica Descentralizada com Blockchain e Proof of Authority

Trabalho de Conclusão de Curso de Engenharia de Computação da Universidade Federal do Rio Grande do Norte, apresentado como requisito parcial para a obtenção do grau de Bacharel em Engenharia de Computação

Orientador: Prof. Dr. Eduardo de Lucena Falcão

Trabalho aprovado. Natal – RN, ___ de _____ de _____:

Prof. Dr. Eduardo de Lucena Falcão
Orientador
Universidade Federal do Rio Grande do Norte

Prof. Dr. Samuel Xavier de Souza
Membro avaliador
Universidade Federal do Rio Grande do Norte

Prof. Dr. Carlos Manuel Dias Viegas
Membro avaliador
Universidade Federal do Rio Grande do Norte

Natal – RN
Junho de 2026

Ao meu cachorro Bravo, que esteja correndo atrás de bolinhas em um lindo lugar.

AGRADECIMENTOS

Finalizar uma graduação não é, de maneira alguma, uma conquista individual. Ao longo de cinco anos que empilharam trabalhos, provas, pesquisas, incertezas e pequenas vitórias, contei com o apoio incondicional de pessoas que me sustentaram nos dias em que tudo parecia desabar e multiplicaram minha alegria nos bons momentos, que foram muitos e constituem a maior parte desta trajetória. Como já reconhecia Aristóteles, a eudaimonia (vida plena) não se realiza isoladamente, mas se constrói na presença daqueles com quem partilhamos o caminho.

Assim, agradeço primeiramente à minha família, pelo amor incondicional que me acompanhou em todos os momentos e escolhas. Embora eu não tenha chegado ao espaço, como sonhava na infância quando queria ser astronauta, foi esse amor que me impulsionou a chegar onde eu gostaria. Pai e mãe, estar com vocês é estar em casa.

À Fernanda, que todos os dias acompanha minhas conquistas, meus cansaços, minhas queixas de trabalho e minhas ideias inventadas no chuveiro. E que, por algum motivo, é a única pessoa neste mundo que gosta da minha comida. Amor, graças a você posso enfim citar o heterônimo Alberto Caeiro quando diz: “O amor é uma companhia. Já não sei andar só pelos caminhos, porque já não posso andar só”.

Aos professores que contribuíram para minha formação, em especial ao meu orientador, Prof. Dr. Eduardo de Lucena Falcão, que ouviu pela primeira vez a ideia deste trabalho em uma aula de Sistemas Distribuídos e, mesmo quando ela ainda parecia um pouco mirabolante, teve a paciência de levá-la adiante comigo. Agradeço pela orientação, pela disponibilidade e por ter me ajudado durante todo o processo de desenvolvimento deste trabalho.

Por fim, a todos os meus amigos, que tornaram esta caminhada mais leve. Vocês ouviram minhas reclamações, celebraram minhas pequenas e grandes vitórias, dividiram comigo conversas sérias e risadas. Se concluir uma graduação não é um ato individual, é porque, em muitos momentos, foram vocês que me lembraram que o caminho também podia ser alegre.

"And in the end, the love you take is equal to the love you make."

— The Beatles, *The End*

RESUMO

Sistemas de votação eletrônica exigem mecanismos capazes de preservar integridade, unicidade de voto, disponibilidade e auditabilidade. Em arquiteturas digitais, esses requisitos dependem não apenas da interface de votação, mas também do modo como o sistema registra votos, valida transações e organiza a confiança entre os participantes. Este trabalho propõe, implementa e avalia um protótipo de sistema de votação eletrônica descentralizado baseado em rede peer-to-peer, blockchain e consenso Proof of Authority (PoA), com foco em redes permissionadas compostas por validadores previamente conhecidos. A solução foi desenvolvida em Go com uso da biblioteca libp2p, adotando arquitetura modular, processamento orientado a eventos e seleção determinística de líder por round-robin. A validação experimental foi conduzida em ambiente controlado com três nós validadores, por meio de testes de integridade do resultado, prevenção de votação dupla e tolerância a falhas do tipo crash-stop. Os resultados mostraram que o sistema finaliza votos válidos corretamente, preserva a unicidade no estado final da blockchain e mantém operação após a falha de um validador. Conclui-se que a abordagem proposta é tecnicamente viável no escopo experimental avaliado, constituindo uma base para evoluções futuras em cenários institucionais de menor porte.

Palavras-chave: votação eletrônica; blockchain; proof of authority; sistemas distribuídos; consenso distribuído.

ABSTRACT

Electronic voting systems require mechanisms capable of preserving integrity, vote uniqueness, availability, and auditability. In digital architectures, these requirements depend not only on the voting interface, but also on how the system records votes, validates transactions, and organizes trust among participants. This work proposes, implements, and evaluates a prototype of a decentralized electronic voting system based on peer-to-peer networking, blockchain, and Proof of Authority (PoA) consensus, focusing on permissioned networks composed of previously known validators. The solution was implemented in Go using the libp2p library, adopting a modular architecture, event-driven processing, and deterministic round-robin leader selection. Experimental validation was conducted in a controlled environment with three validator nodes through tests covering result integrity, double-voting prevention, and crash-stop fault tolerance. The results showed that the system correctly finalizes valid votes, preserves vote uniqueness in the final blockchain state, and maintains operation after the failure of one validator. These findings indicate that the proposed approach is technically feasible within the evaluated experimental scope and provides a basis for future improvements in smaller institutional scenarios.

Keywords: electronic voting; blockchain; proof of authority; distributed systems; distributed consensus.

LISTA DE ILUSTRAÇÕES

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

BFT	*Byzantine Fault Tolerance* (Tolerância a Falhas Bizantinas)
DDD	*Domain-Driven Design* (Design Orientado a Domínio)
DHT	*Distributed Hash Table* (Tabela de Hash Distribuída)
P2P	*Peer-to-Peer* (Par-a-Par)
PBFT	*Practical Byzantine Fault Tolerance* (Tolerância Prática a Falhas Bizantinas)
PoA	*Proof of Authority* (Prova de Autoridade)
PoS	*Proof of Stake* (Prova de Participação)
PoW	*Proof of Work* (Prova de Trabalho)
WAN	*Wide Area Network* (Rede de Área Ampla)

LISTA DE SÍMBOLOS

$\mathcal{M}$	Máquina de estados finita
$\mathcal{P}$	Conjunto das partes (conjunto potência)
$\mathcal{A}$	Conjunto de todas as aprovações possíveis
S	Conjunto de estados da máquina de estados
$\mathcal{E}$	Conjunto de eventos
δ	Função de transição de estados
γ	Função de transição de estado da blockchain
Ω_h	Estado global do nó na altura h
σ_h	Estado da aplicação após finalização do bloco de altura h
$B(h)$	Buffer de aprovações pendentes na altura h
$Q(h)$	Conjunto de aprovações válidas processadas na altura h
$M(h)$	Conjunto de votos disponíveis para a altura h
$P(h)$	Proposta de bloco para a altura h
$A_i(h)$	Aprovação emitida pelo validador v_i na altura h
$L(h)$	Líder determinado para a altura h
n	Número total de validadores
f	Número máximo de falhas toleradas
h	Altura do bloco na blockchain
$\lfloor x \rfloor$	Função piso (maior inteiro menor ou igual a x)
$\lceil x \rceil$	Função teto (menor inteiro maior ou igual a x)
$\rightarrow$	Mapeamento ou transição
$\rightarrow$	Transição com evento
$\prec$	Relação de precedência

$\in$	Pertence
$\subseteq$	Subconjunto
$\cup$	União de conjuntos
$\varnothing$	Conjunto vazio
$\forall$	Para todo (quantificador universal)
$\exists$	Existe (quantificador existencial)
$\geq$	Maior ou igual
mod	Operador módulo (resto da divisão)

SUMÁRIO

1 INTRODUÇÃO

A história dos sistemas de votação pode ser lida como uma busca contínua por mecanismos capazes de transformar escolhas individuais em resultados coletivos legítimos. Seja por manifestação pública, cédulas em papel ou sistemas eletrônicos, o problema central permanece o mesmo: registrar a vontade do eleitor, preservar a integridade do processo e permitir que o resultado seja aceito pelos participantes.

Nesse sentido, a cédula em papel consolidou propriedades importantes, como materialidade do registro, possibilidade de recontagem e separação física entre voto e apuração. Ao mesmo tempo, ela desloca parte da confiança para procedimentos de custódia, transporte, contagem manual e fiscalização local. A informatização do voto surge, nesse contexto, como uma tentativa de reduzir erros operacionais, acelerar a totalização e padronizar etapas do processo eleitoral. A experiência brasileira ilustra bem essa tensão: a votação eletrônica reduziu votos residuais e ampliou o exercício efetivo do voto, mas também produziu novas formas de erro associadas à interação entre eleitor e sistema (JR., 2016).

Esse percurso mostra que a evolução tecnológica não elimina automaticamente os desafios de confiança, apenas muda onde eles aparecem. Em sistemas digitais, propriedades como auditabilidade, sigilo do voto, verificabilidade, usabilidade, acessibilidade e clareza do modelo de confiança tornam-se critérios centrais para avaliar a qualidade da solução (LOWRY; VORA, 2009). Assim, um sistema de votação eletrônica não deve ser analisado apenas por ser informatizado, mas pela forma como protege a integridade do processo.

A motivação deste trabalho, portanto, parte da constatação de que, em sistemas de votação digitais, a confiança passa a depender de componentes menos visíveis ao eleitor, como software, infraestrutura de rede, operadores, mecanismos de auditoria e procedimentos de validação. A centralização desses componentes pode simplificar a operação, mas também concentra responsabilidades críticas em poucos pontos do sistema. Uma falha técnica ou uma decisão não auditável em um componente central pode comprometer a percepção de legitimidade do processo.

1.1 Problema de Pesquisa e Objetivos

Tecnologias de sistemas distribuídos e *blockchain* oferecem um caminho possível para essa investigação. Ao distribuir a validação entre múltiplos nós, uma arquitetura descentralizada reduz a dependência de um único servidor ou operador. Ao registrar dados em uma cadeia encadeada por funções criptográficas, torna-se possível detectar alterações

retroativas no histórico. E, ao empregar mecanismos de consenso, os nós participantes podem concordar sobre uma mesma ordem de eventos sem depender de uma base de dados central.

Ainda assim, aplicar *blockchain* a votação eletrônica não é uma consequência direta. Redes abertas baseadas em *Proof of Work* tendem a apresentar custos computacionais e latências incompatíveis com aplicações de propósito restrito. Alternativas baseadas em *Proof of Stake* introduzem mecanismos econômicos que nem sempre fazem sentido em eleições, pois o poder de validação passa a depender de participação financeira ou de regras de incentivo. Estudos recentes sobre votação eletrônica baseada em *blockchain* reforçam justamente essa dificuldade: segurança, auditabilidade e descentralização precisam ser conciliadas com desempenho, custo operacional e simplicidade de implantação (SHARMA; KRISHNA; BAHGA, 2021).

O consenso *Proof of Authority* (PoA) se encaixa nesse recorte por partir de uma premissa diferente. Em vez de selecionar validadores por competição computacional ou participação econômica, o PoA opera em redes permissionadas, nas quais os nós validadores são previamente conhecidos e autorizados. Essa escolha reduz a complexidade do consenso e torna o modelo adequado ao escopo experimental deste trabalho, que não busca substituir uma infraestrutura eleitoral nacional, mas analisar a preservação de propriedades fundamentais em uma rede de validadores conhecidos.

Este trabalho investiga, em escala experimental, se uma arquitetura permissionada baseada em rede *peer-to-peer*, *blockchain* e consenso *Proof of Authority* é capaz de sustentar um sistema de votação eletrônica atendendo a requisitos fundamentais de integridade, unicidade de voto e tolerância a falhas. O problema de pesquisa, portanto, concentra-se na seguinte questão: uma rede permissionada com validadores conhecidos, registro em *blockchain* e consenso PoA consegue preservar propriedades essenciais de um processo de votação sem depender de um servidor central de decisão?

A partir dessa questão, o objetivo geral consiste em propor, implementar e avaliar um protótipo que materialize essa abordagem em um ambiente distribuído. Para isso, o trabalho desenvolve uma aplicação em Go, utiliza a biblioteca libp2p para comunicação *peer-to-peer*, estrutura um núcleo de consenso PoA com seleção determinística de líder e implementa mecanismos de validação de votos, prevenção de duplicidade, persistência da cadeia e finalização mediante quórum.

De forma mais específica, busca-se modelar o sistema como uma máquina de estado replicada, materializar essa modelagem em um protótipo executável, validar o ciclo completo de submissão e finalização de votos e observar o comportamento da rede diante de votos duplicados e falhas por parada. A avaliação não tem como objetivo medir desempenho em escala de produção, mas verificar se as propriedades funcionais centrais

são preservadas dentro do ambiente experimental definido.

1.2 Questões Investigativas

As questões investigativas derivam do problema de pesquisa e delimitam as propriedades observadas no escopo experimental deste trabalho. Como a validação foi conduzida em uma rede permissionada local com o número mínimo de validadores necessário para observar consenso e tolerância a uma falha por parada, as perguntas não buscam generalizar o comportamento do sistema para qualquer quantidade de nós, mas avaliar sua correção dentro desse cenário controlado. A Tabela 1 apresenta essas questões de forma sintética.

Tabela 1 – Questões investigativas do trabalho

Questão	**Formulação**
Q1	No cenário experimental com número mínimo de validadores, o sistema valida e finaliza corretamente os votos submetidos?
Q2	No mesmo cenário, o sistema detecta e rejeita tentativas de voto duplicado sem comprometer a integridade da *blockchain*?
Q3	Na rede experimental avaliada, o sistema mantém capacidade de finalização após a falha por parada de um validador?

Fonte: Elaborado pelo autor.

Essas questões orientam tanto a modelagem do sistema quanto a interpretação dos resultados apresentados nos capítulos seguintes. Dessa forma, a contribuição do trabalho não se limita à proposição conceitual da arquitetura, mas inclui a implementação de um protótipo funcional e sua análise empírica em ambiente controlado.

1.3 Organização do Trabalho

Este trabalho está organizado em seis capítulos, estruturados de forma a conduzir o leitor desde os fundamentos teóricos até a validação experimental do sistema proposto.

O Capítulo 2 apresenta a fundamentação teórica necessária para compreensão do trabalho, cobrindo sistemas distribuídos, redes peer-to-peer, mecanismos de consenso e fundamentos de blockchain. Particular atenção é dedicada ao problema dos generais bizantinos, aos algoritmos de tolerância a falhas bizantinas e às características do consenso Proof of Authority.

O Capítulo 3 descreve a metodologia empregada no desenvolvimento do sistema, incluindo decisões arquiteturais, escolha de tecnologias e estratégia de validação experimental. A arquitetura modular baseada em Hexagonal Architecture é apresentada em

detalhe, assim como o processamento orientado a eventos e a adaptação do consenso PoA para eleições. A formalização matemática das propriedades de segurança e vivacidade estabelece critérios objetivos para validação da correção do sistema.

O Capítulo 4 apresenta os detalhes de implementação do sistema, cobrindo a estrutura de dados da blockchain, o mecanismo de consenso, a camada de rede peer-to-peer e os componentes de validação e persistência. Decisões de design são justificadas com base em requisitos funcionais e não funcionais, e a integração entre componentes é descrita através de diagramas e exemplos de fluxo de execução.

O Capítulo 5 apresenta os resultados experimentais obtidos através de três testes sistemáticos: validação funcional, unicidade de voto e tolerância a falhas. Cada teste é descrito em termos de configuração, execução e análise de resultados, com métricas quantitativas coletadas e interpretadas à luz das propriedades formais estabelecidas no Capítulo 3. A discussão integrada dos resultados identifica pontos fortes, limitações e direções para trabalhos futuros.

O Capítulo 6 conclui o trabalho, sintetizando as contribuições realizadas, discutindo o alcance das questões investigativas e apresentando limitações e trabalhos futuros.

2 FUNDAMENTAÇÃO TEÓRICA

Sistemas de votação eletrônica descentralizados inserem-se no contexto dos sistemas distribuídos, nos quais múltiplos nós autônomos cooperam por troca de mensagens para manter um serviço comum (TANENBAUM; STEEN, 2017; COULOURIS et al., 2012). No problema abordado neste trabalho, essa distribuição está diretamente relacionada à redução de pontos únicos de falha, à auditabilidade do processo e à continuidade de operação mesmo diante de falhas individuais.

Dentro desse contexto, três fundamentos sustentam a proposta desenvolvida. O primeiro é a rede *peer-to-peer*, responsável pela comunicação direta entre os participantes. O segundo é a estrutura de registro imutável da *blockchain*, utilizada para organizar votos em uma cadeia verificável de blocos. O terceiro é o mecanismo de consenso, que permite aos validadores concordarem sobre quais blocos devem compor a cadeia. As seções seguintes apresentam esses conceitos nessa ordem.

2.1 Redes *peer-to-peer*

Redes *peer-to-peer* (P2P), ou redes par-a-par, organizam aplicações distribuídas de modo que os nós possam simultaneamente consumir e prover recursos, reduzindo a dependência de um coordenador central e distribuindo a responsabilidade pela comunicação entre os participantes (TANENBAUM; STEEN, 2017; COULOURIS et al., 2012). Em sistemas de votação descentralizados, essa característica é relevante porque a propagação de votos, propostas e aprovações não deve depender de um único servidor ou ponto de intermediação.

Na prática, redes P2P operam sobre uma sobreposição lógica, isto é, uma rede construída em nível de aplicação sobre a infraestrutura física existente. Nessa sobreposição, os pares precisam descobrir vizinhos, trocar mensagens e manter a conectividade apesar da entrada, saída ou falha de nós. Esse modelo introduz desafios de descoberta, roteamento e disseminação, mas oferece, como contrapartida, maior resiliência e melhor adequação a aplicações que exigem distribuição de responsabilidade e tolerância a falhas.

As topologias de redes P2P podem ser descritas, de forma geral, como não estruturadas, estruturadas ou híbridas. Nas não estruturadas, a conectividade emerge de decisões locais e a busca tende a depender de difusão oportunística. Nas estruturadas, mecanismos como tabelas de dispersão distribuídas, conhecidas como *Distributed Hash Tables* (DHTs), organizam identificadores e rotas, oferecendo descoberta mais previsível (STOICA et al., 2001). Já as abordagens híbridas combinam coordenação limitada com comunicação direta

entre pares.

A comunicação em redes P2P pode empregar estratégias de difusão ampla, como inundação, ou estratégias epidêmicas, como fofoca, cada uma com diferentes compromissos entre cobertura, latência e custo de rede (COULOURIS et al., 2012; Gnutella Developers, 2001; TANENBAUM; STEEN, 2017). Em linhas gerais, a inundação tende a maximizar o alcance ao custo de maior sobrecarga, enquanto a fofoca reduz o tráfego por meio de disseminação probabilística, preservando boa escalabilidade em ambientes dinâmicos. Essa base de comunicação é importante porque, em uma rede de votação descentralizada, votos e mensagens de consenso precisam alcançar os validadores de forma eventual e redundante.

2.2 Fundamentos de *blockchain*

A tecnologia *blockchain* pode ser compreendida como um livro-razão distribuído, imutável e compartilhado entre diversos nós. Um livro-razão, nesse contexto, é um registro ordenado de operações. No caso de uma *blockchain*, esse registro é dividido em blocos encadeados por funções criptográficas de resumo, chamadas *hashes*. Cada *hash* funciona como uma impressão digital dos dados: pequenas alterações no conteúdo produzem identificadores diferentes, permitindo detectar adulterações.

A concepção moderna de *blockchain* está intimamente ligada ao trabalho de Nakamoto (NAKAMOTO, 2008). Nesse modelo, transações são agrupadas em blocos, cada bloco referencia o *hash* do bloco anterior e a rede adota uma regra de consenso para decidir qual cadeia representa o histórico válido. Esse encadeamento cria uma relação de dependência entre os blocos: alterar um registro antigo exigiria recalcular os blocos subsequentes e convencer a rede a aceitar a nova versão.

A Figura 1 ilustra essa organização. Cada bloco armazena um conjunto de transações, possui um *hash* próprio e inclui uma referência ao *hash* do bloco anterior. Essa referência cria o encadeamento lógico entre os registros e permite que qualquer alteração em um bloco anterior seja percebida nos blocos seguintes.

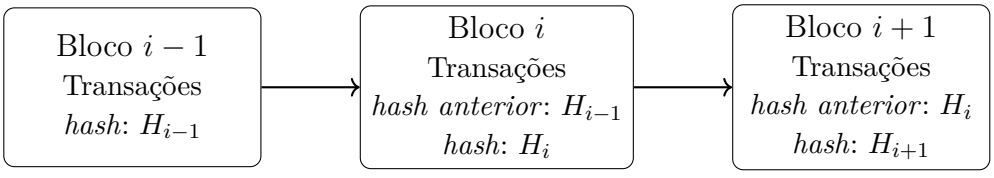

Figura 1 – Encadeamento de blocos por referências criptográficas

Fonte: Elaborado pelo autor.

Um dos problemas clássicos tratados por Nakamoto é o gasto duplo, isto é, a tentativa de utilizar a mesma unidade digital mais de uma vez. Em sistemas centralizados,

esse problema é resolvido por uma autoridade que mantém o registro oficial. Em uma *blockchain*, a solução depende de um histórico compartilhado e verificável, no qual todos os participantes podem conferir a ordem das transações e rejeitar usos incompatíveis do mesmo recurso.

Embora o termo *blockchain* só tenha se popularizado mais tarde, os primeiros trabalhos conceituais com cadeias de blocos criptograficamente encadeados remontam à década de 1990. Em 1991, Haber e Stornetta descreveram um sistema para marcar digitalmente documentos com data e hora de modo resistente a alterações posteriores, estabelecendo bases importantes para estruturas de registro imutável (HABER; STORNETTA, 1991).

Essas propriedades tornam a *blockchain* relevante para aplicações que exigem auditabilidade e resistência a alterações retroativas. Em um sistema de votação, por exemplo, a cadeia pode ser usada para registrar votos ou resultados parciais de maneira verificável, desde que o protocolo também defina quem pode inserir novos blocos e como os nós concordam sobre a ordem dos registros.

2.2.1 Paradigmas e a máquina de estado transacional

A tecnologia *blockchain* também pode ser compreendida como uma máquina de estado replicada. Uma máquina de estado é um modelo no qual o sistema possui um estado atual e evolui por meio da aplicação de transições. Em uma *blockchain*, cada nó mantém uma cópia do estado, e os blocos representam a sequência de transições que conduzem do estado inicial ao estado atual.

Formalmente, o funcionamento de uma *blockchain* pode ser descrito como uma função de transição de estado (WOOD, 2014):

$$\gamma : (S, T) \to S'$$

onde S é o estado atual, T é uma transação e S' é o novo estado após a execução de T. Cada bloco contém um conjunto ordenado de transações $\{T_1, T_2, \ldots, T_n\}$, aplicadas sequencialmente sobre o estado anterior:

$$S_k = \gamma(S_{k-1}, T_k), \quad 1 \leq k \leq n$$

Assim, a cadeia completa dos blocos representa uma série contínua de transformações de estado, verificável por qualquer participante da rede.

Esse modelo é especialmente importante para votação eletrônica porque permite interpretar cada voto válido como uma transição de estado. Ao aplicar um voto, o estado do sistema passa a registrar que determinado eleitor já participou, o que contribui para a prevenção de votação dupla. Entretanto, para que todos os nós apliquem as mesmas transições na mesma ordem, é necessário um mecanismo de consenso.

2.3 Mecanismos de consenso em redes *peer-to-peer*

Uma vez definida a *blockchain* como estrutura de registro compartilhado, surge o problema de decidir quais blocos devem ser aceitos e em que ordem eles devem ser adicionados à cadeia. Em sistemas distribuídos, essa decisão não pode depender apenas da troca de mensagens, pois mensagens podem atrasar, chegar fora de ordem ou ser produzidas por participantes defeituosos. A metáfora clássica do problema dos generais bizantinos ilustra essa dificuldade: participantes que precisam coordenar uma decisão comum podem receber informações contraditórias ou não confiáveis, tornando necessário um protocolo explícito de coordenação (LAMPORT; SHOSTAK; PEASE, 1982).

Nesse contexto, um mecanismo de consenso pode ser definido como o conjunto de regras que permite a múltiplos nós chegar a uma decisão comum sobre o estado do sistema. Essa decisão pode corresponder, por exemplo, à validade de um bloco, à ordem das transações ou à atualização de uma máquina de estado replicada. Independentemente do mecanismo adotado, a camada P2P sustenta a disseminação das mensagens necessárias ao consenso, como propostas, aprovações ou confirmações.

Uma falha bizantina representa o tipo mais severo de falha em sistemas distribuídos: um processo pode adotar comportamento arbitrário, incluindo omitir mensagens, enviar informações incorretas ou encaminhar mensagens contraditórias para diferentes participantes (TANENBAUM; STEEN, 2017). Diferentemente de falhas por parada, em que o nó simplesmente deixa de responder, falhas bizantinas comprometem a confiabilidade do próprio conteúdo das mensagens.

2.3.1 Panorama de mecanismos de consenso

Os mecanismos de consenso podem ser compreendidos como respostas distintas ao mesmo problema de coordenação distribuída. Protocolos clássicos de tolerância a falhas bizantinas, como o *Practical Byzantine Fault Tolerance* (PBFT), buscam preservar acordo e consistência mesmo quando parte dos participantes pode se comportar de forma arbitrária ou maliciosa. Em geral, esses protocolos exigem que um sistema com n réplicas possua $n \geq 3f + 1$ para tolerar até f falhas bizantinas, oferecendo forte robustez ao custo de maior coordenação entre réplicas (LAMPORT; SHOSTAK; PEASE, 1982; CASTRO; LISKOV, 1999).

Em redes públicas, nas quais os participantes não são previamente conhecidos, mecanismos do tipo *Proof of X*, isto é, mecanismos baseados em alguma forma de prova verificável, utilizam critérios específicos para selecionar quem pode propor ou validar blocos. O *Proof of Work* (PoW), introduzido por Nakamoto, usa esforço computacional como critério de decisão, tornando ataques custosos, mas exigindo consumo contínuo de processamento e energia (NAKAMOTO, 2008). O *Proof of Stake* (PoS), por sua vez,

substitui o poder computacional pela participação econômica, reduzindo o custo energético, mas introduzindo desafios próprios de governança e incentivos (CONG et al., 2025).

Essas alternativas evidenciam que não existe um mecanismo de consenso universalmente superior. Cada abordagem assume compromissos entre abertura da rede, custo operacional, desempenho, governança e modelo de confiança. Como este trabalho investiga uma rede permissionada de votação, isto é, uma rede com participação controlada, na qual os validadores podem ser previamente identificados e autorizados, o mecanismo mais alinhado ao escopo proposto é o *Proof of Authority*.

2.3.2 *Proof of Authority*

O *Proof of Authority* é um mecanismo de consenso voltado a redes permissionadas. Diferentemente do PoW, em que a autoridade para propor blocos deriva de esforço computacional, ou do PoS, em que deriva de participação econômica, no PoA a capacidade de validar blocos decorre da identidade e da autoridade atribuída a um conjunto limitado de validadores (MANOLACHE; MANOLACHE; TAPUS, 2022).

Essa característica torna o PoA particularmente adequado a contextos em que a rede não precisa ser completamente aberta, mas ainda se deseja distribuir a responsabilidade de validação entre múltiplos participantes. Em vez de competir por recursos computacionais, os validadores seguem regras determinísticas para propor, assinar e validar blocos. Com isso, o protocolo reduz custo operacional, simplifica a coordenação e permite tempos de confirmação menores, propriedades relevantes para aplicações privadas, corporativas, institucionais ou experimentais.

Uma forma comum de organizar a proposição de blocos em redes PoA é a rotação determinística de validadores. No escalonamento *round-robin*, termo usual para uma alternância circular entre participantes, os validadores autorizados são organizados em uma ordem fixa, e a responsabilidade de propor blocos alterna ciclicamente entre eles. Assim, se $V = \{v_0, v_1, \ldots, v_{n-1}\}$ representa o conjunto ordenado de validadores, a liderança de uma determinada altura da cadeia, ou seja, da posição sequencial do bloco, pode ser definida em função do índice desse bloco. Com isso, todos os nós conseguem determinar localmente qual validador deve propor o próximo bloco.

A Figura 2 ilustra o fluxo conceitual de uma rodada PoA baseada em validadores autorizados. O processo começa com a definição do conjunto de validadores, passa pela seleção determinística do líder, segue com a proposição e assinatura do bloco e termina com a validação pelos demais participantes antes da finalização.

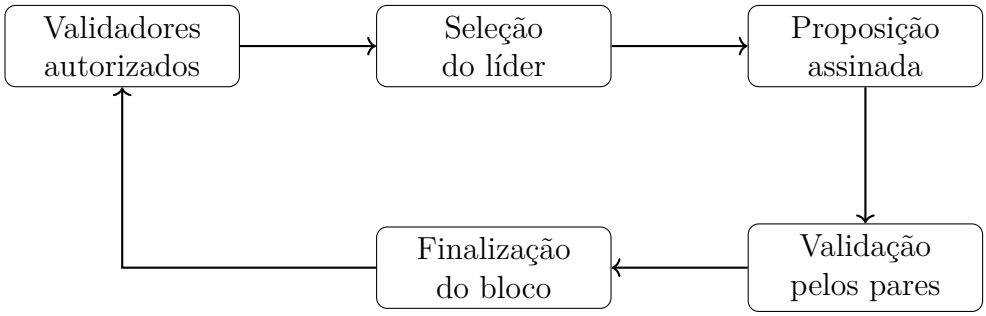

Figura 2 – Fluxo conceitual de uma rodada *Proof of Authority*

Fonte: Elaborado pelo autor.

O principal benefício dessa abordagem é a previsibilidade. Como o conjunto de validadores é conhecido e a escolha do propositor segue uma regra determinística, a rede evita a competição intensiva por recursos e reduz a sobrecarga de coordenação. Além disso, a responsabilidade de proposição pode ser distribuída entre os validadores, evitando que um único nó concentre todo o processo de criação de blocos.

Por outro lado, o PoA desloca parte da confiança do poder computacional ou econômico para a governança da rede. A segurança do protocolo depende da correta seleção, identificação e supervisão dos validadores autorizados. Caso o conjunto de validadores seja pequeno ou controlado por uma única entidade, a descentralização efetiva do sistema é reduzida. Além disso, validadores maliciosos podem tentar censurar transações, atrasar propostas ou agir de maneira coordenada para comprometer a cadeia. Por essa razão, o PoA é mais adequado a redes permissionadas, nas quais a identidade dos validadores e as regras de participação podem ser definidas previamente.

No contexto deste trabalho, essas características tornam o PoA uma escolha coerente com o objetivo de construir um protótipo mínimo viável de votação descentralizada em ambiente controlado. O sistema proposto não busca operar como uma rede pública aberta, mas como uma rede permissionada de validadores conhecidos, na qual a prioridade está em avaliar integridade, unicidade de voto, auditabilidade e tolerância a falhas por parada com menor complexidade operacional.

2.4 Trabalhos relacionados

A aplicação de *blockchain* a sistemas de votação eletrônica tem sido explorada em diferentes direções, especialmente a partir de plataformas já consolidadas, como Ethereum, e de mecanismos de consenso voltados a redes permissionadas. Os trabalhos analisados nesta seção foram selecionados por tratarem de votação ou tomada de decisão apoiada por *blockchain*, com atenção especial ao uso de consenso PoA, contratos inteligentes, auditabilidade e desempenho.

Christyono, Widjaja e Wicaksana (CHRISTYONO; WIDJAJA; WICAKSANA, 2021) propõem um sistema de votação eletrônica baseado em Ethereum, implementado com Go-Ethereum e consenso Clique, uma variante de Proof of Authority. O trabalho utiliza contratos inteligentes para organizar o processo eleitoral e avalia o sistema com dados reais da Comissão Eleitoral da Indonésia, considerando 26 candidatos e 15.725 eleitores do subdistrito de Jelupang. Nos experimentos reportados, a rede foi configurada com três mineradores, tempo de bloco de cinco segundos e limite de gás de 8.000.000 wei. Os autores informam que todos os votos foram registrados com sucesso na *blockchain*, com vazão aproximada de 23 transações por segundo e tempo total de 12,75 minutos para confirmação dos 15.725 votos. O estudo também aponta um custo operacional relevante: a criação das contas de todos os eleitores demandou 13,4 horas. Além disso, os próprios autores observam que, caso a eleição em escala provincial fosse conduzida em uma única *blockchain*, o processo poderia demandar mais de 71 horas, tornando essa configuração inadequada para o prazo eleitoral considerado.

Sharma, Krishna e Bahga (SHARMA; KRISHNA; BAHGA, 2021) investigam uma abordagem distinta, baseada em Proof-of-Stake-Voting sobre TomoChain, com o objetivo de reduzir custo de transação e aumentar a vazão de um sistema de votação auditável. O trabalho implementa uma aplicação descentralizada de votação e compara três mecanismos de consenso: Proof of Work, Proof of Authority e Proof-of-Stake-Voting. A avaliação considera métricas de custo de gás, custo total em Ether e transações por segundo. Segundo os resultados apresentados, o protocolo PoSV alcançou 4.669 transações por segundo, enquanto PoA atingiu 2.390 e PoW 395. O custo total também foi menor no caso do PoSV, reportado como $3{,}1 \times 10^{-7}$ ETH, em comparação com $2{,}49759 \times 10^{-4}$ ETH para PoA e $5{,}42536 \times 10^{-4}$ ETH para PoW. O foco do estudo está, portanto, na comparação econômica e de desempenho entre mecanismos de consenso em um ambiente compatível com contratos inteligentes.

Manolache, Manolache e Tapus (MANOLACHE; MANOLACHE; TAPUS, 2022) não tratam especificamente de eleições políticas, mas propõem um sistema de tomada de decisão baseado em *blockchain* e consenso Proof of Authority. O trabalho é relevante para esta pesquisa por discutir o uso de PoA em ambientes nos quais os participantes responsáveis por validar decisões são conhecidos e identificáveis. A proposta combina *blockchain*, agentes decisores e lógica fuzzy para ponderar votos de acordo com especialização e experiência. A simulação foi conduzida em uma rede Quorum com contratos inteligentes, e os autores relatam que o PoA apresentou melhor desempenho que PoW no tempo de mineração de blocos, além de menor consumo energético estimado. O estudo reforça a adequação do PoA a contextos permissionados, mas seu objetivo principal é apoiar processos colaborativos de decisão, não implementar um fluxo eleitoral com validação de eleitores, unicidade de voto e propagação P2P de mensagens de consenso.

Em comparação com esses trabalhos, a proposta desenvolvida nesta pesquisa se diferencia por não depender de uma plataforma *blockchain* de propósito geral, como Ethereum, Quorum ou TomoChain. Em vez de implementar a votação como contrato inteligente sobre uma infraestrutura existente, este trabalho constrói um protótipo próprio em Go, com rede *peer-to-peer* baseada em *libp2p*, propagação de mensagens por GossipSub, descoberta de pares por DHT e um núcleo de consenso PoA implementado diretamente na aplicação. Essa decisão permite explicitar, controlar e avaliar mecanismos internos que tendem a ficar encapsulados em plataformas prontas, como seleção determinística de líder, validação de propostas, coleta de aprovações, tratamento de votos pendentes e persistência da cadeia.

Outro diferencial está no foco experimental. Enquanto Christyono, Widjaja e Wicaksana (CHRISTYONO; WIDJAJA; WICAKSANA, 2021) concentram a avaliação na vazão e no tempo de registro de votos em uma rede Ethereum PoA, e Sharma, Krishna e Bahga (SHARMA; KRISHNA; BAHGA, 2021) comparam custos de gás e transações por segundo entre mecanismos de consenso, este trabalho prioriza a validação das propriedades funcionais do protocolo em uma rede permissionada mínima. Assim, a avaliação concentra-se em integridade do resultado, prevenção de duplicidade de voto e continuidade de operação diante de falha por parada de um validador. O objetivo não é superar plataformas maduras em desempenho bruto, mas demonstrar, de forma controlada e auditável, como uma arquitetura distribuída própria pode materializar as propriedades essenciais de um sistema de votação baseado em *blockchain* e PoA.

Desse modo, os trabalhos relacionados confirmam a relevância do uso de *blockchain* e consenso permissionado em cenários de votação e decisão coletiva, mas também evidenciam uma lacuna que foi explorada por este trabalho, isto é, a construção de uma implementação própria e de escopo controlado, na qual os componentes de rede, consenso, validação de votos e persistência são projetados explicitamente para o domínio de votação eletrônica descentralizada.

3 MATERIAIS E MÉTODOS

Este capítulo descreve, de forma sistemática, os recursos tecnológicos empregados e a metodologia adotada para o projeto, implementação e validação do sistema de votação descentralizado proposto. A abordagem aqui apresentada visa explicitar não apenas as ferramentas utilizadas, mas principalmente as decisões arquiteturais, os modelos formais adotados e as estratégias empregadas para tratar problemas clássicos de sistemas distribuídos, como consenso, ordenação de eventos, tolerância a falhas e sincronização de estado.

3.1 Ferramentas

O sistema foi integralmente desenvolvido na linguagem Go (Golang), escolhida por suas características de concorrência nativa, tipagem estática, simplicidade sintática e eficiência na construção de aplicações distribuídas. A própria documentação oficial da linguagem destaca seus mecanismos de concorrência e sua adequação à construção de software para ambientes multicore e aplicações de rede (The Go Authors, 2026a). No protótipo desenvolvido, esse modelo foi explorado por meio de goroutines e canais (*channels*), utilizados para estruturar o processamento assíncrono de eventos e a comunicação interna entre componentes.

Para a camada de comunicação peer-to-peer, foi utilizada a biblioteca *libp2p*, que fornece abstrações modulares para construção de redes descentralizadas (Protocol Labs, 2026a). A disseminação de votos, propostas de blocos e aprovações foi implementada com GossipSub, protocolo de publicação e assinatura adotado no ecossistema libp2p para propagação de mensagens entre pares inscritos em tópicos comuns (Protocol Labs, 2026c). A descoberta de pares e o processo de entrada na rede foram apoiados pela DHT baseada em Kademlia, empregada pela libp2p como estrutura distribuída para localização de nós e dados em redes P2P (Protocol Labs, 2026a). Além disso, o protocolo Rendezvous foi utilizado como mecanismo auxiliar de anúncio e descoberta de pares, permitindo que nós registrem sua presença em um ponto comum e sejam posteriormente encontrados por outros participantes da rede (Protocol Labs, 2026b).

A escolha por GossipSub fundamenta-se em sua adequação a redes nas quais os participantes precisam receber mensagens associadas a tópicos compartilhados sem depender de um servidor central. No contexto deste trabalho, essa característica é relevante porque votos, propostas e aprovações devem ser propagados entre validadores autorizados de forma assíncrona, mantendo a descentralização da comunicação.

No que se refere à segurança criptográfica, foram utilizados recursos da biblioteca padrão `crypto` da própria linguagem Go. O pacote `crypto/sha256` foi empregado para geração de hashes de blocos e identificação única de dados, uma vez que implementa os algoritmos SHA-224 e SHA-256(The Go Authors, 2026c). Para geração de pares de chaves públicas e privadas, bem como para assinatura e verificação de votos, foi utilizado o pacote `crypto/ed25519`, que implementa o algoritmo de assinatura digital Ed25519 (The Go Authors, 2026b). A adoção do Ed25519 decorre de sua eficiência computacional, de sua aplicação consolidada em sistemas distribuídos modernos e da disponibilidade de implementação nativa na biblioteca padrão da linguagem.

O armazenamento persistente dos blocos foi implementado por meio de arquivos binários sequenciais no sistema de arquivos local, organizados por altura de bloco. Um índice auxiliar permite a recuperação eficiente dos registros e a reconstrução determinística do estado após reinicialização do nó.

Por fim, para validação experimental, foi desenvolvido um ambiente de simulação multi-nó, permitindo a execução simultânea de diversos validadores em instâncias independentes e a observação controlada do comportamento do protocolo em uma rede permissionada local.

3.2 Metodologia

A metodologia adotada neste trabalho fundamenta-se em princípios de engenharia de software orientada a domínio e em modelos formais de sistemas distribuídos. O sistema foi concebido como uma máquina de estado replicada, cuja evolução é determinada por um protocolo de consenso permissionado baseado em Proof of Authority.

3.2.1 Abordagem Arquitetural

A arquitetura do sistema foi estruturada segundo o modelo *Ports and Adapters*, também conhecido como Arquitetura Hexagonal, proposto por Cockburn (COCKBURN, 2005). Esse modelo favorece a separação entre o núcleo da aplicação e os mecanismos externos de interação, como interfaces, armazenamento e comunicação. No contexto deste trabalho, essa separação permitiu concentrar regras de negócio e lógica de consenso no domínio, enquanto detalhes de rede, persistência e criptografia permaneceram na camada de infraestrutura.

Além disso, adotou-se uma organização modular inspirada em Domain-Driven Design (DDD), abordagem proposta por Evans para lidar com a complexidade de software a partir da modelagem explícita do domínio (EVANS, 2003). Assim, o sistema foi segmentado em contextos bem definidos: *blockchain*, *consensus*, *network*, *voting*, *mempool* e *node*. Cada

módulo possui responsabilidades delimitadas, reduzindo acoplamento e facilitando evolução incremental.

Internamente, o processamento foi estruturado segundo um modelo orientado a eventos. Um *event loop* central serializa o tratamento de eventos externos (recebimento de votos, propostas e aprovações) e internos (timeouts e intervalos de bloco), mitigando condições de corrida e simplificando a consistência do estado.

3.2.2 Modelo Formal e Premissas do Consenso

A definição do consenso é tratada, neste capítulo, como parte da metodologia do trabalho. Antes da descrição concreta dos componentes de software, apresentada no Capítulo 4, é necessário estabelecer as premissas, a notação e as propriedades que orientam o desenvolvimento do protótipo. Dessa forma, a modelagem a seguir delimita o comportamento esperado do protocolo e fornece a base para sua posterior implementação e avaliação.

O mecanismo de consenso adotado é o Proof of Authority, apropriado para ambientes permissionados, nos quais os validadores são entidades previamente autorizadas. Em implementações de PoA, uma estratégia comum para organizar a proposição de blocos é a rotação determinística dos validadores. Neste trabalho, adotou-se o escalonamento *round-robin*, por se tratar de uma alternativa simples, previsível e adequada a um conjunto fixo de validadores.

Seja $V = \{v_0, v_1, \ldots, v_{n-1}\}$ o conjunto ordenado de validadores autorizados e seja $h \in \mathbb{N}$ a altura lógica do próximo bloco. Em sistemas *blockchain*, altura é o termo usual para indicar a posição de um bloco na cadeia; de forma equivalente, pode-se interpretar h como o índice do bloco na sequência, considerando o bloco gênesis como a altura zero. O líder $L(h)$ responsável pela proposição na altura h é definido por:

$$L(h) = v_{h \bmod n}$$

onde $n = |V|$ representa o número total de validadores.

Essa estratégia elimina disputas concorrenciais e reduz a sobrecarga de comunicação, uma vez que todos os nós conseguem determinar localmente quem possui autoridade para propor o próximo bloco da cadeia.

A partir disso, o processo de consenso é estruturado como um protocolo determinístico composto pelas seguintes etapas:

1. Determinação do líder para a altura h;
2. Proposição e assinatura criptográfica do bloco pelo líder;

3. Validação independente do bloco pelos demais validadores;
4. Emissão de aprovações e coleta de quórum;
5. Finalização quando atingida maioria simples ($\lfloor n/2 \rfloor + 1$).

A dinâmica do protocolo é formalizada como uma máquina de estados finita

$$\mathcal{M} = (S, s_0, \mathcal{E}, \delta)$$

onde:

- $S = \{Idle, Proposing, Validating\}$ é o conjunto de estados;
- $s_0 = Idle$ é o estado inicial;
- $\mathcal{E}$ representa o conjunto de eventos (timeout, recebimento de proposta, recebimento de aprovação);
- $\delta : S \times \mathcal{E} \to S$ é a função de transição.

As transições principais da máquina de estados são apresentadas na Figura 3. O estado *Idle* representa a espera por um evento de consenso. Quando o nó local é o líder da rodada, ocorre a transição para *Proposing*. Quando uma proposta válida é recebida de outro líder, o nó passa diretamente para *Validating*. Após a disseminação da proposta, o proponente também entra em *Validating*. Atingido o quórum de aprovações, o bloco é finalizado e a máquina retorna ao estado *Idle*.

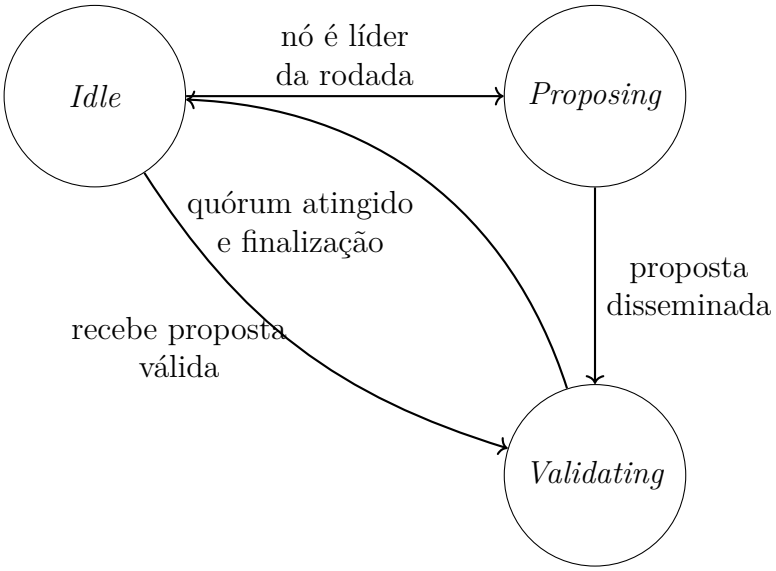

Figura 3 – Máquina de estados do consenso PoA adotado

Fonte: Elaborado pelo autor.

Essa formalização explicita que o sistema evolui por meio de eventos observáveis, preservando coerência entre as fases do protocolo e garantindo que, para cada altura h, exista no máximo um bloco finalizado sob hipótese de maioria honesta.

3.2.3 Estratégias de Tolerância a Falhas

O ambiente distribuído no qual o sistema opera é assíncrono, sujeito a atrasos arbitrários de mensagem, reordenação de pacotes e falhas de processo por parada. Assim, foram incorporadas estratégias explícitas para preservar as propriedades de *safety* (não ocorrência de estados inconsistentes) e *liveness* (progresso contínuo da cadeia), mesmo sob condições adversas.

3.2.3.1 Ordenação Causal

Em um sistema distribuído assíncrono, a rede não garante ordenação global de mensagens. Seja $P(h)$ a proposta de bloco para a altura h e $A_i(h)$ a aprovação emitida pelo validador v_i. Pode ocorrer que, em determinado nó v_j, a ordem de entrega satisfaça:

$$A_i(h) \prec_j P(h),$$

onde $\prec_j$ representa a ordem local de recepção de mensagens no nó v_j.

Para preservar a coerência do protocolo, define-se um buffer temporário indexado por altura:

$$B : \mathbb{N} \to \mathcal{P}(\mathcal{A}),$$

onde $\mathcal{A}$ é o conjunto de todas as aprovações possíveis e $B(h)$ armazena aprovações recebidas cuja proposta correspondente ainda não foi registrada localmente.

O conjunto de aprovações válidas já processadas para a altura h é denotado por:

$$Q(h) \subseteq \mathcal{A}.$$

O critério de finalização permanece definido por:

$$|Q(h)| \geq \left\lfloor \frac{n}{2} \right\rfloor + 1,$$

onde n é o número total de validadores autorizados.

Desse modo, o comportamento formal ao receber uma aprovação $A_i(h)$ pode ser descrito por:

$$\text{onReceive}(A_i(h)) = \begin{cases} Q(h) \leftarrow Q(h) \cup \{A_i(h)\}, & \text{se } P(h) \text{ já conhecida} \\ B(h) \leftarrow B(h) \cup \{A_i(h)\}, & \text{caso contrário} \end{cases}$$

Operacionalmente, o procedimento é descrito no Algoritmo 1.

Algoritmo 1: Tratamento de aprovação sob possível reordenação

Dados: aprovação $A_i(h)$ recebida

Saída: aprovação registrada ou armazenada

```
se proposta P(h) já conhecida então
    validar assinatura de A_i(h);
    inserir A_i(h) em Q(h);
    verificar quórum;
fim
senão
    inserir A_i(h) em B(h);
fim
```

Quando a proposta $P(h)$ é finalmente recebida, as aprovações armazenadas devem ser reintegradas ao fluxo normal de processamento. Formalmente:

$$\text{onReceive}(P(h)) = \begin{cases} \text{registrar } P(h) \\ Q(h) \leftarrow Q(h) \cup B(h) \\ B(h) \leftarrow \varnothing \end{cases}$$

O comportamento correspondente é apresentado no Algoritmo 2.

Algoritmo 2: Processamento de aprovações armazenadas

Dados: proposta $P(h)$ recebida

Saída: aprovações pendentes processadas

```
registrar P(h);
se B(h) ≠ ∅ então
    para cada A_i(h) ∈ B(h) faça
        validar assinatura de A_i(h);
        inserir A_i(h) em Q(h);
    fim
    esvaziar B(h);
fim
verificar quórum;
```

Além do mais, sob a hipótese de maioria honesta e eventual entrega de mensagens, a introdução do buffer $B(h)$ garante:

$$\text{Se } |\{v_i \in V \mid v_i \text{ aprovou } P(h)\}| \geq \left\lfloor \frac{n}{2} \right\rfloor + 1,$$

então o bloco $P(h)$ será eventualmente finalizado, independentemente da ordem de entrega das mensagens.

Dessa forma, a estratégia adotada preserva a propriedade de *safety* ao evitar descarte de aprovações válidas e assegura progresso mesmo sob latências assimétricas e reordenação arbitrária de mensagens.

3.2.3.2 Garantia de Vivacidade

Em sistemas permissionados com líder determinístico, a ausência temporária de transações pode levar à estagnação da cadeia. Entretanto, a proposição imediata e contínua de blocos vazios pode provocar crescimento artificial da blockchain e desperdício de recursos computacionais e de armazenamento.

Para equilibrar esses dois extremos, foi adotado um mecanismo de espera temporizada T_w antes da proposição de um bloco vazio. Seja $M(h)$ o conjunto de votos disponíveis para a altura h. Ao assumir a liderança dessa altura, o nó inicia um temporizador e aguarda a chegada de novas transações por um intervalo limitado. Formalmente, o líder permanece em espera até o instante $\min(\Delta t, T_w)$, onde Δt representa o tempo até a chegada de um novo voto.

Ao término do intervalo de espera, duas situações são possíveis. Se $|M(h)| > 0$, o bloco proposto incorpora os votos acumulados durante o período de observação. Caso contrário, isto é, se $|M(h)| = 0$, é proposto um bloco vazio $B_\varnothing(h)$, garantindo a continuidade do protocolo mesmo na ausência de atividade.

Esse mecanismo assegura a propriedade de vivacidade sob a hipótese de que o líder designado esteja operacional e que as mensagens eventualmente sejam entregues. A vivacidade, nesse contexto, refere-se ao progresso contínuo da cadeia: após uma altura já finalizada, o protocolo deve ser capaz de finalizar alguma altura posterior, ainda que o bloco correspondente não contenha votos. Blocos vazios, portanto, não representam blocos pendentes de rodadas anteriores, mas blocos válidos propostos na própria rodada para evitar estagnação. Em termos formais:

$$\forall h \in \mathbb{N},\ B_h \text{ finalizado} \Rightarrow \exists h' > h \text{ tal que } B_{h'} \text{ é eventualmente finalizado.}$$

Essa formulação indica que a cadeia progride indefinidamente, ainda que períodos de inatividade ocorram. Dessa forma, evita-se tanto a estagnação do sistema quanto a geração excessiva de blocos vazios.

3.2.3.3 Persistência e Recuperação

Para garantir tolerância a falhas do tipo *crash-stop*, a finalização de um bloco é tratada como uma operação logicamente atômica, composta por uma sequência ordenada de etapas que preservam consistência entre estado em memória e armazenamento persistente. Quando um bloco B_h atinge quórum válido, executa-se: (i) a aplicação determinística de suas transações ao estado corrente; (ii) sua inserção na estrutura da cadeia mantida em memória; (iii) sua persistência em formato binário no armazenamento local; e (iv) a remoção das transações correspondentes do mempool.

Denotando por σ_h o estado da aplicação após a finalização do bloco de altura h, define-se σ_0 como o estado inicial após o bloco gênesis, no qual nenhum voto regular foi registrado. Para $h > 0$, a evolução do estado é dada pela relação recursiva

$$\sigma_h = \mathrm{Apply}(\sigma_{h-1}, B_h),$$

onde a função Apply é determinística. Essa propriedade assegura que dois nós honestos que processem a mesma sequência de blocos alcancem estados idênticos.

A persistência do bloco é realizada imediatamente após a atualização do estado, estabelecendo a garantia de que, se B_h é finalizado, então B_h encontra-se armazenado de forma durável.

Em caso de reinicialização do nó, o estado é reconstruído por meio da leitura sequencial dos blocos persistidos e reaplicação determinística de suas transações, isto é,

$$\sigma_h = \mathrm{Apply}(\mathrm{Apply}(\ldots \mathrm{Apply}(\sigma_0, B_1), \ldots), B_h).$$

Esse procedimento assegura que o estado recuperado seja equivalente ao último estado confirmado antes da falha. Sob o modelo *crash-stop*, preserva-se assim a propriedade de segurança segundo a qual nenhum bloco finalizado é perdido após a reinicialização do processo.

Em conjunto com os mecanismos de ordenação causal e de garantia de vivacidade, essa estratégia de persistência determinística contribui para que o sistema mantenha coerência global e progresso contínuo mesmo diante de atrasos de rede e interrupções inesperadas.

3.2.4 Planejamento da Avaliação Experimental

Esta seção descreve o planejamento da avaliação experimental utilizada para verificar o comportamento do sistema. Os resultados obtidos a partir dessa avaliação são apresentados no Capítulo 5. Dessa forma, o Capítulo 3 delimita o desenho experimental,

os cenários e os critérios de observação, enquanto a análise quantitativa e qualitativa dos dados é reservada ao capítulo de resultados.

A avaliação do sistema foi conduzida por meio de um ambiente controlado de simulação multi-nó, projetado para reproduzir, de forma sistemática, o comportamento de uma rede permissionada local com o número mínimo de validadores necessário para observar consenso e tolerância a uma falha por parada. Essa abordagem permitiu avaliar o funcionamento do protocolo sob cenários operacionais delimitados, mantendo controle sobre variáveis como carga de votos e ordem de processamento de mensagens.

O ambiente experimental consistiu na execução simultânea de instâncias independentes de nós validadores, interconectadas por meio da camada peer-to-peer baseada em *libp2p*. A propagação de mensagens ocorreu via protocolo GossipSub, enquanto a descoberta de pares utilizou DHT no modo Kademlia. Clientes votantes independentes foram empregados para submeter votos assinados digitalmente à rede, permitindo exercitar o ciclo completo de criação, propagação, inclusão em bloco e persistência.

Durante os experimentos, foram coletados dados provenientes de logs estruturados de execução e da própria blockchain persistida em disco. Entre as informações registradas encontram-se timestamps de eventos, tamanho do mempool, número de pares conectados, identificação de blocos propostos e finalizados, além de metadados como propositor e altura da cadeia. Esses registros possibilitam posterior análise da consistência do ledger, da formação de blocos e da convergência entre nós.

Para avaliação do sistema, foi definida uma bateria de três testes experimentais, cada um focado em um aspecto específico do protocolo. As subseções seguintes descrevem a motivação, os objetivos e a metodologia de cada teste, estabelecendo a base para a apresentação dos resultados no Capítulo 5.

3.2.4.1 Teste de Validação Funcional

O primeiro teste teve como objetivo validar o funcionamento básico do sistema, exercitando o ciclo completo desde a submissão de votos até sua finalização na blockchain. Esse teste é fundamental para confirmar que a implementação reflete corretamente o modelo formal apresentado nas seções anteriores.

A metodologia consistiu na submissão de um conjunto reduzido de votos a uma rede composta pelo número mínimo de validadores adotado no experimento, permitindo observação detalhada de cada etapa do protocolo. Foram monitorados eventos como recepção de votos, propagação via GossipSub, validação criptográfica, proposição de blocos pelo líder designado, coleta de aprovações e finalização mediante quórum.

O teste foi projetado para verificar três propriedades fundamentais: (i) determinismo da seleção de líder segundo o mecanismo *round-robin* definido por $L(h) = v_{h \bmod n}$; (ii)

corretude do quórum, garantindo que blocos sejam finalizados apenas após obtenção de maioria simples de aprovações; e (iii) consistência da blockchain entre validadores, assegurando que todos os nós honestos convergem para o mesmo estado.

A coleta de dados incluiu análise dos logs de execução para identificar o líder de cada bloco, contagem de aprovações recebidas antes da finalização e comparação dos hashes de blocos entre os três validadores. Essas informações permitem validar empiricamente as garantias formais de determinismo e consistência estabelecidas no modelo teórico.

3.2.4.2 Teste de Unicidade de Voto

O segundo teste foi projetado para avaliar os mecanismos de prevenção de votação dupla, propriedade fundamental para a integridade de qualquer sistema de votação eletrônica. A unicidade de voto, expressa formalmente pela restrição de que cada eleitor e_i pode submeter no máximo um voto válido, deve ser garantida tanto no nível de validação individual quanto no nível de consenso distribuído.

A metodologia adotada consistiu na submissão deliberada de dois votos do mesmo eleitor, com intervalo temporal suficiente para que o primeiro voto fosse completamente finalizado na blockchain antes da tentativa de submissão do segundo. Esse cenário permite avaliar se o sistema detecta corretamente a duplicação e impede a inclusão do segundo voto em qualquer bloco subsequente.

O teste exercita múltiplas camadas de validação. Primeiramente, verifica-se se o mempool rejeita votos duplicados antes mesmo de sua propagação. Em segundo lugar, avalia-se se os validadores, ao receberem um voto duplicado via rede P2P, executam verificação contra o estado consolidado da blockchain. Finalmente, confirma-se que, mesmo que um voto duplicado seja inadvertidamente incluído em uma proposta de bloco, os demais validadores rejeitam essa proposta durante a fase de validação.

A coleta de dados incluiu análise dos logs de todos os validadores para identificar mensagens de erro relacionadas à detecção de duplicação, contagem de votos finalizados na blockchain para o eleitor em questão e verificação da integridade do estado global. Espera-se que, independentemente da ordem de processamento ou de possíveis reordenações de mensagens na rede, apenas um voto do eleitor seja persistido na blockchain, preservando a propriedade de unicidade.

3.2.4.3 Teste de Tolerância a Falhas

O terceiro teste teve como objetivo verificar a capacidade do sistema de manter operação contínua após falha de um validador, propriedade essencial para aplicações que exigem alta disponibilidade. A tolerância a falhas é uma das principais motivações para

adoção de arquiteturas distribuídas, e sua validação empírica é fundamental para confirmar as garantias teóricas estabelecidas.

A metodologia foi estruturada em três fases distintas. Na primeira fase, o sistema opera em condições normais, com todos os validadores ativos, estabelecendo uma linha de base funcional. Na segunda fase, uma falha do tipo *crash-stop* é injetada deliberadamente em um dos validadores, simulando cenário de falha catastrófica sem possibilidade de finalização ordenada. Na terceira fase, o sistema continua operando com número reduzido de validadores, permitindo avaliar se a disponibilidade é mantida e se os votos submetidos após a falha continuam sendo finalizados.

O tipo de falha escolhido, *crash-stop*, corresponde ao modelo no qual um processo simplesmente para de responder, sem enviar mensagens inconsistentes ou contraditórias. Esse modelo é apropriado para falhas de hardware, quedas de energia ou terminações abruptas de processo. Falhas bizantinas, nas quais um validador malicioso envia mensagens arbitrárias, não foram consideradas neste trabalho.

A análise teórica prevê que, para um sistema com n validadores e consenso baseado em maioria simples, a tolerância a falhas é dada por $f = \lfloor (n-1)/2 \rfloor$. Com $n = 3$ validadores, espera-se tolerância de até $f = 1$ falha, mantendo maioria de $\lceil n/2 \rceil = 2$ validadores. O teste empírico visa confirmar essa previsão teórica.

A coleta de dados incluiu medição da disponibilidade, definida como a razão entre votos finalizados e votos submetidos durante todo o período de teste, incluindo as fases antes e depois da falha. Os logs também foram utilizados para verificar continuidade de consenso, finalização de blocos e consistência da cadeia entre os validadores remanescentes. O objetivo desse teste não foi caracterizar vazão ou latência sob carga, mas confirmar a continuidade operacional do protótipo no cenário de falha definido.

3.2.4.4 Considerações sobre a Estratégia de Validação

O ambiente de testes foi executado predominantemente em infraestrutura local, sem introdução artificial de latência ou perda de pacotes. Essa configuração permitiu isolar o comportamento do protocolo de consenso e da lógica de aplicação, reduzindo interferências externas e favorecendo a observação do funcionamento interno do sistema. Embora essa escolha simplifique a análise, é importante reconhecer que ambientes de produção apresentam latências de rede variáveis e possibilidade de perda de pacotes, fatores que podem influenciar o desempenho observado.

A estratégia adotada, portanto, concentrou-se na construção de um ambiente reproduzível e instrumentado, capaz de registrar evidências empíricas do comportamento do protocolo. Cada teste foi executado de forma automatizada através de scripts desenvolvidos especificamente para esse propósito, garantindo reprodutibilidade e reduzindo possibilidade

de erro humano. Os logs de execução foram preservados integralmente, permitindo análise detalhada posterior e auditoria dos resultados.

Os resultados detalhados desses três testes experimentais, incluindo métricas quantitativas, análises comparativas e discussão de limitações identificadas, são apresentados no Capítulo 5.

3.2.5 Uso de IA Generativa

Durante o desenvolvimento e a redação deste trabalho, foi utilizada a ferramenta de IA generativa Codex, da OpenAI, como apoio ao processo de organização, revisão e refinamento. No desenvolvimento do protótipo, a ferramenta auxiliou na organização de especificações, no planejamento de tarefas, na revisão de decisões de implementação e na análise de trechos do código-fonte. Na redação do texto, foi empregada para apoiar revisão textual e melhoria didática de trechos.

A ferramenta não foi utilizada como autora, não substituiu a análise crítica do autor e não gerou os resultados experimentais reportados. O autor assume integral responsabilidade pelo conteúdo, pelas decisões metodológicas, pela implementação do protótipo, pela interpretação dos resultados e pela versão final deste trabalho.

4 ARQUITETURA E IMPLEMENTAÇÃO DO SISTEMA

Este capítulo apresenta a arquitetura e os principais aspectos de implementação do sistema de votação descentralizado desenvolvido neste trabalho. Enquanto o Capítulo 3 definiu as ferramentas, a metodologia e o modelo formal do protocolo, a discussão a seguir descreve como esses elementos foram materializados em componentes de software, estruturas de dados e fluxos operacionais executados por cada nó validador.

A exposição foi organizada de modo a partir da visão geral da arquitetura e avançar gradualmente para o funcionamento interno do nó, a representação de votos e blocos e o ciclo completo de processamento de um voto em uma rede com múltiplos validadores. Com isso, busca-se evidenciar a relação entre as decisões de implementação e as propriedades esperadas do sistema.

4.1 Visão Arquitetural do Sistema

A arquitetura do sistema foi concebida com o objetivo central de transformar o modelo formal apresentado no Capítulo 3 em uma implementação concreta, verificável e modular. Enquanto o capítulo anterior definiu propriedades como determinismo, quórum majoritário e encadeamento criptográfico, a presente seção demonstra como tais propriedades são refletidas na organização estrutural do software.

O princípio orientador da arquitetura foi a separação rigorosa entre **núcleo determinístico** e **infraestrutura distribuída**. As garantias de segurança e consistência do sistema não podem depender de detalhes de rede, latência ou mecanismos de armazenamento. Assim, todas as regras de consenso, validação e transição de estado foram concentradas em um núcleo lógico independente da camada de comunicação.

Cada nó da rede executa integralmente o protocolo. Não há entidade coordenadora externa, nem servidor central responsável por decisões globais. Essa autonomia garante que qualquer validador autorizado mantenha uma visão completa e consistente da cadeia, reforçando a natureza distribuída do sistema.

A arquitetura interna do nó foi organizada em seis módulos principais, com responsabilidades delimitadas e complementares. O **módulo de consenso** implementa a máquina de estados responsável por coordenar proposições, validações e finalizações de blocos, constituindo o núcleo lógico no qual as decisões determinísticas são tomadas. O **gerenciador da blockchain**, por sua vez, mantém a estrutura encadeada de blocos,

realiza verificações de integridade e controla a altura corrente da cadeia.

As regras diretamente associadas ao domínio eleitoral ficam concentradas no **módulo de votação**, que define as estruturas de eleição e voto e aplica as verificações de elegibilidade e unicidade. Entre a recepção dos votos e sua consolidação em blocos, o **mempool** atua como uma área temporária para votos válidos ainda não incluídos na cadeia. Já a **camada de rede P2P** é responsável pela propagação assíncrona de votos, propostas e aprovações entre validadores, enquanto a **camada de persistência** realiza a serialização determinística e o armazenamento durável dos blocos finalizados.

A interação entre esses módulos é estritamente orientada a eventos. Mensagens recebidas da rede são encaminhadas ao núcleo de consenso apenas após validação estrutural mínima, evitando que dados malformados influenciem a lógica determinística do sistema. De maneira simétrica, decisões internas do consenso, como a proposição de um novo bloco, são convertidas em mensagens pela camada de rede e propagadas aos demais nós.

Essa organização permite que o comportamento global do sistema emerja da composição de componentes bem definidos, e não de interações implícitas ou acoplamentos ocultos. O resultado é uma arquitetura previsível e alinhada às propriedades formais previamente estabelecidas.

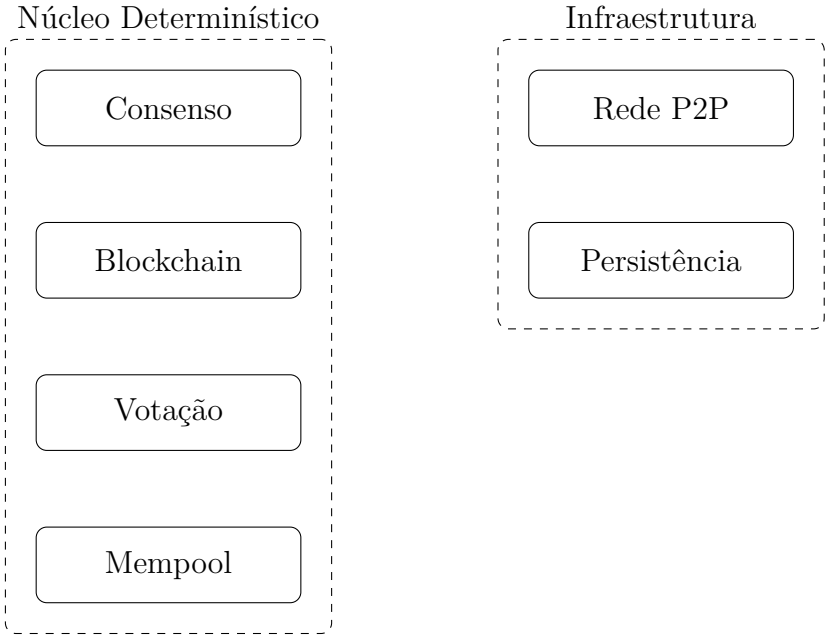

Figura 4 – Arquitetura interna de um nó validador

Fonte: Elaborado pelo autor.

Assim, a arquitetura proposta não apenas organiza o código de forma modular, mas materializa diretamente as garantias formais do protocolo. O isolamento do núcleo determinístico assegura consistência entre validadores honestos, enquanto a camada de infraestrutura permite operação em ambiente distribuído sujeito a latências e falhas por

parada. Essa combinação estabelece a base sobre a qual o restante da implementação será detalhado nas seções seguintes.

4.2 Modelo Operacional do Nó Validador

O comportamento global de cada nó validador emerge da execução contínua de um modelo operacional orientado a eventos. Diferentemente de arquiteturas baseadas em chamadas síncronas encadeadas, o sistema foi concebido como um laço de processamento determinístico, no qual entradas externas, como mensagens de rede, e eventos internos, como o disparo do temporizador de bloco, são processados sequencialmente pelo núcleo de consenso. Essa decisão arquitetural permite serializar logicamente as decisões: ainda que a rede seja assíncrona, as transições de estado do consenso ocorrem de maneira ordenada e controlada. Como consequência, preserva-se o determinismo interno do nó, pois dois validadores honestos submetidos à mesma sequência ordenada de eventos produzem o mesmo estado final.

4.2.1 Modelo de Eventos

O comportamento do nó é estruturado como um sistema orientado a eventos discretos. Cada mensagem recebida ou ação relevante do ambiente é convertida em um evento interno pertencente ao conjunto

$$\mathcal{E} = \{E_{\text{vote}}, E_{\text{proposal}}, E_{\text{approval}}, E_{\text{sync}}, E_{\text{interval}}\}.$$

O evento E_{vote} representa a recepção de um voto, seja ele submetido localmente ou propagado pela rede. O evento E_{proposal} corresponde à chegada de uma proposta de bloco emitida pelo líder da rodada, enquanto E_{approval} representa uma aprovação de bloco recebida de outro validador. O evento E_{sync} é utilizado quando o nó detecta discrepâncias de altura e precisa iniciar uma sincronização de estado. Por fim, E_{interval} é gerado internamente pelo temporizador periódico de bloco e inicia uma tentativa de proposição quando o nó local é o líder da rodada corrente.

Na altura h, o estado global do nó é representado por

$$\Omega_h = (\sigma_h, M_h, C_h),$$

onde σ_h corresponde ao estado determinístico resultante da aplicação sequencial dos blocos confirmados até h, M_h denota o conjunto de votos pendentes mantidos no mempool e C_h representa o estado interno da máquina de consenso.

A evolução do sistema é descrita por uma função de transição determinística

$$\Omega'_h = \delta(\Omega_h, e), \quad e \in \mathcal{E},$$

na qual δ incorpora as validações criptográficas, verificações de elegibilidade, atualização de quórum e possíveis efeitos colaterais controlados (broadcast e persistência). Dessa forma, a execução do nó pode ser interpretada como a aplicação iterativa de δ a uma sequência ordenada de eventos provenientes da rede ou do próprio sistema.

Essa modelagem garante que, para uma mesma sequência de eventos processados na mesma ordem, dois nós honestos produzirão estados finais idênticos. O determinismo da função δ constitui, portanto, o elo fundamental entre a natureza assíncrona da rede e a consistência global da cadeia.

No nível operacional, os eventos são consumidos a partir de uma fila interna e tratados sequencialmente pelo núcleo de consenso. Cada tipo de evento aciona uma transição específica. Votos recebidos são verificados antes de ingressarem no mempool. Propostas de bloco são avaliadas quanto à estrutura, liderança e validade dos votos que contêm. Aprovações atualizam o estado de quórum, e o temporizador periódico introduz, no mesmo fluxo, a possibilidade de proposição pelo líder da rodada. Ao final de cada transição, o nó verifica se há aprovações suficientes para aplicar o bloco, persistir a nova unidade da cadeia e iniciar a próxima altura.

Esse fluxo é apresentado no Algoritmo 3.

Algoritmo 3: Processamento determinístico de eventos no nó validador

Dados: estado inicial $\Omega_0 = (\sigma_0, M_0, C_0)$
Saída: execução contínua do nó

enquanto *nó ativo* **faça**
  receber evento $e \in \mathcal{E}$ da fila de eventos;
  e
  caso E_{vote} **faça**
    validar assinatura e elegibilidade;
    inserir voto em M_h;
    propagar voto se origem local;
  fim
  caso $E_{proposal}$ **faça**
    validar proposta de bloco;
    validar votos contidos no bloco;
    registrar aprovação local;
    propagar aprovação;
  fim
  caso $E_{approval}$ **faça**
    registrar aprovação recebida;
  fim
  caso $E_{interval}$ **faça**
    se *nó é líder da rodada* **então**
      construir bloco a partir de M_h;
      validar proposta localmente;
      registrar auto-aprovação;
      propagar proposta e aprovação;
    fim
  fim
  caso E_{sync} **faça**
    iniciar procedimento de sincronização;
  fim
  se *condição de quórum satisfeita* **então**
    $\sigma_{h+1} \leftarrow \text{Apply}(\sigma_h, B_h)$;
    persistir B_h em armazenamento durável;
    limpar votos correspondentes de M_h;
    reiniciar rodada de consenso;
    incrementar altura $h \leftarrow h + 1$;
  fim
fim

Além disso, a implementação adota fila interna *thread-safe* para entrada de eventos, enquanto o processamento do consenso ocorre em fluxo lógico único. Essa escolha elimina condições de corrida na atualização do estado global e simplifica a verificação de corretude.

O temporizador de bloco não atua como interrupção externa arbitrária, mas como gerador explícito do evento E_{interval}, preservando a uniformidade do modelo de transição.

Desse modo, o loop principal do nó constitui o mecanismo operacional que conecta o ambiente distribuído ao núcleo formal do protocolo. Ele transforma entradas assíncronas em transições determinísticas, preservando simultaneamente segurança, coerência e progresso da cadeia.

4.3 Estrutura de Blocos e Votos

A estrutura interna de votos e blocos constitui o núcleo material do sistema, pois é sobre essas entidades que incidem as garantias de autenticidade, integridade e imutabilidade descritas nos capítulos anteriores. A modelagem dessas estruturas foi realizada de modo a assegurar verificabilidade criptográfica independente, determinismo de representação e encadeamento resistente a modificações retroativas.

4.3.1 Estrutura do Voto

O voto representa a unidade básica de informação persistida na blockchain. Cada voto deve conter dados suficientes para permitir validação autônoma por qualquer validador da rede. Uma representação estrutural simplificada é dada por:

Listing 4.1 – Estrutura simplificada do voto

```
type Vote struct {
    VoterID   string
    Choice    string
    Timestamp int64
    PubKey    []byte
    Signature []byte
}
```

O campo `VoterID` identifica unicamente o eleitor dentro da eleição, enquanto `Choice` registra a opção selecionada. O campo `Timestamp` armazena o instante lógico de criação do voto e auxilia na rastreabilidade do evento sem determinar, por si só, a validade da escolha. A chave pública do eleitor é mantida em `PubKey`, permitindo que qualquer validador verifique a autoria da mensagem, e `Signature` contém a assinatura digital produzida sobre o conteúdo do voto.

A assinatura é calculada sobre a representação canônica do voto, excluindo o próprio campo `Signature`. Formalmente,

$$\sigma = \mathrm{Sign}_{sk}(H(\mathcal{S}(v))),$$

onde sk é a chave privada do eleitor, $\mathcal{S}$ é a função de serialização determinística e H é uma função hash criptográfica resistente a colisões. A verificação ocorre por meio de:

$$\mathrm{Verify}_{pk}(\sigma, H(\mathcal{S}(v))) = \mathrm{true}.$$

A igualdade acima indica que a assinatura σ foi produzida pela chave privada correspondente à chave pública pk, e que o conteúdo do voto não sofreu qualquer modificação após sua assinatura. Dessa forma, o mecanismo garante simultaneamente autenticidade, ao vincular o voto ao seu emissor legítimo, e integridade, ao assegurar que qualquer alteração no conteúdo invalide a verificação criptográfica. Como apenas o detentor da chave privada pode gerar uma assinatura válida, obtém-se também a propriedade de não-repúdio.

4.3.2 Estrutura do Bloco

O bloco é a unidade de consenso da cadeia e agrega um conjunto ordenado de votos validados. Sua estrutura pode ser representada como:

Listing 4.2 – Estrutura simplificada do bloco

```
type Block struct {
    Height       uint64
    PrevHash     []byte
    Timestamp    int64
    Votes        []Vote
    ProposerKey  []byte
    Hash         []byte
}
```

No bloco, `Height` indica sua posição na cadeia, ao passo que `PrevHash` armazena o hash criptográfico do bloco anterior e estabelece o vínculo com o histórico já confirmado. O campo `Votes` contém o conjunto ordenado de votos incluídos na unidade de consenso. A chave pública do líder responsável pela proposição é registrada em `ProposerKey`, e o campo `Hash` funciona como identificador criptográfico do próprio bloco.

O encadeamento entre blocos é definido pela relação:

$$\texttt{PrevHash}_h = H(\mathcal{S}(B_{h-1})),$$

garantindo que qualquer modificação em B_{h-1} invalide todos os blocos subsequentes.

4.3.3 Serialização Determinística

Para preservar consistência entre validadores, a serialização das estruturas deve ser determinística. Define-se:

$$\mathcal{S} : \text{Block} \to \{0,1\}^*$$

como uma função que produz uma representação binária canônica. Essa representação preserva uma ordem fixa dos campos, impõe uma ordenação determinística aos votos contidos no bloco, elimina campos implícitos dependentes do ambiente de execução e utiliza uma codificação binária estável entre plataformas. Com isso, a representação de um mesmo bloco não varia em função do nó que realiza a serialização.

O hash do bloco é então calculado por:

$$\texttt{Hash}_h = H(\mathcal{S}(B_h)).$$

O determinismo da serialização implica:

$$B_h^{(1)} = B_h^{(2)} \Rightarrow H(B_h^{(1)}) = H(B_h^{(2)}),$$

assegurando que validadores honestos sempre obtenham o mesmo identificador criptográfico para blocos semanticamente idênticos.

4.3.4 Integridade Criptográfica e Imutabilidade

A integridade do sistema decorre da composição entre assinaturas digitais individuais, hash criptográfico do bloco e encadeamento por referência hash entre blocos consecutivos. As assinaturas protegem a autenticidade e a integridade de cada voto isoladamente. O hash do bloco resume criptograficamente o conjunto de dados aprovado pelo consenso, e a referência ao bloco anterior propaga essa garantia para toda a cadeia.

Se um voto dentro de B_h for alterado, sua serialização muda, resultando em:

$$H(\mathcal{S}(B_h)) \neq H(\mathcal{S}'(B_h)).$$

Consequentemente, o campo `PrevHash` de B_{h+1} torna-se inconsistente, propagando a invalidação para toda a cadeia subsequente.

Essa propriedade estabelece a imutabilidade prática da blockchain sob a hipótese de que a função hash seja resistente a colisões e que o quórum de consenso seja mantido por validadores honestos.

Portanto, a modelagem adotada assegura que blocos e votos não sejam meros recipientes de dados, mas estruturas criptograficamente vinculadas. A serialização determinística garante convergência entre nós, enquanto o encadeamento hash e as assinaturas digitais fornecem mecanismos formais de verificação e detecção de adulteração.

4.4 Exemplo Completo do Ciclo de um Voto em Rede com Três Validadores

Para consolidar a compreensão do funcionamento do protocolo, apresenta-se nesta seção um exemplo completo do ciclo de vida de um voto em uma rede composta por três validadores autorizados: V_1, V_2 e V_3. Assume-se que o mecanismo de liderança determinística define V_1 como líder da rodada na altura h.

Como $n = 3$, o quórum necessário para finalização é dado por:

$$q = \left\lfloor \frac{n}{2} \right\rfloor + 1 = 2.$$

Assim, são necessárias pelo menos duas aprovações para que um bloco seja finalizado.

4.4.1 Submissão e Propagação do Voto

No protótipo implementado, o eleitor não atua como nó persistente da rede de consenso. A submissão é realizada por um cliente temporário, responsável por criar e assinar o voto, conectar-se a um validador conhecido e publicar a mensagem no tópico de votos da rede P2P. A partir desse ponto, a validação, a proposição de blocos, a coleta de aprovações e a finalização ocorrem de forma distribuída entre os validadores. Assim, a entrada inicial do voto segue um padrão cliente-para-validador, enquanto a decisão sobre sua inclusão na blockchain permanece condicionada ao consenso distribuído.

No exemplo considerado, o voto é inicialmente recebido pelo nó V_2. Esse evento gera E_{vote} e aciona o procedimento local de validação. Primeiro, a assinatura digital é verificada para confirmar a autoria e a integridade do conteúdo. Em seguida, o nó consulta as regras de elegibilidade para assegurar que o eleitor está autorizado a participar da eleição. Por fim, é feita a checagem de não-duplicidade tanto no estado já consolidado

quanto no mempool local, evitando que um mesmo eleitor tenha mais de um voto aceito no fluxo de consenso.

Satisfeitas as pré-condições, o voto é inserido no mempool $M_h^{(2)}$ e propagado à rede. Após a propagação, os nós V_1 e V_3 recebem o voto, executam as mesmas validações determinísticas e o adicionam a seus respectivos mempools. Nesse ponto, temos $M_h^{(1)} = M_h^{(2)} = M_h^{(3)} = \{v\}$.

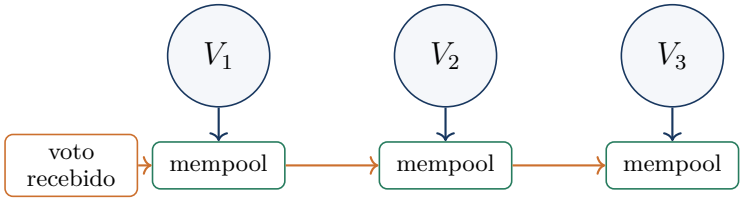

Figura 5 – Submissão e propagação do voto entre validadores

Fonte: Elaborado pelo autor.

4.4.2 Proposição do Bloco pelo Líder

Ao disparo do evento E_{interval}, o nó V_1, identificado como líder da rodada, executa:

$$B_h = \text{ProposeBlock}(M_h^{(1)}, \sigma_h).$$

O bloco B_h contém o voto v e referencia o hash do bloco anterior B_{h-1}. Após construir e assinar a proposta, V_1 a submete ao mesmo caminho de validação aplicado aos blocos recebidos pela rede. Essa validação local evita que o líder avance com uma proposta incompatível com as regras do próprio protocolo. Uma vez aceita, o nó registra sua auto-aprovação e propaga a proposta aos demais validadores.

O conjunto de aprovações nesse momento é:

$$\mathcal{A}(B_h) = \{V_1\}.$$

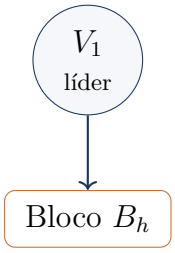

Figura 6 – Proposição do bloco pelo líder da rodada

Fonte: Elaborado pelo autor.

4.4.3 Validação pelos Demais Validadores

Os nós V_2 e V_3 recebem o evento E_{proposal} contendo B_h e verificam, inicialmente, se o bloco foi proposto pelo líder esperado para a altura corrente. Em seguida, cada validador avalia a estrutura do bloco, incluindo altura, referência ao bloco anterior e representação serializada. Os votos contidos na proposta são então validados individualmente, com as mesmas regras de assinatura, elegibilidade e unicidade aplicadas no momento da recepção. Apenas quando essas verificações são satisfeitas o nó registra sua aprovação local e propaga essa aprovação à rede.

Após a aprovação de V_2, o conjunto torna-se:

$$\mathcal{A}(B_h) = \{V_1, V_2\}.$$

Como $|\mathcal{A}(B_h)| = 2 \geq q$, a condição de finalização é satisfeita.

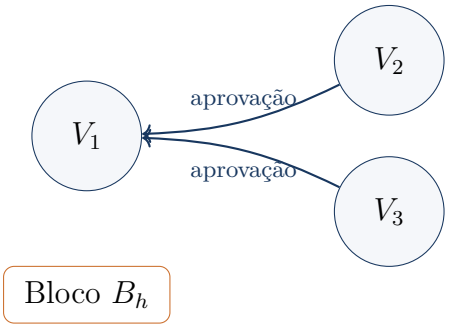

Figura 7 – Validação da proposta e coleta de aprovações

Fonte: Elaborado pelo autor.

4.4.4 Finalização do Bloco

Ao detectar que o quórum foi atingido, o nó executa:

$$\sigma_{h+1} = \text{Apply}(\sigma_h, B_h).$$

Em seguida, o bloco é persistido em armazenamento durável e passa a compor a cadeia local do validador. O voto v é removido do mempool, pois já foi incorporado ao estado confirmado, a altura da cadeia é incrementada e a rodada de consenso é reiniciada para a próxima posição da blockchain.

Mesmo que V_3 ainda não tenha processado todas as aprovações no mesmo instante, a propriedade de entrega eventual garante que todos os nós honestos convergirão para o mesmo estado σ_{h+1}.

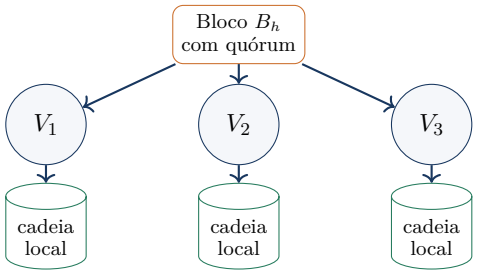

Figura 8 – Finalização local do bloco pelos validadores

Fonte: Elaborado pelo autor.

Observa-se que, após a submissão do voto em V_2, ocorre sua propagação para os demais validadores. O líder V_1 propõe o bloco B_h, que é validado por V_2 e V_3. Ao atingir o quórum mínimo de duas aprovações, a condição de finalização é satisfeita, levando à aplicação determinística do bloco e à atualização consistente do estado global em todos os nós honestos.

5 RESULTADOS

Este capítulo apresenta os resultados obtidos através da validação experimental do sistema proposto. Conforme descrito no Capítulo 3, a estratégia de validação foi conduzida por meio de um ambiente controlado de simulação multi-nó, projetado para reproduzir o comportamento de uma rede permissionada local com o número mínimo de validadores necessário para observar consenso e tolerância a uma falha por parada. Os testes foram estruturados de forma a exercitar as propriedades formais estabelecidas nos capítulos anteriores, incluindo determinismo da máquina de estados, garantia de maioria para finalização e tolerância a falhas do tipo *crash-stop*.

Foram realizados três experimentos distintos, cada um focado em uma das questões investigativas do trabalho: validação funcional, unicidade de voto e tolerância a falhas. Os resultados aqui apresentados fornecem evidências empíricas do comportamento do protocolo e permitem avaliar a viabilidade técnica da abordagem proposta no escopo experimental definido.

5.1 Ambiente de Testes

O ambiente experimental foi configurado para simular uma rede peer-to-peer local composta por três nós validadores independentes. Cada nó executou uma instância completa do sistema, incluindo o núcleo de consenso, a camada de rede P2P baseada em *libp2p* e o mecanismo de persistência em disco. A comunicação entre validadores ocorreu por meio do protocolo GossipSub, enquanto a descoberta de pares utilizou DHT no modo Kademlia.

A configuração de hardware e software utilizada é apresentada na Tabela 2. O sistema operacional escolhido foi macOS, executando sobre arquitetura arm64. A linguagem de implementação foi Go versão 1.21 ou superior, selecionada por suas características de concorrência nativa e eficiência na construção de aplicações distribuídas. A biblioteca *libp2p* forneceu as abstrações necessárias para construção da camada de rede peer-to-peer, incluindo o protocolo GossipSub para propagação de mensagens e DHT no modo Kademlia para descoberta de pares.

Tabela 2 – Configuração do ambiente de testes

Componente	**Especificação**
Sistema Operacional	macOS (darwin)
Arquitetura	arm64
Linguagem	Go 1.21+
Biblioteca P2P	libp2p
Protocolo de Propagação	GossipSub
Descoberta de Pares	DHT (Kademlia)
Consenso	PoA (Proof of Authority)
Número de Validadores	3
Topologia	Rede local (127.0.0.1)

Fonte: Elaborado pelo autor.

Todos os testes foram executados em ambiente local, sem introdução artificial de latência ou perda de pacotes. Essa configuração permitiu isolar o comportamento do protocolo de consenso e da lógica de aplicação, reduzindo interferências externas e favorecendo a observação do funcionamento interno do sistema. Embora ambientes de produção apresentem latências de rede variáveis e possibilidade de perda de pacotes, a escolha por ambiente controlado justifica-se pela necessidade de estabelecer uma linha de base de comportamento, livre de variáveis externas que dificultariam a interpretação dos resultados.

5.2 Metodologia de Avaliação

Para validar o sistema proposto, foram definidos três testes experimentais cobrindo aspectos funcionais, unicidade de voto e tolerância a falhas. Cada teste foi executado de forma automatizada através de scripts desenvolvidos especificamente para esse propósito, com resultados coletados em arquivos de log estruturados para análise posterior. A automação dos testes garante reprodutibilidade e reduz a possibilidade de erro humano durante a execução.

A Tabela 3 apresenta um resumo dos testes realizados, as questões associadas e as principais métricas avaliadas. O primeiro teste focou na validação funcional básica, verificando se o ciclo completo de submissão, propagação, consenso e finalização opera corretamente. O segundo teste avaliou a propriedade de unicidade de voto, especificamente a prevenção de votação dupla. O terceiro teste verificou a tolerância a falhas mediante simulação de crash de um validador.

Tabela 3 – Resumo dos testes experimentais realizados

Teste	**Questão associada**	**Métrica Principal**
Teste 1	Q1: finalização no cenário experimental	Taxa de finalização
Teste 2	Q2: rejeição de voto duplicado	Detecção de duplicação
Teste 3	Q3: finalização após falha por parada	Disponibilidade sob falha

Fonte: Elaborado pelo autor.

Os critérios de sucesso foram estabelecidos com base nas propriedades formais descritas no Capítulo 3. Para o teste funcional, esperava-se que todos os votos submetidos fossem finalizados na blockchain, confirmando a corretude do protocolo no cenário observado. Para o teste de unicidade, o sistema deveria detectar e bloquear tentativas de votação dupla, preservando no máximo um voto finalizado por eleitor. Para o teste de tolerância a falhas, o sistema deveria manter operação contínua mesmo com a falha de um validador, demonstrando resiliência dentro do modelo *crash-stop* adotado.

Cada teste foi executado em uma instância limpa do sistema, com dados de execuções anteriores removidos para evitar interferências. Os logs de execução foram preservados integralmente para análise detalhada, incluindo timestamps de eventos, tamanho do mempool, número de pares conectados e metadados dos blocos finalizados. Essa instrumentação completa permite não apenas verificar o resultado final, mas também compreender o comportamento interno do sistema durante a execução.

5.3 Teste de Validação Funcional

O primeiro teste teve como objetivo validar o funcionamento básico do sistema, exercitando o ciclo completo desde a submissão de votos até sua finalização na blockchain. Foram submetidos cinco votos a uma rede composta por três validadores, com tempo de estabilização de 15 segundos e período de processamento de 30 segundos. Esse conjunto reduzido de votos permite observação detalhada de cada etapa do protocolo, facilitando a identificação de possíveis anomalias ou comportamentos inesperados.

5.3.1 Configuração do Teste

A Tabela 4 apresenta os parâmetros utilizados no teste de validação funcional. O número de validadores foi fixado em três, valor mínimo necessário para demonstrar tolerância a falhas em consenso baseado em maioria simples. O tempo de estabilização de 15 segundos foi estabelecido para garantir que todos os nós completassem o processo de descoberta de pares e estabelecessem conexões P2P antes do início da submissão de votos.

O período de processamento de 30 segundos foi dimensionado para permitir múltiplos ciclos de consenso, assegurando que todos os votos tivessem oportunidade de serem incluídos em blocos e finalizados. Esses valores foram escolhidos para favorecer a observação controlada do ciclo de consenso, e não para caracterizar a vazão máxima do sistema.

Tabela 4 – Configuração do teste de validação funcional

Parâmetro	**Valor**
Número de validadores	3
Votos submetidos	5
Tempo de estabilização	15s
Tempo de processamento	30s

Fonte: Elaborado pelo autor.

5.3.2 Resultados Obtidos

Os resultados do teste demonstraram funcionamento correto de todos os componentes do sistema. A Tabela 5 apresenta as métricas coletadas durante a execução. Todos os cinco votos submetidos foram corretamente propagados pela rede P2P, validados pelos três validadores e finalizados na blockchain. A taxa de sucesso de 100% confirma que o sistema opera conforme especificado, sem perda de transações durante o processo de consenso.

Tabela 5 – Resultados do teste de validação funcional

Métrica	**Valor**
Votos submetidos	5
Votos no mempool	5
Blocos finalizados	17
Votos finalizados	5
Taxa de sucesso	100%

Fonte: Elaborado pelo autor.

O número de blocos finalizados (17) excede significativamente o número de votos (5), indicando que o sistema continuou operando e produzindo blocos mesmo após o processamento de todos os votos. Esse comportamento é esperado e desejável, pois demonstra que o mecanismo de garantia de vivacidade, descrito no Capítulo 3, opera corretamente. Blocos vazios são propostos periodicamente para manter o progresso da cadeia, evitando estagnação mesmo na ausência de transações pendentes.

5.3.3 Verificação do Consenso Round-Robin

O algoritmo de consenso PoA implementado utiliza seleção determinística de líder baseada em escalonamento *round-robin*. Para verificar o funcionamento correto desse mecanismo, foi realizada análise dos blocos finalizados, identificando o validador

responsável pela proposição de cada bloco. A análise confirmou o comportamento esperado do algoritmo. Considerando que o líder na altura h é determinado por $L(h) = v_{h \bmod n}$, onde $n = 3$ é o número de validadores, observou-se rotação sistemática da liderança entre os três nós.

A Tabela 6 apresenta uma amostra de alturas consecutivas, de modo a evidenciar a alternância entre os validadores. A consistência observada na seleção de líder demonstra que o mecanismo determinístico opera corretamente, eliminando disputas concorrenciais e reduzindo overhead de comunicação, conforme previsto na modelagem formal apresentada no Capítulo 3. A ausência de blocos propostos por validadores não autorizados ou fora da ordem esperada confirma que o protocolo é respeitado por todos os participantes.

Tabela 6 – Verificação do mecanismo round-robin de seleção de líder

Altura do Bloco	**Líder**
1	node2
2	node3
3	node1
4	node2
5	node3
6	node1

Fonte: Elaborado pelo autor.

5.3.4 Análise

O teste de validação funcional foi bem-sucedido, confirmando que o sistema implementa corretamente o protocolo especificado. A propagação P2P via GossipSub mostrou-se efetiva, com todos os votos alcançando os três validadores sem necessidade de retransmissões ou mecanismos de recuperação. O consenso PoA operou conforme esperado, com seleção determinística de líder e finalização mediante quórum de maioria simples. A ausência de perda de votos durante o processo valida a robustez do pipeline de processamento, desde a submissão até a persistência em disco.

A observação de que o sistema continua produzindo blocos após o processamento de todos os votos demonstra que o mecanismo de garantia de vivacidade opera adequadamente. Esse comportamento é fundamental para evitar estagnação da cadeia em períodos de baixa atividade, mantendo o progresso contínuo do sistema conforme especificado nas propriedades formais do Capítulo 3.

5.4 Teste de Unicidade de Voto

O segundo teste teve como objetivo verificar se o sistema previne votação dupla (*double voting*), garantindo que cada eleitor tenha no máximo um voto finalizado. Esse

requisito é fundamental para a integridade de qualquer sistema de votação eletrônica, pois a possibilidade de um eleitor votar múltiplas vezes comprometeria a legitimidade do resultado.

A prevenção de votação dupla é realizada em mais de uma etapa do processamento. No recebimento de um voto, o nó verifica se o eleitor já aparece no estado consolidado da blockchain e se já existe voto pendente do mesmo eleitor em seu mempool local. Além disso, durante a validação de uma proposta de bloco, cada validador confere se os votos incluídos são compatíveis com o estado já finalizado. Dessa forma, o mempool atua como uma barreira local contra duplicações conhecidas, enquanto o estado da blockchain finalizada funciona como referência definitiva para a propriedade de unicidade.

5.4.1 Configuração do Teste

O cenário de teste consistiu na submissão de dois votos do mesmo eleitor, identificado como `voter001`, com intervalo de 15 segundos entre as submissões. O intervalo foi estabelecido para garantir que o primeiro voto fosse completamente finalizado na blockchain antes da tentativa de submissão do segundo voto. Essa escolha metodológica permite avaliar se o sistema detecta duplicação mesmo quando o voto original já foi consolidado, representando o cenário mais desafiador para o mecanismo de validação.

5.4.2 Resultados Obtidos

O teste foi executado em duas fases distintas. Na primeira fase, o voto inicial foi submetido e processado normalmente, seguindo o fluxo padrão de propagação, validação e finalização. A Tabela 7 apresenta o resultado dessa fase, confirmando que o primeiro voto foi aceito e finalizado conforme esperado.

Tabela 7 – Resultado da Fase 1 do teste de unicidade (primeiro voto)

Métrica	**Valor**
Status do voto	Aceito e finalizado
Votos na blockchain	1

Fonte: Elaborado pelo autor.

Na segunda fase, após a finalização do primeiro voto, foi submetido um segundo voto do mesmo eleitor. O sistema detectou a tentativa de votação dupla, conforme evidenciado pelos logs de todos os três validadores. A Tabela 8 apresenta o resultado final, demonstrando que apenas um voto permaneceu registrado na blockchain, preservando a propriedade de unicidade.

Tabela 8 – Resultado da Fase 2 do teste de unicidade (voto duplicado)

Métrica	**Valor**
Status do segundo voto	Rejeitado
Votos na blockchain	1
Detecção pelos validadores	3/3

Fonte: Elaborado pelo autor.

5.4.3 Análise dos Logs de Detecção

A análise dos logs de execução revelou que todos os três validadores detectaram corretamente a tentativa de votação dupla. Os registros de erro foram consistentes entre os nós, indicando funcionamento correto do mecanismo de validação distribuída. Um exemplo representativo da mensagem de erro registrada é apresentado a seguir:

```
ERROR: Event handling failed for VoteReceived:
       voter voter001 has already voted (in blockchain)
```

A presença dessa mensagem nos logs dos três validadores confirma que a verificação de unicidade é executada de forma independente por cada nó, preservando a propriedade de validação distribuída do sistema. Não há dependência de um validador central ou autoridade única para detectar duplicações, o que aumenta a robustez e a confiabilidade do mecanismo de prevenção de voto duplicado.

5.4.4 Análise

O teste demonstrou que o sistema garante a propriedade de unicidade de voto no cenário avaliado. A blockchain manteve apenas um voto do eleitor `voter001`, bloqueando efetivamente a tentativa de votação dupla. Todos os validadores detectaram independentemente a duplicação, evidenciando robustez do mecanismo de validação e ausência de pontos únicos de falha no processo de verificação.

Entretanto, observou-se que o segundo voto foi inicialmente aceito pelo sistema antes da validação completa, sendo rejeitado apenas durante o processamento pelo núcleo de consenso. Embora essa característica não comprometa a integridade da blockchain, uma vez que o voto duplicado não é incluído em nenhum bloco, representa uma oportunidade de otimização. A validação prévia no mempool pode reduzir o processamento de duplicações já conhecidas localmente, mas não substitui a validação contra o estado consolidado da blockchain, especialmente em uma rede distribuída na qual diferentes nós podem observar votos em ordens distintas.

Não obstante essa limitação, o resultado fundamental foi alcançado: a blockchain permaneceu íntegra, contendo exatamente um voto por eleitor, conforme especificado nos

requisitos do sistema. A detecção consistente por todos os validadores demonstra que o mecanismo de unicidade opera corretamente mesmo sob condições adversas, quando um eleitor tenta deliberadamente violar as regras do protocolo.

5.5 Teste de Tolerância a Falhas

O terceiro teste teve como objetivo verificar se o sistema mantém disponibilidade e continua processando votos após falha de um validador. O cenário simulou uma falha do tipo *crash-stop*, na qual um dos três validadores é abruptamente terminado durante a operação, sem possibilidade de finalização ordenada ou notificação aos demais nós.

5.5.1 Configuração do Teste

O teste foi estruturado em três fases distintas: operação normal, injeção de falha e operação sob falha. A Tabela 9 apresenta os parâmetros utilizados. Na primeira fase, o sistema opera com todos os três validadores ativos, estabelecendo uma linha de base de comportamento. Na segunda fase, o Node 2 é abruptamente terminado através do sinal SIGKILL, simulando uma falha catastrófica. Na terceira fase, o sistema continua operando com apenas dois validadores ativos, permitindo avaliar se a disponibilidade é mantida.

Tabela 9 – Configuração do teste de tolerância a falhas

Parâmetro	**Valor**
Número de validadores	3
Validador com falha	Node 2
Tipo de falha	Crash-stop (SIGKILL)
Votos na Fase 1 (normal)	3
Votos na Fase 3 (sob falha)	5

Fonte: Elaborado pelo autor.

5.5.2 Resultados Obtidos

O sistema demonstrou tolerância à falha de um validador conforme o modelo de maioria adotado na implementação. A Tabela 10 apresenta os resultados das três fases do teste. Na Fase 1, com todos os três validadores operacionais, o sistema processou normalmente três votos, finalizando oito blocos. Na Fase 2, o Node 2 foi abruptamente terminado, simulando uma falha catastrófica. Na Fase 3, com apenas dois validadores ativos, o sistema continuou operando normalmente, processando cinco votos adicionais e finalizando dez novos blocos.

Tabela 10 – Resultados do teste de tolerância a falhas

Métrica	**Fase 1**	**Fase 2**	**Fase 3**
Validadores ativos	3/3	-	2/3
Votos submetidos	3	-	5
Votos finalizados	3	-	5
Blocos finalizados	8	-	10

Fonte: Elaborado pelo autor.

5.5.3 Análise da Disponibilidade

A disponibilidade do sistema foi calculada considerando a capacidade de processar votos durante todo o período de teste. Seja V_t o número total de votos submetidos e V_f o número de votos finalizados, a disponibilidade D é definida por:

$$D = \frac{V_f}{V_t} \times 100\%$$

Substituindo os valores obtidos, considerando os três votos da Fase 1 e os cinco votos da Fase 3:

$$D = \frac{8}{8} \times 100\% = 100\%$$

O resultado demonstra que o sistema manteve disponibilidade total mesmo após a falha de um validador. Todos os votos submetidos, tanto antes quanto depois da falha, foram corretamente finalizados na blockchain. Não houve interrupção de serviço ou necessidade de intervenção manual para restaurar a operação do sistema.

Esse comportamento decorre diretamente da regra de quórum utilizada pelo protótipo. Como a finalização de blocos depende de maioria simples entre os validadores configurados, a quantidade de falhas por parada que pode ser tolerada é dada por:

$$f = \left\lfloor \frac{n-1}{2} \right\rfloor = \left\lfloor \frac{2}{2} \right\rfloor = 1$$

Assim, para $n = 3$, o sistema tolera uma falha por parada mantendo a maioria de $\lceil n/2 \rceil = 2$ validadores. Após a falha do Node 2, os dois validadores restantes (Node 1 e Node 3) constituem maioria suficiente para atingir quórum e finalizar blocos. O quórum necessário para finalização é de $\lfloor n/2 \rfloor + 1 = 2$ aprovações, que pode ser satisfeito pelos dois validadores remanescentes.

5.5.4 Análise

O teste de tolerância a falhas demonstrou que o sistema implementa corretamente os mecanismos de resiliência especificados para o cenário avaliado. A capacidade de manter operação contínua após a falha de um validador é uma propriedade relevante em sistemas distribuídos, nos quais falhas de hardware, software ou rede podem ocorrer.

O primeiro aspecto que merece destaque é a continuidade de operação. O sistema não experimentou interrupção de serviço. Votos continuaram sendo processados imediatamente após a falha, sem necessidade de intervenção manual ou reconfiguração. Esse comportamento demonstra que o protocolo de consenso é verdadeiramente descentralizado, não dependendo de nenhum validador específico para manter a operação.

O segundo aspecto relevante é a preservação da consistência. A blockchain permaneceu consistente entre os validadores remanescentes. Não foram observadas divergências ou blocos conflitantes. Os dois validadores restantes continuaram seguindo o mesmo protocolo, propondo e validando blocos de forma coordenada, mantendo a integridade da cadeia.

O terceiro aspecto importante é que a falha não impediu a continuidade do consenso entre os validadores remanescentes. Embora este experimento não tenha sido projetado para caracterizar desempenho ou vazão, os registros coletados mostram que os votos submetidos após a falha continuaram sendo finalizados dentro da janela de observação. Assim, o resultado sustenta a análise de disponibilidade do protótipo, sem pretender estabelecer limites de desempenho sob carga.

5.6 Discussão dos Resultados

A análise integrada dos três testes permite compreender o comportamento global do sistema e identificar padrões que não seriam evidentes através da análise isolada de cada teste.

5.6.1 Validação dos Requisitos

A Tabela 11 apresenta uma comparação entre as questões investigativas, os critérios de validação e os resultados obtidos nos testes experimentais. Os critérios definidos para o escopo experimental foram atendidos, demonstrando que o sistema implementa corretamente as propriedades avaliadas.

Tabela 11 – Comparação entre questões investigativas e resultados obtidos

Questão	**Critério**	**Obtido**	**Status**
Q1	Finalizar votos válidos no cenário experimental	5/5 votos	Atendido
Q2	Rejeitar voto duplicado	1 voto finalizado	Atendido
Q3	Manter finalização após falha por parada	8/8 votos	Atendido
Q3	Preservar quórum com uma falha	2/3 validadores	Atendido

Fonte: Elaborado pelo autor.

O sistema demonstrou integridade no teste funcional, com todos os votos válidos corretamente finalizados em condições normais de operação. Esse resultado confirma que o pipeline de processamento, desde a submissão até a persistência, opera de forma consistente no cenário avaliado. A unicidade de voto também foi validada, embora com oportunidade de otimização no pipeline de validação, conforme discutido no segundo teste. A disponibilidade manteve-se em 100% no experimento de falha, demonstrando que o protótipo preserva finalização de votos quando dois dos três validadores permanecem ativos. A tolerância a falhas operou conforme o modelo de quórum adotado, mantendo operação com dois terços dos validadores ativos.

5.6.2 Limitações Identificadas

Uma limitação identificada durante os testes experimentais é a validação tardia de duplicação. Embora o sistema detecte e bloqueie votação dupla de forma efetiva, a validação pode ocorrer após o aceite inicial do voto no fluxo de processamento. Isso resulta em overhead desnecessário de processamento e propagação de mensagens inválidas pela rede.

A validação no mempool deve ser compreendida como uma otimização local, e não como o mecanismo definitivo de segurança contra votação dupla. Em uma rede distribuída, diferentes nós podem receber votos conflitantes em ordens distintas, de modo que um mempool pode observar apenas uma das tentativas enquanto outra ainda não foi propagada. Por essa razão, a garantia forte de unicidade deve permanecer associada ao estado consolidado da blockchain e à validação independente das propostas de bloco pelos validadores.

É importante ressaltar que essa limitação não compromete a correção ou segurança do sistema. Ela manifesta-se apenas na eficiência operacional, não nas garantias fundamentais de integridade e consistência. Um voto que é finalizado na blockchain possui as mesmas garantias de segurança, independentemente da presença de tentativas de votação dupla.

5.6.3 Trabalhos Futuros

Com base nos resultados obtidos e nas limitações identificadas, propõem-se duas direções principais para trabalhos futuros. A primeira direção envolve melhoria de segurança e eficiência. O aprimoramento da validação prévia no mempool poderia reduzir o overhead associado a votos duplicados já conhecidos localmente, desde que preservada a validação final contra o estado consolidado da blockchain. A adição de mecanismos de *rate limiting* por eleitor poderia prevenir ataques de negação de serviço, nos quais um eleitor malicioso submete múltiplos votos inválidos para sobrecarregar o sistema. A investigação de resistência a outros tipos de ataques, como ataques de eclipse na camada P2P ou ataques de temporização no consenso, também seria relevante.

A segunda direção envolve testes adicionais. A avaliação do comportamento com mais de três validadores permitiria compreender como o sistema escala com o aumento do número de participantes no consenso. A simulação de latência de rede em ambiente WAN permitiria avaliar o comportamento em condições mais realistas de produção. A avaliação de mecanismos de recuperação após falha, incluindo sincronização de estado e reconexão de validadores, completaria a validação da resiliência do sistema.

6 CONCLUSÃO

Este trabalho investigou, em escala experimental, a viabilidade técnica de um sistema de votação eletrônica descentralizado baseado em rede *peer-to-peer*, *blockchain* e consenso *Proof of Authority*. O recorte adotado não teve como objetivo propor substituição para infraestruturas eleitorais nacionais, mas avaliar se uma arquitetura permissionada, composta por validadores conhecidos, seria capaz de sustentar propriedades fundamentais para um processo de votação no cenário controlado definido: integridade do resultado, unicidade de voto e continuidade de operação diante de uma falha por parada.

A implementação desenvolvida em Go, com comunicação *peer-to-peer* via libp2p, estruturou essas propriedades em um protótipo funcional. O uso de *blockchain* forneceu um registro encadeado e verificável dos votos finalizados, enquanto o consenso PoA permitiu coordenar a produção de blocos sem recorrer a competição computacional ou mecanismos econômicos. A seleção determinística de líder por *round-robin*, a validação criptográfica dos votos e a finalização por maioria simples formaram o núcleo operacional avaliado nos experimentos.

6.1 Contribuições do Trabalho

A primeira contribuição deste trabalho foi a adaptação de um consenso PoA para o contexto de votação em uma rede permissionada. Essa adaptação incluiu a modelagem da seleção de líder, a definição do quórum de finalização e a formalização das propriedades esperadas de segurança e vivacidade. Com isso, o trabalho mostrou como um protocolo relativamente simples pode ser organizado para atender a requisitos fundamentais de uma aplicação eleitoral em ambiente controlado.

A segunda contribuição foi a implementação completa do protótipo. O sistema integra camada de rede, mempool, validação de votos, consenso, persistência e reconstrução de estado, permitindo observar o comportamento da proposta para além de uma especificação conceitual. O código-fonte do protótipo está disponível publicamente em `https://github.com/matscats/peer-vote`. A arquitetura modular baseada em *Hexagonal Architecture* também favorece evolução futura do código, pois separa o núcleo de regras de consenso e validação das dependências de infraestrutura.

A terceira contribuição foi a validação experimental do protótipo. Os testes realizados foram organizados em torno das três questões investigativas do trabalho: validação funcional para integridade do resultado, teste de unicidade para prevenção de voto duplicado e teste de tolerância a falhas para disponibilidade sob uma falha *crash-stop*. Essa

avaliação forneceu evidências empíricas do comportamento do sistema e estabeleceu uma base inicial para comparações e extensões futuras.

6.2 Validação das Questões Investigativas

Em relação à primeira questão investigativa, os resultados indicaram que o sistema foi capaz de finalizar corretamente todos os votos válidos submetidos no teste funcional, dentro do cenário experimental com número mínimo de validadores. A rotação determinística dos líderes, a obtenção de quórum e a consistência da cadeia entre validadores confirmaram que o fluxo de submissão, propagação, validação e finalização operou conforme o modelo definido.

Quanto à segunda questão, o teste de unicidade demonstrou que a tentativa de votação dupla foi detectada e bloqueada no cenário avaliado. O estado final da *blockchain* manteve apenas um voto para o eleitor testado, preservando a propriedade de que cada identificador de eleitor pode produzir no máximo um voto finalizado. Esse resultado é importante porque a unicidade não depende de um validador central: cada nó executa sua própria verificação contra o estado consolidado.

Para a terceira questão, o teste de tolerância a falhas mostrou que o sistema continuou finalizando votos após a parada abrupta de um dos três validadores. Com dois validadores remanescentes, o quórum de maioria simples ainda pôde ser satisfeito, permitindo continuidade de operação no cenário avaliado. Esse resultado confirma a coerência entre o modelo de falhas *crash-stop*, a regra de maioria e o comportamento observado experimentalmente.

6.3 Limitações e Trabalhos Futuros

As conclusões deste trabalho devem ser interpretadas dentro do escopo experimental adotado. Os testes foram executados em ambiente local, com três validadores, sem latência WAN, perda de pacotes, particionamento de rede ou carga elevada. Portanto, os resultados não caracterizam desempenho em escala, vazão máxima, comportamento sob redes adversas ou adequação imediata a eleições de grande porte.

Outra limitação está relacionada ao tratamento de votos duplicados. Embora a implementação preserve a unicidade no estado final da cadeia, a validação de duplicidade pode ocorrer tardiamente no fluxo de processamento. Em alguns cenários distribuídos, um voto conflitante pode permanecer no mempool de um validador até que seja rejeitado durante a validação de uma proposta de bloco. Melhorias futuras devem tratar esse ponto como uma otimização de eficiência, preservando a validação definitiva contra o estado consolidado da *blockchain*.

Como trabalhos futuros, destacam-se a avaliação com maior número de validadores, a execução em ambientes com latência realista, a simulação de perda de pacotes e particionamentos temporários, além da análise de mecanismos de recuperação após falhas. Também são caminhos relevantes o aprimoramento da política de mempool, a introdução de mecanismos de limitação de taxa e a investigação de ataques na camada *peer-to-peer*, como ataques de eclipse.

6.4 Considerações Finais

Em síntese, o trabalho mostrou que a combinação de rede *peer-to-peer*, *blockchain* e consenso *Proof of Authority* é uma abordagem tecnicamente viável para construir um protótipo de votação descentralizada em rede permissionada. Os experimentos não encerram a discussão sobre votação eletrônica descentralizada, mas validam um conjunto fundamental de propriedades em um protótipo funcional. A partir dessa base, a proposta pode evoluir para avaliações mais amplas de escala, desempenho, segurança adversarial e adequação a contextos institucionais reais de menor porte.

REFERÊNCIAS

CASTRO, M.; LISKOV, B. Practical byzantine fault tolerance. In: *Proceedings of the Third Symposium on Operating Systems Design and Implementation (OSDI '99)*. New Orleans, LA, USA: USENIX Association, 1999. p. 173–186. Disponível em: http://pmg.csail.mit.edu/papers/osdi99.pdf. Acesso em: 15 out. 2025.

CHRISTYONO, B. B. A.; WIDJAJA, M.; WICAKSANA, A. Go-ethereum for electronic voting system using clique as proof-of-authority. *TELKOMNIKA Telecommunication, Computing, Electronics and Control*, v. 19, n. 5, p. 1565–1572, 2021. Estudo sobre votação eletrônica baseada em Ethereum, Go-Ethereum e consenso Clique.

COCKBURN, A. *Hexagonal Architecture*. 2005. Artigo técnico. Disponível em: https://alistair.cockburn.us/hexagonal-architecture/. Acesso em: 17 jun. 2026.

CONG, T. D. et al. Proof-of-stake consensus mechanisms for future blockchain networks: Fundamentals, applications, and opportunities. *Sensors*, MDPI, v. 25, n. 9, p. 1320, 2025. Disponível em: https://www.mdpi.com/1099-4300/25/9/1320. Acesso em: 15 out. 2025.

COULOURIS, G. et al. *Distributed Systems: Concepts and Design*. 5. ed. Harlow: Addison-Wesley / Pearson, 2012.

EVANS, E. *Domain-Driven Design: Tackling Complexity in the Heart of Software*. Boston: Addison-Wesley, 2003.

Gnutella Developers. *Gnutella Protocol Specification v0.4*. s.l., 2001. Disponível em: rfc-gnutella.sourceforge.net/developer/stable/index.html. Acesso em: 01 out. 2025.

HABER, S.; STORNETTA, W. S. How to time-stamp a digital document. *Journal of Cryptology*, Springer, v. 3, n. 2, p. 99–111, 1991. Propostas iniciais de encadeamento por hashes e timestamping.

JR., C. Z. Trading old errors for new errors? the impact of electronic voting technology on party label votes in brazil. *Electoral Studies*, Elsevier, v. 43, p. 10–20, 2016. Análise dos efeitos da votação eletrônica sobre erros e comportamento do eleitor no Brasil.

LAMPORT, L.; SHOSTAK, R.; PEASE, M. The byzantine generals problem. *ACM Transactions on Programming Languages and Systems (TOPLAS)*, Association for Computing Machinery, v. 4, n. 3, p. 382–401, 1982. Artigo clássico que formaliza o problema dos generais bizantinos. Disponível em: https://lamport.org/pubs/pubs.html. Acesso em: 01 out. 2025.

LOWRY, S. Z.; VORA, P. L. *Desirable Properties of Voting Systems*. 2009. End-to-End Workshop Discussion Paper, National Institute of Standards and Technology. Disponível em: https://www.nist.gov/publications/desirable-properties-voting-systems. Acesso em: 01 jun. 2026.

MANOLACHE, M. A.; MANOLACHE, S.; TAPUS, N. Decision making using the blockchain proof of authority consensus protocol. *Procedia Computer Science*, Elsevier, v. 199, p. 580–588, 2022. Disponível em: https://doi.org/10.1016/j.procs.2022.01.071. Acesso em: 15 out. 2025.

NAKAMOTO, S. *Bitcoin: A Peer-to-Peer Electronic Cash System*. 2008. Whitepaper. Disponível em: https://bitcoin.org/bitcoin.pdf. Acesso em: 01 out. 2025.

Protocol Labs. *Kademlia DHT*. 2026. Documentação oficial libp2p. Disponível em: https://libp2p.io/docs/kademlia-dht/. Acesso em: 17 jun. 2026.

Protocol Labs. *Rendezvous*. 2026. Documentação oficial libp2p. Disponível em: https://libp2p.io/docs/rendezvous/. Acesso em: 17 jun. 2026.

Protocol Labs. *What is Publish/Subscribe*. 2026. Documentação oficial libp2p. Disponível em: https://docs.libp2p.io/concepts/pubsub/. Acesso em: 17 jun. 2026.

SHARMA, T.; KRISHNA, C. R.; BAHGA, A. A cost-efficient proof-of-stake-voting based auditable blockchain e-voting system. *IOP Conference Series: Materials Science and Engineering*, IOP Publishing, v. 1099, n. 1, p. 012038, 2021. Discussão sobre custo, auditabilidade e desempenho em sistemas de votação baseados em blockchain.

STOICA, I. et al. Chord: A scalable peer-to-peer lookup service for internet applications. In: *Proceedings of the 2001 conference on Applications, technologies, architectures, and protocols for computer communications (SIGCOMM '01)*. New York: ACM, 2001. p. 149–160.

TANENBAUM, A. S.; STEEN, M. V. *Distributed Systems*. 3. ed. Amsterdam: Vrije Universiteit Amsterdam, 2017. Preliminary version 3.01 (2017). Disponível em: v.ua.nl/ds/ ou repositórios acadêmicos.

The Go Authors. *Documentation*. 2026. Documentação oficial. Disponível em: https://go.dev/doc/. Acesso em: 17 jun. 2026.

The Go Authors. *Package ed25519*. 2026. Documentação oficial da biblioteca padrão Go. Disponível em: https://pkg.go.dev/crypto/ed25519. Acesso em: 17 jun. 2026.

The Go Authors. *Package sha256*. 2026. Documentação oficial da biblioteca padrão Go. Disponível em: https://pkg.go.dev/crypto/sha256. Acesso em: 17 jun. 2026.

WOOD, G. *Ethereum: A Secure Decentralised Generalised Transaction Ledger*. Zug, Suíça, 2014. Versão revisada 2023. Disponível em: ethereum.github.io/yellowpaper/. Acesso em: 01 out. 2025.