UNIVERSIDADE ESTADUAL DE CAMPINAS
Faculdade de Engenharia Elétrica e de Computação

Yoshitomi Eduardo Maehara Aliaga

Estudo sobre mecanismos de consenso de baixo custo para Blockchain

Campinas

2019

UNIVERSIDADE ESTADUAL DE CAMPINAS
Faculdade de Engenharia Elétrica e de Computação

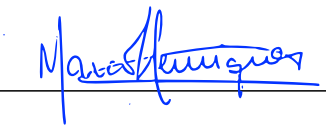
Yoshitomi Eduardo Maehara Aliaga

Estudo sobre mecanismos de consenso de baixo custo para Blockchain

Dissertação apresentada à Faculdade de Engenharia Elétrica e de Computação da Universidade Estadual de Campinas como parte dos requisitos exigidos para a obtenção do título de Mestre em Engenharia Elétrica, na Área de Engenharia de Computação.

Orientador: Prof. Dr. Marco Aurélio Amaral Henriques

Este exemplar corresponde à versão final da dissertação defendida pelo aluno Yoshitomi Eduardo Maehara Aliaga e orientada pelo Prof. Dr. Marco Aurélio Amaral Henriques.



Campinas

2019

Ficha catalográfica
Universidade Estadual de Campinas
Biblioteca da Área de Engenharia e Arquitetura
Elizangela Aparecida dos Santos Souza - CRB 8/8098

M268e

Maehara Aliaga, Yoshitomi Eduardo, 1990-
Estudo sobre mecanismos de consenso de baixo custo para Blockchain / Yoshitomi Eduardo Maehara Aliaga. – Campinas, SP : [s.n.], 2019.

Orientador: Marco Aurélio Amaral Henriques.
Dissertação (mestrado) – Universidade Estadual de Campinas, Faculdade de Engenharia Elétrica e de Computação.

1. Cadeia de blocos. 2. Consenso distribuido (Computação). 3. Bitcoin. I. Henriques, Marco Aurélio Amaral, 1963-. II. Universidade Estadual de Campinas. Faculdade de Engenharia Elétrica e de Computação. III. Título.

Informações para Biblioteca Digital

Título em outro idioma: Study on low cost consensus mechanisms for Blockchain
Palavras-chave em inglês:
Blockchain
Distributed consensus (Computing)
Bitcoin
Área de concentração: Engenharia de Computação
Titulação: Mestre em Engenharia Elétrica
Banca examinadora:
Marco Aurélio Amaral Henriques [Orientador]
Christian Rodolfo Esteve Rothenberg
Julio Cesar López Hernández
Data de defesa: 11-04-2019
Programa de Pós-Graduação: Engenharia Elétrica

Identificação e informações acadêmicas do(a) aluno(a)
- ORCID do autor: https://orcid.org/0000-0003-4383-3013
- Currículo Lattes do autor: http://lattes.cnpq.br/2020733886633076

COMISSÃO JULGADORA - DISSERTAÇÃO DE MESTRADO

Candidato: Yoshitomi Eduardo Maehara Aliaga **RA:** 192686
Data da Defesa: 11 de abril de 2019
Título: "Estudo sobre mecanismos de consenso de baixo custo para Blockchain"

Prof. Dr. Marco Aurélio Amaral Henriques (Presidente, FEEC/UNICAMP)
Prof. Dr. Christian Rodolfo Esteve Rothenberg (FEEC/UNICAMP)
Prof. Dr. Julio Cesar Lopez Hernandez (IC/UNICAMP)

A ata de defesa, com as respectivas assinaturas dos membros da Comissão Julgadora, encontra-se no SIGA (Sistema de Fluxo de Dissertação/Tese) e na Secretaria de Pós-Graduação da Faculdade de Engenharia Elétrica e de Computação.

A Jesus, meu Senhor e Salvador. . . .

Agradecimentos

- A Jesus, meu Senhor e Salvador, pela sua presença constante na minha vida, pelo auxílio nas minhas escolhas e me confortar nas horas difíceis
- Aos meus pais, Víctor e Doris, por todo amor, carinho e apoio incondicionais. Amo muito vocês!
- A minha irmã Rocio, pelo apoio e compreensão.
- A meu cunhado Luis, pelo apoio e compreensão.
- A meu orientador, Professor Marco Aurélio Amaral Henriques, pelos importantes ensinamentos, pela paciência e apoio na elaboração deste projeto
- Aos meus colegas Antônio Unias de Lucena, Victor Cerqueira Leal e Diego Fernandes Gonçalves Martins pelas suas amáveis colaborações e apoio neste projeto.
- O presente trabalho foi realizado com apoio da Coordenação de Aperfeiçoamento de Pessoal de Nível Superior - Brasil (CAPES) - Código de Financiamento 001.

"Sonhe com aquilo que você quer ser,
porque você possui apenas uma vida
e nela só se tem uma chance
de fazer aquilo que quer."
(Clarice Lispector)

Resumo

O conceito de *blockchain* (cadeia de blocos) introduzido com a invenção da criptomoeda *Bitcoin* tem sido objeto de intensas discussões sobre seu potencial de viabilizar novos tipos de aplicações e serviços na Internet. As *blockchains* mais usadas atualmente (nos sistemas *Bitcoin* e *Ethereum*) são baseadas no conceito de Prova de Trabalho (*Proof-of-Work* ou PoW), no qual o consenso entre partes adversárias é garantido pela execução de um grande volume de cálculos de funções de dispersão (*hash*). Como consequência, tem havido uma grande competição e um aumento constante dos requisitos computacionais e de energia necessários para criar blocos em uma destas *blockchains*, o que cada vez mais afasta os usuários comuns e concentra o controle do sistema nas mãos de alguns poucos grupos. O primeiro objetivo deste trabalho foi estudar e comparar mecanismos de consenso alternativos para *blockchains*. Um segundo objetivo foi definir os requisitos que um novo mecanismo de consenso deve possuir para viabilizar a participação de praticamente qualquer usuário na *blockchain*, sem exigir recursos vultuosos para investimentos em equipamentos e energia elétrica. Como terceiro objetivo, tivemos o projeto e implementação de uma nova *blockchain* que serviu como prova de conceito para o novo mecanismo de consenso proposto. Finalmente, foram feitas discussões e comparações do novo mecanismo implementado com similares na literatura, a fim de mostrar o quão próximo se chegou de um mecanismo ideal, que facilita a participação de todos e torna as *blockchains* mais distribuídas, como idealizadas originalmente.

Palavras-chave: *Blockchain*; Mecanismos de Consenso; Prova de Trabalho; *Bitcoin*; *Ethereum*.

Abstract

The blockchain concept, introduced with the invention of Bitcoin cryptocurrency, has been the subject of intensive discussions about its potential to enable new kinds of applications and services on Internet. The most commonly used blockchains (in the Bitcoin and Ethereum systems) are based on the concept of Proof-of-Work (PoW), in which the consensus between opposing parties is guaranteed by the execution of a large volume of hash function calculations. As a consequence, there has been a great deal of competition and a steady increase in the computational and power requirements needed to create blocks in one of these blockchains, which puts away ordinary users and concentrates the control of the system in the hands of a few groups. The first objective of this work was to study and compare alternative consensus mechanisms for blockchains. A second objective was to define the requirements that a new consensus mechanism must have to enable practically any user to participate in the blockchain, without demanding huge resources for investments in equipment and energy. As a third objective, we had the design and implementation of a new blockchain that served as a proof of concept for the proposed new consensus mechanism. Finally, we discussed and compared the new implemented mechanism with similar ones, in order to show how close it has come to an ideal consensus, which facilitates the participation of everybody and makes blockchains more distributed, as originally idealized.

Keywords: Blockchains; Consensus Mechanism; Proof-of-Work; Bitcoin; Ethereum.

Lista de ilustrações

Lista de tabelas

Sumário

1 Introdução

Nos últimos anos as *blockchains* têm atraído muita atenção devido ao sucesso da criptomoeda Bitcoin e da rede *Ethereum*. Aplicações baseadas em *blockchains* tornam-se cada dia mais populares e surgem propostas interessantes e inovadoras para uso de *blockchains* em cartórios digitais, gestão de identidades, sistemas de reputação, armazenamento confiável de informação, entre outras.

Entre as novidades que a *blockchain* traz, destacam-se os mecanismos de consenso, que permitem que partes desconhecidas entre si e até mesmo concorrentes cheguem a um acordo sobre o estado atual e passado dos dados armazenados na *blockchain*, permitindo criar aplicações distribuídas descentralizadas de forma confiável.

Dentre estes mecanismos, o mais conhecido é o *Proof-of-Work* (PoW), utilizado na criptomoeda *Bitcoin* (BASHIR, 2017). Porém, apesar de dar uma solução ao problema do consenso, ele também tem gerado problemas como a necessidade de alto poder computacional e de elevado consumo de energia elétrica (ZHENG *et al.*, 2017b).

Por este motivo, foram propostos outros mecanismos tais como *Proof-of-Stake, Proof-of-Activity, Delegated-Proof-of-Stake,* entre outros, como uma forma de contornar as desvantagens presentes no PoW (ZHENG *et al.*, 2017b; BANO *et al.*, 2017). Apesar de contornar essas desvantagens, estes mecanismos ainda têm alguns problemas.

No presente trabalho é feita uma comparação dos principais mecanismos de consenso em uso ou propostos, com o fim de permitir uma avaliação mais precisa dos pontos fortes e fracos dos mesmos por aqueles que pretendem projetar uma nova *blockchain* pública.

Além disso, este trabalho procura determinar as caraterísticas que seriam desejáveis para um novo mecanismo de consenso com melhores condições para a participação de qualquer usuário, isto é, que diminuíssem as exigências de investimento para participação, evitando assim a concentração da capacidade de criar novos blocos em alguns poucos grupos mais bem equipados material e financeiramente.

Com base nas características levantadas é apresentada uma proposta de algoritmo de consenso baseado em sorteio. Trata-se de uma abordagem simplificada de um algoritmo *Proof-of-Stake,* (PoS) onde novos blocos podem ser gerados pelos nós participantes com base na probabilidade de serem contemplados em um sorteio via função hash. A versão inicial deste consenso considera que todos os participantes têm iguais chances de serem sorteados. Entretanto, ele pode ser facilmente alterado para dar maior chance aos participantes que

tiverem mais recursos investidos no sistema (*stake*), aproveitando então de alguns benefícios do esquema PoS.

O algoritmo proposto é uma alternativa ao problema da centralização do poder computacional, pois dá condições iguais para os participantes serem criadores de blocos (e serem remunerados por recompensas e taxas), reforçando o caráter distribuído da *blockchain* e aumentando a segurança e a tolerância a falhas.

A proposta foi implementada e uma *blockchain* foi criada desde o zero como prova de conceito. Vários testes foram realizados e, apesar de ainda haver muito espaço para melhorias, os resultados mostram que o novo mecanismo de consenso pode ser uma alternativa interessante para novas *blockchains* que desejam ter uma base de participantes mais ampla e distribuída.

O texto da dissertação está organizado da seguinte forma: o Cap. 2 descreve mais detalhadamente a estrutura de uma *blockchain* típica. No Cap. 3 são feitos um estudo e uma comparação detalhados das principais *blockchains* propostas na literatura. Como os estudos até este ponto mostraram que o mecanismo PoS pode ser uma interessante alternativa ao custoso PoW, o Cap. 4 é dedicado ao estudo mais detalhado de uma das principais referências de PoS atualmente: o protocolo Casper que está sendo implementado e testado para substituir o PoW em uma futura versão da plataforma Ethereum. Em seguida, no Cap. 5 é apresentada a proposta do novo mecanismo de consenso e são discutidas suas principais características e formas de funcionamento. Os testes comparativos do novo consenso são apresentados no Cap. 6, mostrando como ele se comporta de acordo com mudanças de seus principais parâmetros. O Cap. 7 discute os principais resultados deste trabalho e aponta algumas possíveis direções para trabalhos futuros.

2 Estrutura básica de uma *blockchain*

A *blockchain* é uma cadeia de blocos interligados (Fig. 1) através de valores específicos (*hashes*) calculados por funções de dispersão (também conhecidas como funções de espalhamento ou de *hash*). A utilização de funções de *hash* junto com a criptografia assimétrica na forma de assinaturas digitais faz com que a *blockchain* seja robusta a fraudes (PEREZ-SOLA; HERRERA-JOANCOMARTI, 2014).

Os blocos da cadeia guardam informações de forma semelhante a um livro razão em contabilidade (ZHENG *et al.*, 2017b). Nas *blockchains* voltadas para aplicações com criptomoedas, tais informações são transações de remessa de valores de uma entidade (endereço ou chave) para outra. Os blocos são armazenados em nós que são interligados através de uma rede *peer-to-peer* (P2P), o que resulta em uma grande redundância das informações pelos nós, dando à *blockchain* um caráter distribuído.

Como é descrito por Greve et al. (GREVE *et al.*, 2018) o conjunto de transações é implementado como uma maquina de estados replicada, o que permite ter uma réplica dos dados em cada nó dentro da rede P2P, provendo tolerância a falhas por meio da replicação ativa (COULORIS *et al.*, 2013).

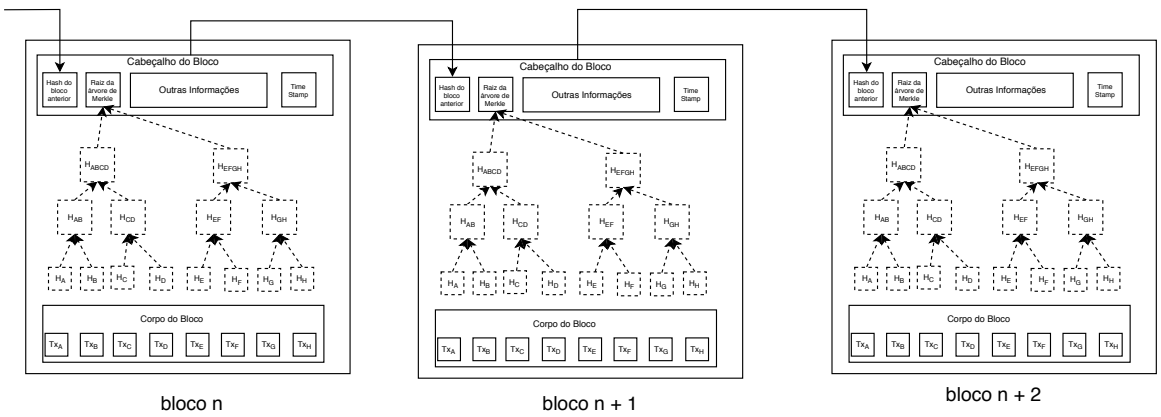


Figura 1 – Cadeia de blocos

Os nós criam, repassam, aceitam e conferem um novo bloco de forma descentralizada por meio de um mecanismo de consenso. O primeiro bloco de uma *blockchain* é chamado *Genesis Block*: ele não possui bloco anterior e a partir dele são criados os outros.

2.1 Classificação

As *blockchains* podem ser classificadas em um Públicas e Privadas.

2.1.1 Públicas

Neste tipo de *blockchain* qualquer nó participante pode verificar transações, criar blocos e participar do processo de consenso (LIN; LIAO, 2017). Não existe uma autoridade ou grupo de nós central que possua mais poder que os outros nós(SANKAR *et al.*, 2017).

Segundo (GREVE *et al.*, 2018), o conjunto de nós da rede é desconhecido e sua composição é dinâmica, o que permite entradas e saídas aleatórias dos nós. Estes não precisam de uma identificação já que normalmente costumam ser anônimos. Este tipo de *blockchain* pode atuar em uma escala bem grande, já que não há necessidade de controle dos participantes e nem de confiança mútua entre eles.

As características principais de uma *blockchain* pública são:

a) Descentralização: o processamento das transações e blocos é feito por qualquer nó da rede;

b) Persistência: toda transação armazenada na *blockchain* não pode ser alterada ou removida sem quebrar toda a consistência dos blocos posteriores;

c) Anonimato: cada participante interage com a *blockchain* por meio de endereços não diretamente relacionados com sua identidade;

d) Auditabilidade: toda transação é validada e registrada com um carimbo de tempo, o que permite a qualquer entidade verificá-la e rastreá-la permanentemente na *blockchain*.

2.1.2 Privadas

Neste tipo de *blockchains* nem todo nó pode criar blocos ou participar do processo de consenso e o processo todo tende a ser centralizado (SANKAR *et al.*, 2017). A *blockchain* pode ser mais ou menos centralizada, dependendo do tamanho do grupo pré-selecionado para participar do processo de consenso e manutenção dos blocos.

Diferentemente das públicas, nas *blockchains* privadas há a necessidade de se ter controle sobre os participantes, os quais devem estar devidamente identificados e ter sua atuação bem definida em uma organização ou grupo especifico. Esta *blockchain* atende normalmente interesses corporativos onde se encontram características compatíveis com o tipo.

Neste trabalho o foco está nas *blockchains* públicas, já que estamos buscando facilitar ainda mais a participação de novos usuários nas mesmas a fim de torná-las mais distribuídas, seguras e tolerantes a falhas.

2.2 Estrutura

Cada bloco consiste geralmente de duas partes: cabeçalho e corpo. A Fig. 2 mostra os detalhes de um bloco tipico da *blockchain* da criptomoeda *Bitcoin*.

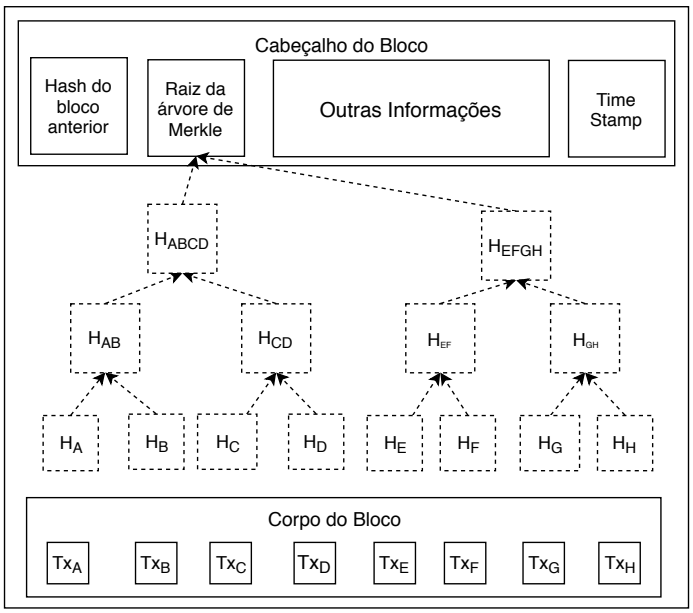


Figura 2 – Estrutura interna de um bloco da *blockchain* do *Bitcoin*

A seguir cada parte é melhor detalhada.

a. Cabeçalho

(i) TimeStamp: representa a hora e a data universais (formato UNIX) em que o bloco foi criado.

(ii) Raiz da árvore de Merkle: é o hash que representa todas as transações organizadas como folhas de uma árvore de Merkle (BRYSON *et al.*, 2017).

(iii) Outras Informações: Estas informações são auxiliares mas podem ser úteis para algumas criptomoedas (PEREZ-SOLA; HERRERA-JOANCOMARTI, 2014).

- **Número de bits:** indica a dificuldade para realizar a mineração, ou seja, a quantidade de bits zero no hash do desafio.

- **Nonce:** é o valor arbitrário usado na criação de blocos para se produzir o hash do desafio; é um valor normalmente aleatório (é mais usual em *blockchains* do tipo PoW).
- **Versão:** indica a versão das regras de validação dos blocos, isto é, quais regras seguir para obtenção de consenso entre os blocos.

(iv) Hash do bloco anterior: contém o valor do hash calculado para o bloco anterior, que já está consolidado. Este valor é usado no cálculo do hash do bloco atual, o que cria a amarração entre blocos consecutivos.

b. Corpo do Bloco

(i) **Transações:** São estrutura de dados que registram as interações (troca de valores, por exemplo) entre os usuários da rede que faz uso da *blockchain*. Uma interação envolve uma assinatura digital de uma mensagem do remetente para o endereço do destinatário, além de entradas e saídas com os valores transacionados, no caso de aplicações com criptomoedas (GREVE *et al.*, 2018).

A quantidade máxima de transações que um bloco pode ter depende do tamanho do bloco e do tamanho da transação. No caso do *Bitcoin* , o bloco padrão é de 1 MB, mas mudanças já foram propostas para se aumentar este tamanho, o que levou a uma ramificação para outra criptomoeda (BitcoinCash), com 8 MB atualmente (até 32 MB no futuro).

A validação de cada transação é bem rápida, pois só depende do cálculo da função de verificação de sua assinatura que, no caso de *Bitcoin* , é feita sobre curvas elípticas (ECDSA). Entretanto, a tentativa de inserção de transações inválidas não pode ser feita com a mesma facilidade. A verificação de um bloco consiste em ver que o bloco está bem formado, se o *hash* é válido e atende o número de bits da dificuldade, se seu tamanho está dentro do limite da rede e se o conjunto de transações é válido (GREVE *et al.*, 2018).

Devido à ligação que um bloco tem com todos os seus anteriores, é praticamente impossível alterar alguma transação que já tenha sido incluída em um bloco. A alteração exigiria um recálculo de *hash* não só do bloco sendo alterado, mas de todos os demais que estejam após ele. Além disso, este recálculo deveria ocorrer antes que qualquer outro nó da rede acrescentasse novos blocos à *blockchain*, algo que ocorre com muita frequência (em média, a cada 10 minutos no Bitcoin) e inviabiliza, portanto, ataques de alteração de conteúdo em blocos passados.

Outra característica fundamental de uma *blockchain* é o mecanismo de consenso, isto é, o algoritmo que permite a criação de um novo bloco num ambiente distribuído e de forma consensual entre os nós da rede P2P que são adversários entre si. A importância desta

característica é tal que, sem ela, seria muito difícil que a *blockchain* trabalhasse com rapidez e de forma estável e confiável.

2.3 Funcionamento básico de uma estrutura de *blockchain*

Como é descrito por Lin e Liao (LIN; LIAO, 2017), o funcionamento do processo principal da *blockchain* consiste em coletar as transações enviadas na forma de mensagem pela rede por *broadcast*. Estas transações normalmente residem em um *pool* de transações, de onde um nó recolhe algumas para verificar e inserir em um novo bloco. Se uma mensagem (transação) está correta então ela será armazenada no bloco junto com as demais. Sendo atingido o critério adotado pelo nó para o número máximo de transações a incluir, este procura calcular o valor *hash* de todas as transações e formar com estes valores a Árvore de Merkle, cuja raiz será inserida no cabeçalho do bloco.

Após terem sido preenchidos todos os campos do cabeçalho, o bloco é anexado ao final da cópia da *blockchain* que o nó possui naquele momento e transmitido para outros nós que estejam conectados a (conheçam) este nó.

Ao receber o novo bloco, cada um dos demais nós da rede executará um mecanismo de consenso para verificar a consistência do bloco recém-gerado e decidir se o aceita ou se o mesmo deve ser descartado. Se o mecanismo de consenso é finalizado com sucesso, o novo bloco é anexado às cópias da *blockchain* que cada nó possui. Deste modo a *blockchain* vai sendo estendida de forma contínua com blocos construídos por diferentes nós participantes.

Pode ocorrer que dois ou mais nós produzam ao mesmo tempo o próximo bloco da *blockchain*. Neste caso, se os dois blocos forem válidos, ambos são anexados à cadeia, processo conhecido como bifurcação ($fork$). Tal situação é indesejável e precisa ser corrigida, conforme detalhado no próximo capítulo.

2.4 Conclusões

Como pôde ser visto neste capítulo, as *blockchains* permitem criar uma estrutura de dados segura, confiável e praticamente imutável (dentro de certos limites). A estrutura e o funcionamento básicos de uma *blockchain* foram detalhados e foi possível perceber que ambos são relativamente simples. A maior complexidade reside no mecanismo de consenso, o qual tem uma explicação detalhada no próximo capítulo. É este mecanismo de consenso que define os detalhes de funcionamento de uma *blockchain*, ditando suas características básicas.

3 Consenso em *blockchains*

3.1 Introdução

O processo de consenso dentro de um sistema distribuído é um tema muito estudado por muitas pessoas já que permite criar sistemas tolerantes a falhas. Bashir (BASHIR, 2017) define o consenso como o processo de acordo entre nós não confiáveis sobre o estado final de um conjunto de dados. Para atingir o consenso podem ser usados diferentes algoritmos. Quando múltiplos nós estão participando em um sistema distribuído e eles precisam concordar sobre um valor único, é mais difícil atingir o consenso. Este conceito de chegar ao consenso entre muitos nós é chamado de consenso distribuído.

Este capítulo apresenta vários mecanismos de consenso encontrados na literatura e faz uma comparação entre eles, mostrando as principais características de cada um, seus pontos fortes e pontos fracos. Este estudo servirá de base para levantarmos um conjunto de características desejáveis em uma nova *blockchain* para que não sofra dos problemas que assolam as principais *blockchains* em uso atualmente.

3.2 Mecanismo de Consenso

Bashir (BASHIR, 2017) define o mecanismo de consenso como um conjunto de passos (algoritmo) que são dados por todos ou pela maioria dos nós para entrar em um acordo a respeito de um estado proposto ou valor. Em outras palavras este algoritmo permite que dois ou mais processos concordem sobre um determinado valor.

Um mecanismo de consenso pode ser visto como uma solução ao clássico Problema dos Generais Bizantinos que consiste em resolver o dilema de atingir um consenso entre os participantes (que neste caso são os generais) com um objetivo comum (vencer uma batalha). Embora possam haver generais traidores, os quais têm objetivos opostos ao consenso e tentarão atrapalhar o processo, a ideia é que, se houver um número mínimo de generais leais (normalmente a maioria), o mecanismo de consenso garanta a conquista do objetivo comum, sem que os traidores consigam impedir. Uma discussão mais detalhada do Problema dos Generais Bizantinos é apresentada no Anexo A.

3.3 Consenso Assíncrono

Couloris et al. (COULORIS *et al.*, 2007) mencionam que nenhum algoritmo pode garantir um consenso em um sistema assíncrono, mesmo com a falha por colapso de um único processo. Em um sistema assíncrono, os processos podem responder às mensagens em tempos arbitrários; portanto, um processo falho não pode ser distinguido de um lento. Mas o fato de não poder garantir não significa que nunca se poderá chegar a um consenso.

Para tentar atingir um consenso, os autores propõem muitos métodos. O mais conhecido é utilizar o consenso aleatório, que consiste em inserir um elemento de aleatoriedade no comportamento dos processos de modo que o adversário não possa empregar uma estratégia de impedimento de maneira eficiente.

Por outro lado Wattenhoffer (WATTENHOFER, 2016) descreve um algoritmo criado por Ben-Or, que dá uma solução ao problema dos generais bizantinos em uma abordagem assíncrona, mas este não pode ser usado na prática porque precisa um tempo muito longo para convergir.

3.4 Principais Mecanismos de Consenso em Blockchains

A proposta da criptomoeda *Bitcoin* veio acompanhada da proposta de criação de uma *blockchain* baseada em um mecanismo de consenso usando a ideia de *Proof-of-Work*, a qual já havia sido criada anos antes como forma de se combater *spams*. A *blockchain* do *Bitcoin* foi considerada uma nova forma de ser resolver o Problema dos Generais Bizantinos e despertou o interesse da comunidade para novas aplicações além daquelas relacionadas a criptomoedas. Logo surgiram discussões sobre outras formas de se alcançar o consenso em *blockchain* usando técnicas alternativas ao *Proof-of-Work*.

A seguir são apresentados os principais mecanismos de consenso usados ou propostos para *blockchains* públicas, isto é, as *blockchains* que não estão sob controle de nenhum grupo de indivíduos ou organizações e que permitem a participação de qualquer novo usuário interessado na construção e validação de blocos.

3.4.1 *Proof-of-Work* (PoW)

É um mecanismo de consenso em que o nó busca uma solução para um desafio criptográfico a fim de poder criar um novo bloco. No caso específico do *Bitcoin*, a ideia consiste em variar um valor inteiro chamado *nonce* até que o valor do *hash* do cabeçalho do bloco (que contém este *nonce*) seja menor que um parâmetro pré-definido chamado *dificuldade*, como

mostrado na Fig. 3 (NAKAMOTO, 2009).

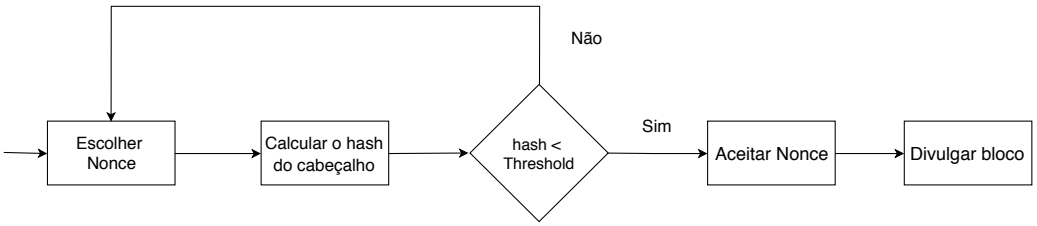


Figura 3 – Proof of Work na *blockchain* do *Bitcoin*

Cynthia Dwork e Moni Naor (DWORK; NAOR, 1992) fizeram uma proposta para lidar com spam de *emails*. Ela é baseada em uma função de cálculo moderadamente fácil, quando utiliza um valor de atalho. Porém, é difícil calcular a função inversa sem o valor de atalho. Este mecanismo exige que, para cada mensagem cujo remetente seja desconhecido, seja investido um certo tempo de computação, o que impede o envio de uma grande quantidade de *emails* por unidade de tempo.

Baseado neste trabalho, Adam Back criou *HashCash* em 1997 (BACK, 1997) para dar solução ao problema do *spam* de *emails* e de ataques DoS (*Denial of Service*) por meio do que ele chama de funções de custo que são eficientemente verificáveis porém parametrizavelmente caras de calcular, as quais usam uma ficha (*token*) e funcionam como uma prova de trabalho (*Proof-of-Work*). Neste esquema são definidos dois papéis: um de cliente e outro de servidor. O cliente calcula uma ficha por meio de uma função MINT(), que cunha fichas para o protocolo. O servidor checa o valor das fichas usando uma função de avaliação VALUE() e só continua com o protocolo se a ficha tiver um valor requerido. Usando a implementação e os conceitos aplicados em *HashCash*, Satoshi Nakamoto (NAKAMOTO, 2009) propôs em 2009 sua própria versão de PoW para o uso na criptomoeda *Bitcoin*, mecanismo que também é chamado de consenso de Nakamoto.

Segundo Gramoli (GRAMOLI, 2017) o processo de criação do bloco consiste em os participantes do protocolo da *blockchain* (chamados de mineradores) tentarem dar solução ao desafio criptográfico antes da criação de um novo bloco. Dado um *threshold* global (dificuldade) e o maior índice de bloco conhecido pelo minerador, este seleciona um *nonce* (número aleatório) e aplica uma função pseudoaleatória (*hash*) para os campos do cabeçalho (*header*) deste bloco, incluindo o *nonce* selecionado. Enquanto o resultado do *hash* não ficar abaixo do *threshold*, o *nonce* é trocado por outro e o *hash* recalculado. Se a condição for satisfeita antes que o minerador receba um novo bloco da rede, ele cria um bloco que contém o *nonce* bem sucedido como um PoW, bem como o *hash* do bloco anterior, *hash* das transações incluídas no bloco, entre outras informações, e o transmite para a rede que mantém a *blockchain*.

Como não existe estratégia conhecida para dar solução fácil ao desafio criptográfico, os mineradores simplesmente continuam testando por força bruta se os números escolhidos aleatoriamente (*nonces*) resolvem o desafio definido abaixo.

$$hash\_de\_prova < threshold \tag{3.1}$$

A dificuldade deste desafio, definido pelo *threshold*, limita a taxa com que os novos blocos podem ser gerados pela rede. A dificuldade existe para garantir um intervalo de tempo entre criações de blocos consecutivos aproximadamente constante. No caso do *Bitcoin*, tal intervalo é de 10 minutos em média. Além disso, esta dificuldade permite que o sistema seja mais robusto e estável. O criador do bloco recebe uma recompensa dada pelo protocolo (ROSIC, 2017) na forma de novas criptomoedas criadas especialmente para este fim.

À medida que alguns mineradores aumentam seu poder computacional a dificuldade para minerar um bloco também deve aumentar. Esse fato permite (no caso do *Bitcoin*) que o tempo médio entre criações de blocos permaneça aproximadamente estável em torno de 10 minutos. Alguns mineradores com maior recurso computacional, utilizam hardwares dedicados para mineração alcançando com isso alto desempenho e alta taxa de sucesso no desafio.

Um dos problemas gerados por este mecanismo de consenso é o forte incentivo a se usar um grande poder computacional, o que resultou em uma corrida por maior desempenho e na concentração do poder de mineração naqueles que controlam uma grande quantidade de equipamentos capazes de calcular *hashes* em paralelo. Isto traz como resultado o consumo excessivo de energia elétrica, especialmente considerando que todos os que tentaram simultaneamente resolver o problema e não conseguiram vencer, perderam toda o esforço e energia gastos na tentativa de criar o próximo bloco.

Como pode ser observado na Fig. 4 o controle do poder computacional está concentrado em poucas empresas. Nota-se que mais de 75% do poder computacional está sob o controle de apenas 6 grupos.

Sabe-se que a maneira de fraudar (pelo menos teoricamente) uma *blockchain* baseada em PoW é ter o controle de mais de 50% de todo o poder computacional da rede. Assim é possível fazer com que a maioria dos cálculos sejam a favor de um *fork* que contenha uma transação fraudulenta, criando um falso consenso em torno dela.

Como já mencionado, um outro problema que se relaciona com o PoW é o das ramificações (bifurcações ou *forks*). Dois nós mineradores podem achar dois blocos válidos a partir de um mesmo bloco pai de forma quase simultânea como mostra a Fig. 5.

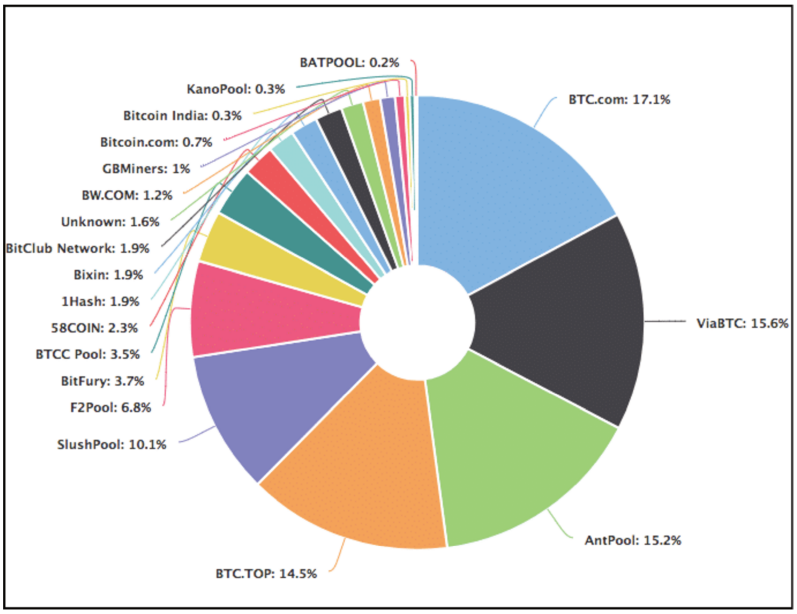


Figura 4 – Controle de poder de computação pelos grupos de mineração (BLOCKGEEKS, 2018)

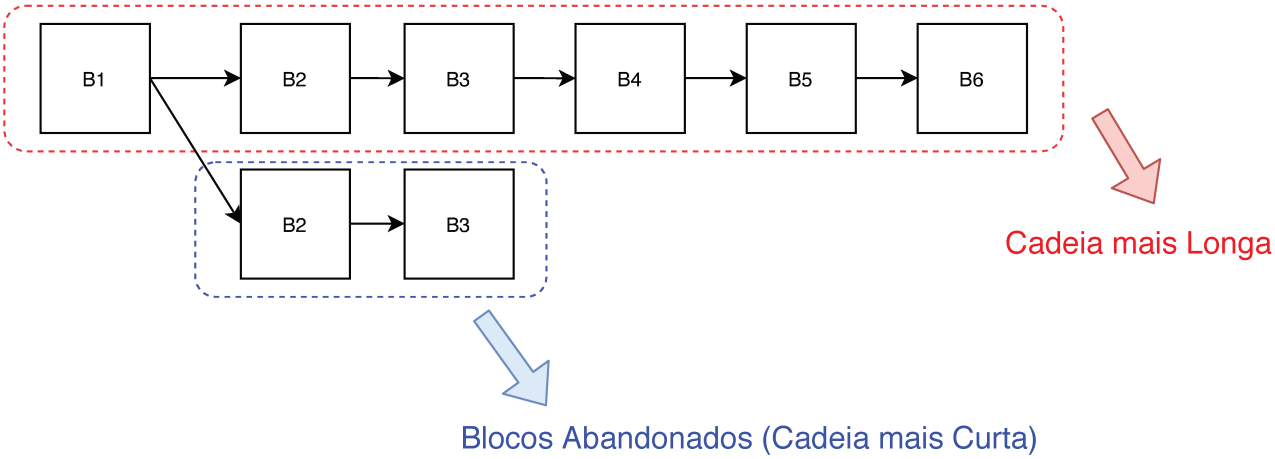


Figura 5 – Ramificação no *Bitcoin*

Tal ramificação é resolvida e não se mantém, pois há regras claras que determinam que a cadeia de blocos mais longa é a que será aceita e usada pelos demais nós, devendo a outra (mais curta) ser totalmente abandonada. Isso acaba resultando em outra fonte de desperdício de energia e tempo, uma vez que todo o trabalho que vários nós da rede estejam fazendo ao mesmo tempo para criar um novo bloco para o ramo mais curto (por acharem que este seria o mais longo) pode ser perdido no momento que receberem outros blocos da cadeia mais longa e perceberem que estavam se esforçando na cadeia de blocos errada.

As principais moedas que utilizam este mecanismo de consenso são: *Bitcoin* (NAKAMOTO, 2009), Litecoin (COBLEE, 2011), *Primecoin* (KING, 2013), *Ether* (*blockchain Ethereum*) (BUTERIN, 2013).

3.4.2 *Proof-of-Stake* (PoS)

O *Proof-of-Stake* é um mecanismo de consenso em que o sistema faz uma escolha do nó que poderá criar um novo bloco por meio do sorteio de um nó líder. A probabilidade de o nó ser sorteado é baseada na quantidade de moedas que ele possui (ou coloca à disposição) como mostrado na Fig. 6.

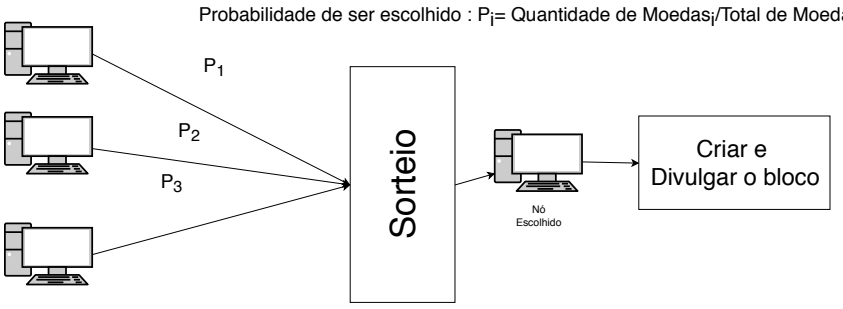


Figura 6 – Processo de criação de blocos no PoS

A ideia deste mecanismo foi baseada em uma publicação sobre *Bitcoin* escrita por Nick Szabo em 2011 (SZABO, 2011). No mesmo ano, no fórum *bitcointalk*, o usuário Quantum-Mechanic descreveu o termo *Proof-of-Stake* e outros aspectos relacionados com seu potencial (QUANTUMMECHANIC, 2011). Meses depois Sunny King e Scott Nadal submeteram a primeira implementação do mecanismo no *github* o qual seria parte da criptomoeda *Peercoin* (KING; NADAL, 2012).

Para Bano (BANO *et al.*, 2017) um tema comum entre os sistemas que utilizam este tipo de consenso é a escolha aleatória do líder a partir de outros usuários, a qual pode ser pública (o resultado é visível para todos os nós) ou privada (é usada informação privada para verificar se algum nó foi escolhido como líder).

Vasin (VASIN, 2013) menciona que, para a geração de um bloco, é necessária uma prova de que o nó possui uma certa quantidade de moedas. Gerar o bloco pode envolver o envio de moedas a si mesmo, o que prova a propriedade das mesmas. A quantidade de moedas declarada como disponível por um nó é utilizada pela rede para ajustar a dificuldade que o nó precisa vencer para criar um novo bloco. Neste cenário nós que possuem mais moedas possuem uma probabilidade maior de serem sorteados.

Assim, como no *Bitcoin*, o participante deve calcular o *hash* de prova a partir dos campos do cabeçalho do bloco. Uma vez calculado o *hash* de prova, este é comparado com o objetivo, que é o desafio pelo qual os nós devem passar para criar o próximo bloco. Caso o *hash* de prova seja menor que objetivo,temos um sucesso e o participante pode criar o bloco.

A inequação 3.2 exige que o *hash* de prova calculado anteriormente, seja menor que o objetivo.

$$hash\_de\_prova < objetivo \tag{3.2}$$

Quanto maior for o objetivo menor é a dificuldade de satisfazer a expressão 3.2, ou seja maior é a probabilidade de sucesso.

Segundo o artigo que descreve a criptomoeda *NxT*, também baseada em PoS (COMMUNITY, 2014), o valor objetivo (que neste caso define a facilidade para criação do bloco) é diferente para cada nó, pois depende do saldo em conta que este nó possui.

Cada nó tem seu próprio objetivo para o próximo bloco, o qual é chamado objetivo base, servindo de parâmetro para o protocolo tentar achar um valor de *hash* de prova menor que ele. Este objetivo base muda de bloco para bloco, tomando como referência o objetivo base do bloco anterior.

Para a definição do objetivo a ser usado para decidir quem vai criar o próximo bloco, existem duas formas conhecidas.

- **Definição baseada em idade das moedas (*coin age*):** este tipo de cálculo de objetivo permite um esquema de sorteio cuja chance de ganhar é proporcional ao número de moedas mantidas por um tempo determinado. É como se um nó que possuísse moedas durante um maior período de tempo tivesse mais chances de ganhar o sorteio. Mas se nesse tempo ele faz uso de uma certa quantidade de moedas, o novo saldo será considerado em um novo intervalo de tempo, como pode ser observado na Fig. 7.

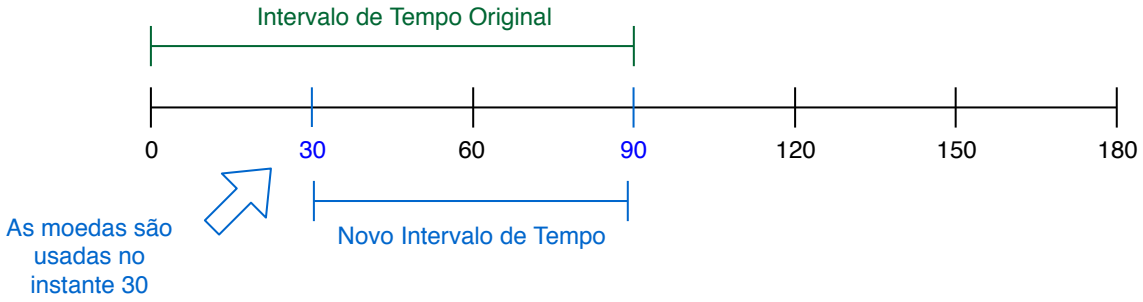


Figura 7 – Novo intervalo de tempo para definição da idade das moedas.

A expressão 3.3 mostra o cálculo do novo objetivo

$$objetivo_{novo} = objetivo_{anterior} \times tempo \times \underbrace{moedas \times idade}_{\text{idade da moeda(coin age)}} \tag{3.3}$$

tal que:

$objetivo_{novo}$: é o objetivo do bloco atual

$objetivo_{anterior}$: é o objetivo do bloco anterior

tempo : é o tempo desde a criação do último bloco, em segundos

moedas: é a quantidade de moedas que o nó possui durante um intervalo de tempo.

idade: é intervalo de tempo em que as moedas não foram usadas.

O parâmetro *tempo* é usado para aumentar o objetivo e, consequentemente, a facilidade de se alcançá-lo à medida que o tempo passa. Ou seja, quanto mais tempo se passar desde a criação do último bloco, maior será o valor de *tempo* e maior será a facilidade de se alcançar este objetivo, não permitindo assim que um tempo muito longo se passe entre a criação de blocos consecutivos. Este tipo de objetivo é usado na criptomoeda *Peercoin* (KING, 2013).

- **Definição aleatória:** Este tipo de cálculo de objetivo permite um esquema de sorteio cuja chance de ganhar é proporcional ao tamanho da riqueza que o nó possui no momento, ou seja, neste caso não existe a influência da idade das moedas, apenas a quantidade de moedas de um nó.

  A expressão 3.4, mostra o cálculo do novo objetivo

$$objetivo_{novo} = objetivo_{anterior} \times tempo \times moedas \tag{3.4}$$

  tal que:

$objetivo_{novo}$: é o objetivo do bloco atual

$objetivo_{anterior}$: é o objetivo do bloco anterior

tempo : é o tempo desde a criação do último bloco, em segundos

moedas: é a quantidade de moedas que foram adquiridas durante um intervalo de tempo.

Esta abordagem é usada nas criptomoedas *NxT* (COMMUNITY, 2014) e *BlackCoin* (VASIN, 2013).

O esquema de sorteio definido por este tipo de cálculo de objetivo pode ser dividido em modelos dinâmico e estático. No dinâmico, cada nó tem uma quantidade diferente de moedas, o que faz com que cada um tenha uma probabilidade diferente de ser escolhido. No estático, considera-se que todos têm a mesma quantidade de moedas; logo todos os nós têm a mesma probabilidade de serem sorteados (isto é, de vencer o desafio da expressão 3.2).

Algumas características do mecanismo PoS:

- não há recompensa por bloco, mas os mineradores recebem as taxas das transações incluídas no bloco criado (GREENFIELDIV, 2017);
- há um desafio em rastrear de forma confiável as mudanças de saldo de cada participante (BANO *et al.*, 2017);
- este mecanismo diminuiu o gasto de energia em comparação com PoW já que não incentiva e não depende de poder computacional elevado;
- existe um problema conhecido como *"Nothing-at-Stake"*, onde os mineradores não têm nada a perder ao minerar em dois ou mais ramos (*forks*) ao mesmo tempo, com o objetivo de recolher mais taxas. Este problema é contornável, já que as taxas serão devolvidas quando ramos forem cancelados. Entretanto, esta facilidade em criar blocos para ramos secundários não é benéfica;
- é mais caro realizar uma fraude em PoS que em PoW, já que, para tentar possuir 51% do poder de decisão, o atacante precisará conseguir 51% das moedas, o que irá gerar uma demanda pelas mesmas, causando imediatamente uma valorização das moedas e dificultando a obtenção das que faltam;
- permite aumentar uma eventual concentração de riquezas, já que quem possui mais moedas acaba tendo mais chances de ser escolhido para criar novos blocos e ganhar as taxas por isso, ficando ainda mais rico.

As principais moedas que utilizam este mecanismo de consenso são *NxT* (COMMUNITY, 2014), *BlackCoin* (VASIN, 2013) , *PeerCoin* (KING, 2013), *Ethereum* (*Slasher* (BUTERIN, 2014) & *Casper* (BUTERIN; GRIFFITH, 2017)) e *Cardano* (*Ouroboros*)(KIAYIAS *et al.*, 2017) .

3.4.3 *Delegated-Proof-of-Stake* (DPoS)

É um mecanismo de consenso derivado do PoS, mas com a diferença de que neste alguns nós escolhem um grupo de outros nós para gerar e validar os blocos. A quantidade de nós que validam é significativamente menor e, consequentemente, os blocos podem ser criados mais rapidamente se as transações forem validadas da mesma forma. Parâmetros como tamanho do bloco e intervalo entre criação de blocos podem ser ajustados.

O DPoS foi proposto por Daniel Larimer para sua moeda *Bitshare* em 2014 (LARIMER, 2014), onde se alega que o sistema pode ser visto como uma democracia representativa. No artigo (BITSHARE, 2016) o autor divide o grupo de nós escolhidos em testemunhas (*witness*), que têm o papel de validar assinaturas e *timestamps* das transações e delegados, que têm o papel de propor mudanças nos parâmetros da rede.

Os usuários podem escolher qualquer uma das testemunhas disponíveis para gerar blocos. Cada nó tem direito a um voto por testemunha e este processo é chamado de "votação de aprovação". O processo de votação é realizado por meio do compartilhamento do voto com os outros nós da rede. O número de testemunhas N é definido tal que pelo menos 50% dos usuários votantes acreditem que ele é suficiente para a descentralização. O usuário não pode

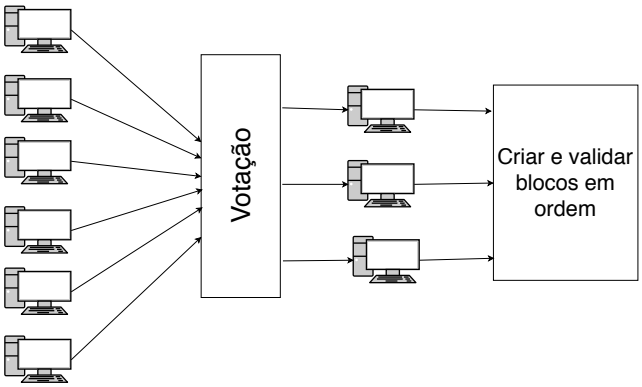


Figura 8 – Processo de criação de blocos no DPoS

votar em um número de testemunhas maior do que o que foi definido anteriormente. Cada testemunha recebe voto de algum nó para assinar os blocos (BITSHARE, 2016) (Fig. 8).

O voto de um nó tem um poder que depende da quantidade de moedas que este possui (NAQVI, 2017). Portanto, os nós com mais moedas têm mais possibilidades de ganhar. Ao

final, as testemunhas escolhidas serão as N que possuem mais votos.

A cada certo período tempo as testemunhas produzem um bloco, sendo pagas para isso. A taxa de pagamento é estabelecida pelos nós por meio de seus delegados escolhidos. Se a testemunha falha em produzir um bloco então ela não é paga e pode ser eliminada no futuro.

O quadro de testemunhas ativas é atualizado uma vez a cada intervalo de manutenção (um dia), quando os votos são contabilizados. As testemunhas são embaralhadas e a cada uma delas é dado um turno para produzir um bloco num horário fixo de dois segundos. Depois disto, é dado outro turno a todas as testemunhas e elas são embaralhadas de novo. Se a testemunha não produz o bloco em seu intervalo de tempo, seu turno é pulado e a próxima testemunha é quem produz o bloco. Segundo o artigo em (BITSHARETEAM, 2016), os turnos serão gerenciados por meio de um algoritmo de agendamento *Round-Robin*.

Os delegados são escolhidos de forma similar às testemunhas. Eles se tornam co-assinantes em uma conta especial, conhecida como a conta gênesis, que tem o privilégio de propor mudanças nos parâmetros da rede. Estes parâmetros são: taxas de transação, tamanhos de bloco, pagamento de testemunha e intervalos de bloco. Depois que a maioria de delegados aprova as mudanças propostas, os usuários têm duas semanas para revisar as mesmas, período no qual eles podem eliminar delegados e anular as mudanças propostas.

Pelo acima exposto, pode-se dizer que os usuários não precisam se preocupar com delegados desonestos, pois estes podem ser eliminados facilmente. O projeto se assegura de que os delegados não possuam poder direto e que todas as mudanças dos parâmetros da rede sejam aprovadas em última instância pelos usuários. Diferentemente das testemunhas, os delegados não recebem nenhum pagamento. Além disso, os parâmetros citados não mudam frequentemente.

Este mecanismo oferece uma escolha mais democrática por meio de eleições, o que fornece uma solução ao problema de PoS favorecer a concentração de riqueza. Ele ainda apresenta uma certa injustiça na eleição de representantes (seja para testemunhas ou delegados), o que é demonstrado pelo fato de que um voto para um candidato a representante com muitas moedas tem mais valor que para um candidato a representante com poucas moedas.

As principais moedas que utilizam este mecanismo de consenso são *Bitshare* (LARIMER, 2014), *Steem* (SCOTT, 2017), *ARK* (ARKCREW, 2018), *EOS* (BLOCK.ONE, 2018), *Lisk* (KORDEK; BEDDOWS, 2016).

3.4.4 *Proof-of-Burn* (PoB)

É uma alternativa ao mecanismos de consenso PoW e PoS. A ideia deste mecanismo consiste em que cada nó deve investir (ou queimar) alguma quantidade de moedas para ter alguma probabilidade de ser o criador do bloco: quem queimar mais moedas tem a maior probabilidade de ganhar (Fig. 9).

Foi proposto por P4Titan para a moeda *Slimcoin* em 2014 (P4TITAN, 2014). Sua estrutura está baseada na moeda *Peercoin* (KING; NADAL, 2012) como solução às inconveniências de PoW, como o uso excessivo de recursos computacionais, por exemplo.

O papel de queimar moedas é servir de prova no momento da mineração por PoB. Este conceito de queimar consiste em enviar moedas a um endereço especial chamado endereço de queima. Quando as moedas são enviadas para lá, estas se perdem para sempre, já que não há mecanismo que permita utilizar uma moeda atrelada a este endereço.

No momento de queimar moedas, uma transação é feita para o endereço de queima. O *hash* desta transação é gravado de forma separada de outras transações, para indicar a transação que contém moedas queimadas.

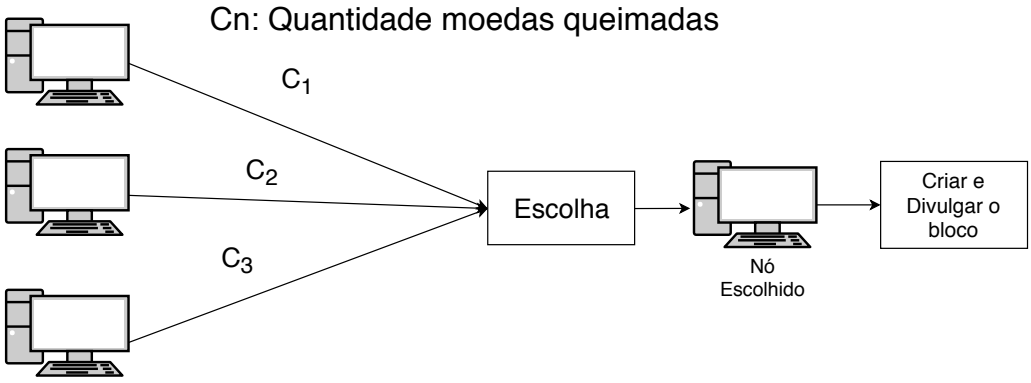


Figura 9 – Processo de criação de blocos no PoB

O bloco vai incluir as coordenadas da transação de queima do respectivo *hash* de queima. Estas coordenadas dão o ponto exato onde a transação de queima pode ser encontrada na *blockchain*. Estas são usadas para procurar a transação de queima durante o processo de verificação. A assinatura do bloco deve bater com a assinatura da transação de queima apontada pelo bloco. Isto vai prevenir que um mesmo bloco possa ser copiado e tenha a assinatura mudada e usada por um atacante.

Este mecanismo soluciona o problema de alto consumo de energia e poder computacional do PoW e tenta solucionar o problema de concentração de riqueza de PoS por meio do processo de queima de moedas. Entretanto, PoB não resolve completamente o problema de PoS, já que no processo de eleição do criador do bloco, o candidato precisa possuir uma

quantidade maior de moedas para ser escolhido, implicando que somente quem possa queimar mais moedas terá mais chances de criação do blocos.

A principal moeda que utiliza este mecanismo de consenso é *Slimcoin* (P4TITAN, 2014).

3.4.5 *Proof-of-Activity* (PoA)

É um mecanismo de consenso híbrido que mistura PoW e PoS aproveitando seus benefícios. Foi proposto por Iddo Bentov, Charles Lee, Alex Mizrahi e Meni Rosenfeld em 2014 (BENTOV *et al.*, 2014) e determina que o bloco minerado precisa ser assinado por N mineradores para ser válido. Neste sentido, mesmo que algum dos nós possua mais de 50% de todas as moedas, ele não conseguirá controlar a criação de novos blocos por conta própria (ZHENG *et al.*, 2017a).

O termo "*activity*" enfatiza o ponto que somente os nós ativos (os que mantêm um nó completo online, isto é, um nó totalmente sincronizado com os demais) são recompensados pelos serviços vitais que eles proveem para a rede.

A sub-rotina principal é chamada *follow-the-satoshi* que consiste em converter alguns valores pseudoaleatórios em *satoshis* (10^{-8} *bitcoins*) os quais são tomados de maneira uniforme de todos os *satoshis* que foram gerados até o momento. Isto é feito por seleção de um índice pseudoaleatório entre zero e o total de *satoshis* criados até o último bloco, examinando o bloco no qual o *satoshi* indicado pelo índice foi gerado e seguindo as transações que o transferiram para endereços subsequentes até chegar ao endereço que atualmente o controla. Segundo Bentov et al. (BENTOV *et al.*, 2014) o processo de criação do bloco é o seguinte (Figs. 10 e 11):

- primeiro os nós tentam gerar um cabeçalho de bloco vazio;
- depois que algum nó crie o cabeçalho, se escolhe N nós pseudoaleatórios utilizando a função *follow-the-satoshi*, o que permitirá fazer um sorteio com uma probabilidade baseada na quantidade de *satoshis* que o nó possui;
- depois que os $N - 1$ primeiros nós assinam o *hash* do cabeçalho do bloco e enviam a assinatura a toda a rede, o N-ésimo nó cria o bloco em questão, estendendo este cabeçalho de bloco vazio com algumas transações, as N-1 assinaturas e sua própria assinatura para o *hash* de todo o bloco;
- este N-ésimo nó envia o bloco criado à rede para que seja validado e seja considerado uma extensão legítima da *blockchain*.

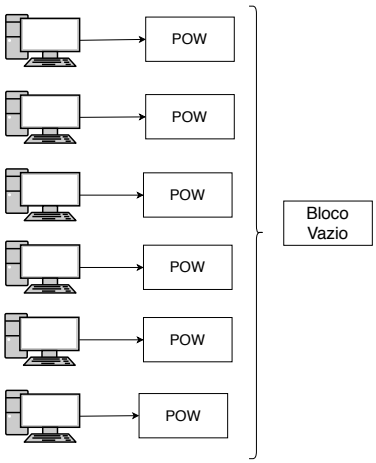


Figura 10 – Processo de criação de blocos no PoA parte 1

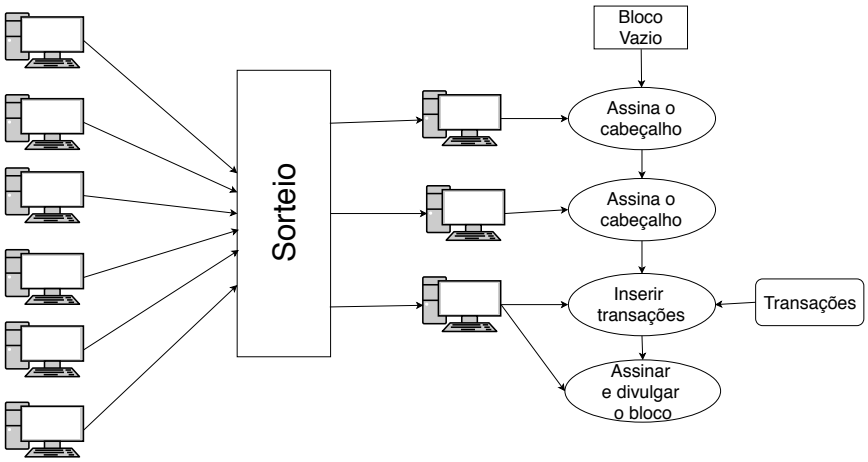


Figura 11 – Processo de criação de blocos no PoA parte 2

As taxas de transações que o N-ésimo nó coleta são compartilhadas entre o minerador e os outros $N-1$ nós que foram sorteados. Se algum dos N nós sorteados estiver *offline*, então outros mineradores poderão se envolver na criação do bloco e desse modo derivar outros N nós pseudoaleatórios. Assim, a dificuldade seria reajustada tanto pelo seu valor de *hash* como pela fração de todos os nós online.

O processo de criação do bloco é realizado em duas rodadas de comunicação. Na verdade, se estabelecermos que $N = 1$, o processo vai requerer somente uma rodada de comunicação.

O componente PoW do protocolo é essencial para diminuir as tentativas de escolha de nós sortudos. Sem ele as derivações acontecem a grandes velocidades. Um problema subjacente é que, sem o componente PoW, não há forma clara de fazer o sistema convergir. Outro problema é que os nós racionais podem maximizar suas expectativa de recompensa, assinando cada ramo bifurcado que seja encontrado, desde que PoW não seja requerido para

estender os ramos.

O mecanismo PoA é uma solução para o problema de tentativa de fraude do sistema, por meio da regulação moral da tragédia dos comuns. Esta consiste em que, se um nó age mal, ele prejudica todos incluindo ele mesmo. Isso assegura que se tenha baixa probabilidade de se criar ramificações. Esta regulação não assegura que o mecanismo seja totalmente justo, mas somente que os usuários tenderão a agir de forma certa.

Por outro lado, por ser híbrido, sabe-se que possui duas partes, uma PoW e outra PoS, cada parte com seus problemas vindos dos mecanismos puros. Em outras palavras, se algum usuário quer participar do processo de criação de blocos precisa de poder computacional para realizar a proposta do cabeçalho, que corresponde à parte de PoW, e precisa de moedas para ter alguma probabilidade de ser escolhido por meio da função *follow-the-satoshi* para ser testemunha sorteada, que faz parte de PoS. Assim, a parte PoS tenta ser a mais justa possível, mas o processo em geral continua sendo injusto.

A principal moeda que utiliza este mecanismo de consenso é *Decred* (JEPSON, 2015).

3.4.6 *Algorand*

É uma criptomoeda proposta por Jing Chen e Silvio Micali em 2017 (CHEN; MICALI, 2017). Esta nova criptomoeda está projetada para que tenha uma confirmação de transação em 1 minuto.

O núcleo do *Algorand* é um protocolo de acordo bizantino chamado BA* que foi proposto pouco tempo antes por Silvio Micali (MICALI, 2017). Este mecanismo escala para muitos usuários, permitindo atingir o consenso em um novo bloco com baixa latência e baixa probabilidade de *forks*.

O protocolo *Algorand* atribui pesos aos usuários com base na quantidade de moedas em suas contas e obtém escalabilidade pela eleição de um pequeno conjunto de usuários representativos escolhidos de forma aleatória e que irão formar um comitê.

O algoritmo soluciona três desafios: o primeiro consiste em evitar o ataque *Sybil*, onde um usuário influencia no protocolo de acordo bizantino; o segundo consiste em BA* ser escalável a milhões de usuários, o que é, de longe, muito mais do que conseguem fazer os protocolos de acordo bizantino no estado da arte; o terceiro consiste em ser resiliente aos ataques DoS e continuar funcionando mesmo que o adversário desconecte alguns usuários.

O processo de criação do bloco começa com os usuários submetendo as transações por meio do protocolo *gossip* (boato). Cada usuário coleta um conjunto de transações pendentes que ele pode ter acesso e então se escolhe quem vai propor o bloco em cada rodada por meio

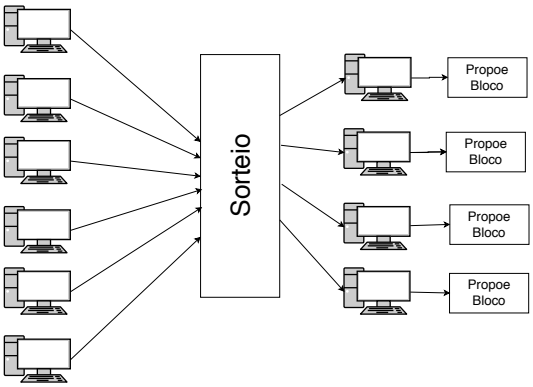


Figura 12 – Processo simplificado de criação de blocos no *Algorand* parte 1

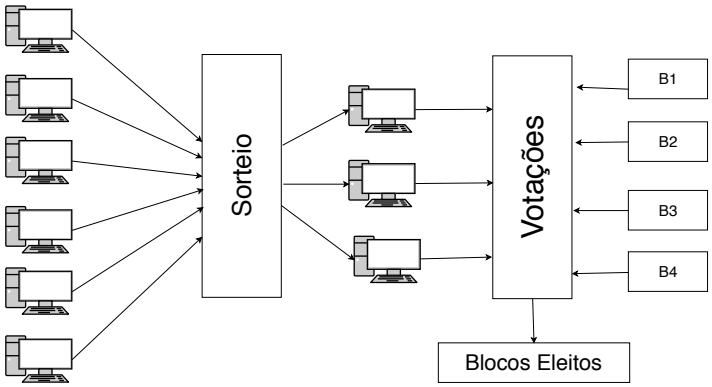


Figura 13 – Processo simplificado de criação de blocos no *Algorand* parte 2

de uma função de sorteio criptográfico (Figs. 12 e 13). O sorteio será influenciado pelo peso (quantidade de moedas), prioridade e uma prova da prioridade. Como o sorteio é um processo aleatório, pode haver múltiplos usuários propondo um bloco e a prioridade determina que bloco todos deveriam adotar. Entretanto, é difícil saber se todos os usuários receberam este mesmo bloco.

Para chegar a um consenso sobre um bloco, usa-se o BA* tal que cada usuário inicia com o bloco de mais alta prioridade que ele recebeu. Cada rodada começa com um sorteio criptográfico para checar os escolhidos para fazer parte do comitê. Em seguida, cada membro do comitê envia uma mensagem com uma prova de sua seleção por meio do *gossip*. Estas rodadas são repetidas até que, em alguma rodada, suficientes membros do comitê atinjam o consenso mediante voto.

Se um usuário alcança o consenso final, isto significa que algum outro usuário que chegar ao consenso final ou tentativo na mesma rodada deve aceitar o mesmo valor do bloco. Isto vai prover segurança já que as futuras transações estarão encadeadas neste bloco final.

Outros usuários podem atingir um consenso tentativo num bloco diferente (enquanto

nenhum usuário atinja um consenso final). Um usuário só confirmará a transação de um bloco tentativo se o bloco sucessor atingir o consenso final.

Este mecanismo de consenso propõe uma maior justiça (ou democracia) na participação dos usuários por meio da introdução do conceito de prioridade. Esta prioridade vai permitir ao usuário que tenha a menor participação, ter a maior possibilidade de ser parte do comitê em uma rodada já que, se não chegar a um consenso, o comitê é renovado. Além disso, a escolha tanto dos possíveis proponentes de blocos como dos membros do comitê é por meio do processo de sorteio aleatório, o que evita o problema de exigência de alto poder computacional de PoW. Em certa medida, o problema da concentração de riqueza de PoS ainda continua, dado o fato de se precisar de moedas para os cálculos.

3.5 Comparação e discussão de objetivos

Como os diferentes mecanismos de consenso aqui analisados têm várias particularidades, similaridades e diferenças, decidimos compilar uma tabela que destaca as principais características de cada um deles e permite uma comparação mais direta entre seus pontos fortes e fracos. Para facilitar a visualização, esta tabela está dividida em três partes e delas (Tabs. 1,2 e 3) se pode extrair algumas informações relevantes.

O consenso dos mecanismos DPoS, PoA e *Algorand* é obtido de forma similar, já que é escolhido um grupo de usuários (ou nós), com a diferença de que DPoS faz uma votação direta e PoA e Algorand possuem uma escolha aleatória baseada na quantidade de moedas que determinado nó possui.

Pode ser notado ainda que *Algorand* tem uma tolerância bizantina de 1/3 de nós falhos ou maliciosos, enquanto o restante dos mecanismos trabalha com uma tolerância de 50%.

Muitos dos mecanismos observados estão atrelados ao uso de moedas para chegar a um consenso. Isto é feito para reforçar o conceito de pesos diferentes para usuários diferentes nos processos de escolha baseados em PoS ou similares.

As características que dificultam a fraude são muito variadas. Entre as mais interessantes estão a tragédia dos comuns, que é um principio moral que governa o mecanismo PoA, tentando manter o bem comum entre os usuários.

Outro ponto interessante é o sorteio criptográfico usado na moeda *Algorand* como forma de gerar uma aleatoriedade durante todo o processo de criação de blocos.

Por último o conceito de consenso tentativo que também é usado no *Algorand*. Junto

com a renovação de membros, este conceito torna difícil que um adversário corrompa a rede.

Depois de uma revisão dos mecanismos de consenso descritos e comparados neste trabalho, acreditamos que uma proposta de mecanismo de consenso mais justo poderia ser baseada em *Algorand*, já que sua estrutura e forma de trabalhar oferece mais justiça (democracia) na participação dos usuários.

Uma possível proposta de melhoria seria procurar evitar o gasto de moedas ou diminuir seu impacto no processo de consenso. Uma outra forma de se buscar um melhor mecanismo de consenso poderia ser aproveitando algumas caraterísticas de outros mecanismos estudados, como de DPoS, onde os usuários têm decisão sobre o conjunto de testemunhas. Também poderíamos tentar aproveitar e melhorar o conceito de pagamento às testemunhas como ocorre em DPoS e PoA.

Tabela 1 – Comparação dos mecanismos de consenso: parte 1

MECANISMOS DE CONSENSO	**Proof-of-Work (PoW)**	**Proof-of-Stake (PoS)**	**Algorand**	**Delegated-Proof-of-Stake (DPoS)**	**Proof-of-Burn (PoB)**	**Proof-of-Activity (PoA)**
Tipo de Mecanismo	Puro	Puro	Puro	Puro	Puro	Hibrido
Criadores e/ou Precursores	1. Cynthia Dwork e Moni Naor 2. Adam Back (Hashcash) 3. Satoshi Nakamoto (Bitcoin)	1. Nick Szabo (Ideia sobre o mecanismo) 2. Quantum Mechanic (Primeiro que descreveu o mecanismo) 3. Sunny King e Scott Nadal (Primeira moeda)	1. Silvio Micali e Jing Chen (Teoria) 2. Yossi Giland, Rottem Hemo, Giorgio Vlachos e Nickolai Zeldovich (Implementação)	Daniel Larimer	P4Titan	Iddo Bentov, Charles Lee, Alex Mizrahi e Meni Rosenfeld
Consenso	Verificação de que algum minerador chegou a uma solução	Verificação do ganhador do sorteio baseado na quantidade de moedas	Verificação do usuário que propõe o bloco Verificação dos membros do comitê em cada rodada Votação para concordar com o *hash* do bloco proposto	Processo de seleção de testemunhas para a criação dos blocos e delegados	Verificação da quantidade de moedas queimadas na transação de queima pelo minerador	Verificação do cabeçalho do bloco vazio Verificação dos nós ganhadores do sorteio Verificação se é uma extensão legítima do blockchain
Tolerância a Falhas	$\leq$ 50% do poder de mineração	$\leq$ 50% da posse da quantidade de moedas	$\leq$ 33,33% da quantidade de usuários maliciosos	$\leq$ 50% da quantidade de testemunhas maliciosas	$\leq$ 50% da quantidade de moedas queimadas	$\leq$ 50% dos nós maliciosos online

3.6 Conclusões

Da comparação entre os principais mecanismos de consenso existentes foi possível avaliar melhor os pontos fortes e fracos de cada um, o que nos trouxe algumas diretrizes sobre como estruturar um novo mecanismo de consenso que permita a criação de um novo

Tabela 2 – Comparação dos mecanismos de consenso: parte 2

MECANISMOS DE CONSENSO	Proof-of-Work (PoW)	Proof-of-Stake (PoS)	Algorand	Delegated-Proof-of-Stake (DPoS)	Proof-of-Burn (PoB)	Proof-of-Activity (PoA)
Produz Ramificações?	Sim (−)	Sim (−)	Sim (−)	Não (+)	Não (+)	Sim (−)
Recompensa por criação do bloco	Sim (+)	Sim (+)	Não (−)	Não (−)	Sim (+)	Sim (+)
Precisa de moedas para crescer?	Não (+)	Sim (−)	Sim (−)	Sim (−)	Sim (−)	Sim (−)
Poder Computacional e Consumo Energético (relativo)	Alto (−)	Baixo (+)	Baixo (+)	Baixo (+)	Baixo (+)	Médio (−)
Quem cria o bloco?	Mineradores	O usuário escolhido no sorteio	O usuário escolhido pelo sorteio que propõe com a maior prioridade	As testemunhas escolhidas por votos	O usuário que queima mais moedas	A última testemunha aleatória
Características que dificultam a fraude	Computação Intensiva	Valorização das moedas pela demanda	Sorteio Criptográfico Renovação dos membros do comitê Janela de tempo de consenso (1 min) Consenso Tentativo	Embaralhamento das testemunhas Vigilância	Investimento Contínuo	Tragédia dos comuns (se você age mal, prejudica a todos incluindo você mesmo)

modelo de *blockchain* que seja menos exigente em termos de recursos materiais e energéticos, mantendo os mesmos níveis de segurança existentes nos sistemas atuais.

Como resultado final deste esforço, pretendemos chegar a um *blockchain* mais distribuído, isto é, com participação mais ativa e com maior equidade de condições para todos os usuários interessados, o que certamente trará mais segurança (será mais difícil algum grupo ultrapassar os 50% do poder computacional da rede) e mais estabilidade (será preciso remover uma quantidade maior de nós da rede para interrompê-la) para as aplicações baseadas neste tipo de *blockchain*.

Tabela 3 – Comparação dos mecanismos de consenso: parte 3

MECANISMOS DE CONSENSO	**Proof-of-Work (PoW)**	**Proof-of-Stake (PoS)**	**Algorand**	**Delegated-Proof-of-Stake (DPoS)**	**Proof-of-Burn (PoB)**	**Proof-of-Activity (PoA)**
Vantagens	• Adoção e aceitação mais difundidas • Segurança pela dificuldade de fraude • Simplicidade	• Baixo uso de recursos de software e hardware • È mais rápido que PoW • É mais complexo que PoW	• Possui uma boa probabilidade de que todos os usuários concordem nas mesmas transações • Evita concentração de riqueza • Mais seguro que PoS	• É mais rápido que PoW e PoS • mesmas vantagens que PoS • Tem mais participação dos nós já que eles decidem sobre as testemunhas e delegados	• Mesmas vantagens que PoS • É difícil de se fraudar já que tentar obter o controle da rede o tempo todo se torna cada vez mais caro	• Tenta ser o mais justo possível na repartição das taxas e na escolha dos nós escolhidos • Apenas nós onlines • Não fica dependente dos nós offline
Desvantagens	• Consumo energético ineficiente • Concentração de mineração	• Problema de concentração de riqueza	• Originalmente sem incentivos • Pode gerar blocos vazios • Mais complexo que PoS	• Muito dependente da participação dos votantes • Mais Complexo que PoS	• Obter um ganho ou recuperar o investimento é um processo muito lento e precisa de constância no investimento (queima)	• Consumo energético ineficiente • Concentração de mineração • Mais complexo que PoS e PoW

4 Mecanismo de Consenso *Casper*

4.1 Introdução

Neste capítulo iremos avaliar mais detalhadamente o mecanismo de consenso baseado em PoS chamado Casper. Trata-se de um dos protocolo PoS mais estudados, já que é a proposta para suceder o mecanismo de consenso atual baseado em PoW (*EtHash*) da criptomoeda *Ethereum* (a segunda mais usada atualmente). Ele oferece soluções a alguns problemas do mecanismo PoW.

Como descreve (BUTERIN; GRIFFITH, 2017), a ideia principal do *Casper* é reduzir o problema de um nó conseguir trabalhar em múltiplos *forks*. O protocolo obriga o nó a seguir a cadeia mais longa, sob pena de receber punições.

Em (BUTERIN, 2014) menciona-se que o protocolo PoS está baseado em algumas ideias de *Peercoin* (KING; NADAL, 2012), onde destaca-se a definição de *checkpoints*, uma forma de proteção contra mudanças nos blocos anteriores a um *checkpoint*.

Neste algoritmo a cada 100 blocos são fixados *checkpoints*, onde a maioria dos participantes vota entre dois *checkpoints* criando um enlace majoritário entre dois *checkpoints* consecutivos. Desde que dois terços dos participantes sejam honestos é possível saber sempre qual cadeia seguir. Além disso, o protocolo garante que é possível identificar e punir com perda de moedas os participantes que não atendam as regras estabelecidas.

4.2 Conceitos básicos

4.2.1 *Checkpoints*, validadores e votos

O protocolo não lida com uma árvore de blocos completa. Por motivos de eficiência ele considera uma subárvore. O bloco gênesis é um *checkpoint* e todo bloco que tenha altura múltipla de 100, também é um *checkpoint*. A altura de um *checkpoint* cujo bloco correspondente é $100 * k$ é simplesmente k. Em outras palavras, a altura $h(c)$ de um *checkpoint* c é o número de *checkpoints* na cadeia de c até a raiz (bloco $gênesis$) como se pode observar na Fig. 14, onde os retângulos e o triângulo representam *checkpoints*. As setas da árvore mostram um enlace do *checkpoint* com seu *checkpoint* pai.

Cada validador tem que fazer um depósito quando se une à rede, o qual pode crescer ou diminuir com punições ou recompensas.

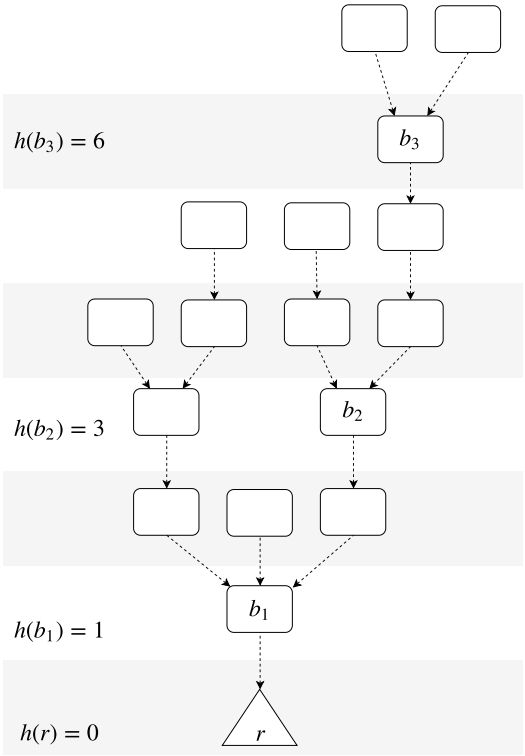


Figura 14 – Árvore de *checkpoints*

A segurança do PoS deriva do tamanho dos depósitos e não do número de validadores. Os validadores podem difundir uma mensagem de voto contendo quatro informações: dois *checkpoints* s e t e suas alturas $h(s)$ e $h(t)$. Este s deve ser o antecessor de t na árvore de *checkpoints*; caso contrário o voto é inválido. Se a chave pública do validador v não está dentro do conjunto de validadores, o voto é inválido juntamente com a assinatura do validador. Os parâmetros que compõem o voto são:

s **:** *hash* de qualquer *checkpoint* justificado (a fonte)

t **:** hash de um *checkpoint* descendente de s

h(s) **:** altura do *checkpoint* s na árvore de *checkpoints*

h(t) **:** altura do *checkpoint* t na árvore de *checkpoints*

S **:** assinatura de $\langle s, t, h(s), h(t)\rangle$ com a chave privada do validador v

'VOTE' **:** palavra que indica que é uma mensagem de voto

O formato da mensagem é o seguinte $['VOTE', \langle v, s, t, h(s), h(t)\rangle, S]$

O conjunto de validadores deve mudar e, para isso, novos validadores devem estar habilitados para se unirem aos validadores já existentes.

4.2.2 Enlace Majoritário

Um enlace majoritário é um par ordenado de *checkpoints* (a, b), também escrito como $a \rightarrow b$, tal que pelo menos 2/3 dos validadores tenham publicado votos no *checkpoint* com fonte em a e destino em b. O enlace majoritário pode pular *checkpoints*, ou seja, a ligação pode ocorrer entre *checkpoints* não sucessivos.

4.2.3 *Checkpoints* Conflitantes, Justificados e Finalizados

Dois *checkpoints* a e b são chamados conflitantes se, e apenas se, são de diferentes ramos.

Um *checkpoint* c é chamado de justificado se:

1. este é a raiz da árvore

   ou

2. existe um enlace majoritário $c' \rightarrow c$ onde c' é justificado (Fig. 15).

Um *checkpoint* c é chamado finalizado se este é justificado e tem um enlace majoritário $c' \rightarrow c$ onde c' é um filho direto de c.

Se dois *checkpoints* a e b são conflitantes, ambos não podem ser finalizados.

Os enlaces majoritários podem sempre produzir novos *checkpoints* finalizados, desde que existam novos *checkpoints* filhos que estendam a cadeia finalizada.

4.3 Propriedades principais

Condições de Slashing: um validador individual v não deve publicar dois votos,

$$\langle v, s_1, t_1, h(s_1), h(t_1) \rangle$$

e

$$\langle v, s_2, t_2, h(s_2), h(t_2) \rangle,$$

tais que em ambos:

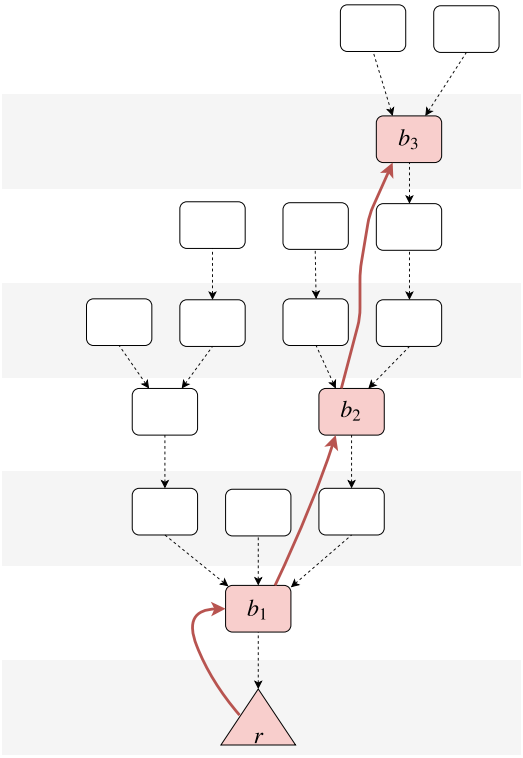


Figura 15 – Cadeia Justificada $r \rightarrow b_1 \rightarrow b_2 \rightarrow b_3$

I. $h(t_1) = h(t_2)$
(ou seja, um validador não deve publicar dois votos diferentes para uma mesma altura de *checkpoint*)

ou

II. $h(s_1) < h(s_2) < h(t_2) < h(t_1)$
(ou seja, um validador não deve criar um voto cujos *checkpoints* estejam dentro de um intervalo dos *checkpoints* do outro voto).

Uma propriedade notável deste protocolo é que é impossível que dois *checkpoints* conflitantes sejam finalizados sem que 1/3 dos validadores violem ao menos uma condição de *slashing*.

Se um validador viola ambas as condições de *slashing*, a evidência de violação pode ser incluída dentro da *blockchain* como uma transação. Neste ponto, todo o depósito do validador é confiscado como taxa de penalização.

Supondo que 2/3 dos validadores não violam as condições acima, as seguintes propriedades também podem ser derivadas:

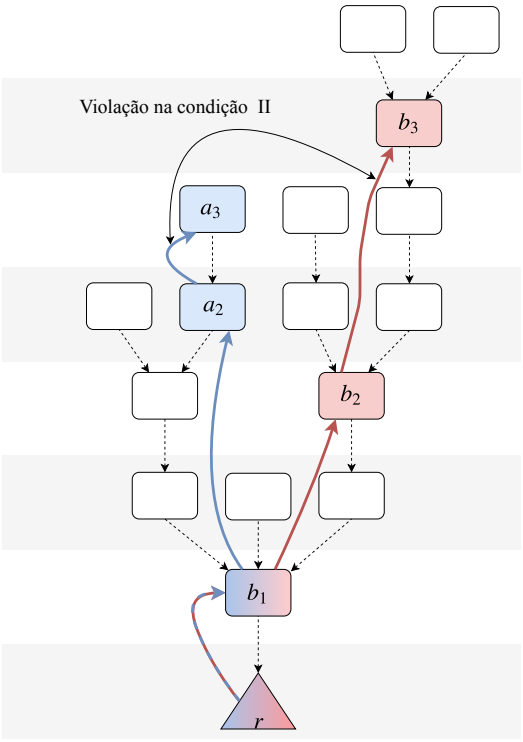


Figura 16 – Propriedade (ii) do mecanismo de consenso *Casper*

(i) Se $s_1 \rightarrow t_1$ e $s_2 \rightarrow t_2$ são enlaces majoritários distintos , então $h(t_1) \neq h(t_2)$.

(ii) Se $s_1 \rightarrow t_1$ e $s_2 \rightarrow t_2$ são enlaces majoritários distintos , então a desigualdade $h(s_1) < h(s_2) < h(t_2) < h(t_1)$ não pode ocorrer (Fig. 16).

Para estas duas propriedades, podemos ver que, para uma altura n:

(iii) existe no máximo um enlace majoritário $s \rightarrow t$ com $h(t) = n$;

(iv) existe no máximo um *checkpoint* justificado com altura n.

4.4 Conclusões

O mecanismo de consenso *Casper* ainda não foi colocado em produção na plataforma *Ethereum* devido à sua elevada complexidade. Além disso, ainda não está claro como algumas funcionalidades e características poderão ser implementadas.

Entretanto, deve ser destacado que este protocolo resolve, a principio, o problema dos *forks* por meio da aplicação de penalidade aos nós que não seguirem as regras à risca. A punição via perda financeira é uma boa alternativa para mitigar os efeitos nocivos que poderiam ser causados pela existência de *forks*.

Com base nos conhecimentos adquiridos com o estudo de diversos mecanismos de consenso e, em particular, com a proposta do mecanismo PoS *Casper* do *Ethereum*, elaboramos uma proposta de um novo mecanismo, também baseado em PoS, mas mais simples em termos de implementação e funcionamento, conforme poderá ser avaliado no próximo capítulo.

5 Proposta de um novo Mecanismo de Consenso baseado em PoS

5.1 Características Desejáveis

Comparando os pontos fortes e fracos dos mecanismos de consenso descritos no Cap. 3, foi possível chegar a um conjunto de características desejáveis para um novo mecanismo de consenso. Este mecanismo deve procurar resolver os principais problemas apontados e facilitar a participação em igualdade de condições para qualquer usuário que deseja fazer parte da rede. As características que vão ao encontro de tais premissas são:

1. baixa tolerância a fraudes;
2. baixo consumo energético;
3. boa capacidade de armazenamento de dados;
4. escolha do criador do bloco por sorteio;
5. recompensa para os criadores de blocos;
6. punição para fraudadores.

Em outros trabalhos (SANKAR *et al.*, 2017; LIN; LIAO, 2017; CACHIN; VUKOLIć, 2017) são descritos mecanismos que atendem algumas destas características. Entretanto, eles consideram *blockchains* privadas para a implantação das propostas, nos quais a questão é muito simplificada por haver um controle total dos usuários e de sua participação.

Destacamos uma vez mais que, neste projeto, buscamos viabilizar as características acima descritas em uma *blockchain* pública, onde os desafios são maiores devido à inexistência de controles sobre quem entra e quem sai da rede.

A baixa tolerância a fraudes é uma característica que vai permitir que o nó tenha a confiança de poder utilizar o mecanismo (JOSHI *et al.*, 2018), sem temer que conluios possam afetar a rede, entre outros tipos de ataques (LI *et al.*, 2018).

O baixo consumo energético é uma caraterística que está diretamente relacionada ao volume de cálculos necessários. Uma redução é possível se a escolha de mineradores for feita por sorteio, já que esta abordagem não exige alto poder computacional. É necessário

um mecanismo que produza um baixo consumo energético já que, como mencionam Holthaus et.al (HOLTHAUS, 2018), o gasto energético da criptomoeda *Bitcoin* esta crescendo de forma paulatina, podendo afetar a pegada climática no futuro. Por sua vez Mora et.al (MORA *et al.*, 2018) comenta que, caso o consumo de energia do *Bitcoin* continue aumentando, daqui a uns 22 anos ele poderá acrescentar $2°C$ à temperatura global. A escolha do criador por meio sorteio oferece uma opção mais econômica para a criação do bloco, já que o nó não precisa de muito poder computacional para tentar ser o escolhido (SALEH, 2018) (HARPER, 2017).

A caraterística da alta capacidade de armazenamento de dados, pode ser questionável. Tschorsch et. al (TSCHORSCH; SCHEUERMANN, 2016) mencionam que um aumento considerável no tamanho do bloco traz problemas relacionados à transmissão dos blocos pela rede, o que acrescentaria um aumento no tempo para se alcançar um consenso. Apesar de ser questionável, esta característica pode ser desejável no caso em que se queira armazenar informações mais completas na *blockchain* e não apenas uma referência. Ela também pode ser desejável para viabilizar mecanismos de consenso que sejam resistentes a ataques de computadores quânticos. Por exemplo, se uma nova *blockchain* pós-quântica fizer uso de assinaturas baseadas em *hash* (BUCHMANN *et al.*, 2008), será necessário aumentar o espaço para armazenar as grandes assinaturas que serão geradas.

A recompensa é uma forma de incentivar primeiramente que os participantes permaneçam na rede, mantendo a atividade de criação de blocos. Essa recompensa deve permitir também que outros participantes tenham interesse em entrar na rede e com isso aumentar a descentralização do processo de criação de blocos. Além disso, ela também pode incentivar que os nós atuem de forma correta e honesta na rede (LUU *et al.*, 2015). A punição de fraudadores é uma forma de controlar o comportamento dos nós na rede, incentivando a criação de blocos corretos (BUTERIN, 2014). Isto tem muita relação com as recompensas, já que, caso o nó tente trapacear o sistema, ele será punido com uma diminuição de suas moedas (multa ou confisco) ou a anulação de recompensas (LUU *et al.*, 2015).

A fim de contornar o problema de que quem possui mais moedas tem mais chance de ser beneficiado, que está presente nos mecanismos baseados em escolha usando moedas, um novo mecanismo de consenso deveria considerar outras formas de *stake* diferentes do recurso financeiro.

Na proposta descrita a seguir consideramos inicialmente que todos os participantes possuem o mesmo *stake*, o que dá chances iguais a todos de serem escolhidos mineradores, evitando o problema de privilegiar os mais abastados.

5.2 Proof-of-Stake de tempo discreto

Na busca de um algoritmo que atenda algumas das caraterísticas desejadas em uma *blockchain*, chegou-se a uma proposta baseada em PoS, já que esta família de protocolos de consenso não exige alto poder computacional do participante. Quando comparado com mecanismo de consenso baseados em PoW, o PoS traz ganhos com relação ao consumo de energia, mas também os desafios listados abaixo.

1. Como definir qual foi o nó sorteado em um determinado instante?
2. Como limitar o crescimento de cadeias secundárias geradas a partir de *forks*?
3. Como permitir que a geração de novos blocos ocorra de forma descentralizada?

Estas questões são tratadas nas seções seguintes.

5.2.1 O conceito de rodada

Como em todo mecanismo de sorteio, é necessário garantir que um determinado participante realmente tenha sido sorteado. Em um ambiente distribuído é necessário que todos os outros participantes possam verificar quem de fato foi sorteado em um determinado momento.

Cada bloco possui um referencial de tempo definido pelo seu instante de criação ou de chegada em um nó. O uso do tempo de chegada é interessante, pois cria uma escala de tempo relativo ao próprio nó que recebe o bloco, permitindo trabalhar com um intervalo de tempo mais consistente.

No caso do bloco gênesis este referencial de tempo é um padrão que serve como referência para outros blocos. Este referencial de tempo global permite que todos os blocos possam ser verificados igualmente em qualquer nó da rede.

Sendo assim, o protocolo passa a usar um tempo discreto, no qual cada intervalo de tempo é chamado de rodada. As rodadas são números inteiros crescentes usados igualmente por todos os participantes, onde cada rodada tem duração fixa de um *timeout*. O *timeout* é definido de forma global na rede e utilizado por todos os nós. Isto controla a criação dos blocos, evitando que os participantes com maior poder computacional tenham grandes vantagens na mineração.

As rodadas são definidas pela Eq. 5.1.

$$rodada = último\_bloco.rodada + \left\lfloor \frac{TIME() - último\_bloco.tempo\_chegada}{TIMEOUT} \right\rfloor \quad (5.1)$$

onde:

- $TIME()$ representa a data e hora atual no padrão UTC,
- $último\_bloco.tempo\_chegada$ representa o tempo de chegada do bloco anterior,
- $último\_bloco.rodada$ representa a rodada do bloco anterior,
- $TIMEOUT$ representa a duração de cada rodada.

Além do conceito de rodada, temos também o conceito de rodada esperada, que é o valor de rodada estimada com base no tempo transcorrido desde a geração do último bloco da cadeia. Este conceito é utilizando pelos outros nós para verificar se um novo bloco, recém recebido da rede foi efetivamente criado dentro da rodada esperada.

Este mecanismo é um importante controle de fraudes, já que é possível de forma local verificar se um nó, criador de um determinado bloco, respeitou a rodada esperada no momento da criação. Todos os nós verificam se um determinado bloco é consistente com a rodada esperada, caso não seja, o bloco é descartado. Nesta verificação, há uma tolerância para compensar eventuais atrasos de propagação na rede.

$$rodada\_esperada = último\_bloco.rodada + rodadas\_transcorridas \tag{5.2}$$

onde

$$rodadas\_transcorridas = \left\lfloor \frac{novo\_bloco.tempo\_chegada - último\_bloco.tempo\_chegada}{TIMEOUT} \right\rfloor \tag{5.3}$$

5.2.2 Sorteio do nó em uma rodada

O novo mecanismo proposto não irá considerar inicialmente diferenciação entre os *stakes* dos participantes (PoS estático), igualando as chances dos mesmos serem sorteados para a criação de um novo bloco (mineração) (FAN; ZHOU, 2017).

Como já foi mostrado pela Eq. 3.2, cada participante deve calcular um *hash de prova* e o resultado do mesmo deve ser inferior a um dado objetivo. Supõe-se o uso de uma função *hash* de boa qualidade, na qual as saídas não demonstram correlação entre si, mesmo se as entradas forem correlacionadas. Neste caso, tal função *hash* pode ser usada como gerador de

números pseudoaleatórios ou como um mecanismo de sorteio. Se este *hash de prova* usa como entrada a identidade (fixa) do nó participante e outros dados que não podem ser alterados com facilidade por ele, então conseguir vencer o objetivo depende em parte da sorte.

O *hash de prova* usado neste novo mecanismo está baseado na função de *hash* criptográfico SHA-256 e é calculado pela Eq. 5.4.

$$hash\_de\_prova = H(\text{último\_bloco.hash} \| rodada \| identificador\_do\_nó) \tag{5.4}$$

onde:

- $H()$: é a função *hash* SHA-256,
- último_bloco.hash: é o *hash* do último bloco da cadeia principal,
- *rodada*: é o valor da rodada atual,
- $identificador\_do\_nó$: é a identidade do nó

O cálculo do *hash de prova* funciona então como uma espécie de sorteio baseado nas três informações de entrada. Dependendo do valor definido para o objetivo (Eq. 3.2), é provável que o nó não consiga um *hash de prova* pequeno o suficiente e tenha que fazer nova tentativa, o que só poderá ocorrer em uma próxima rodada, ou seja, após esperar pelo menos um *timeout*.

5.2.3 Estrutura do bloco para o novo consenso proposto

Neste protocolo foi necessário alterar o cabeçalho do bloco de referência apresentado no Cap. 2 (Fig. 2) para viabilizar as novas funcionalidades. Na Fig. 17, pode-se observar a nova estrutura.

Nota-se que há grande semelhança entre as duas estruturas, havendo na prática só a troca do campo *nonce* pelo campo rodada. Entretanto, há uma diferença significativa no uso destes campos pelos protocolos PoW e o proposto aqui (PoS simplificado): o valor da raiz da árvore de Merkle não é usado no cálculo do *hash de prova* (sorteio para mineração), de maneira a evitar que um nó possa, mediante trocas das transações que irão compor o bloco, alterar a árvore de Merkle e, assim, fazer várias tentativas de sorteio dentro de uma mesma rodada.

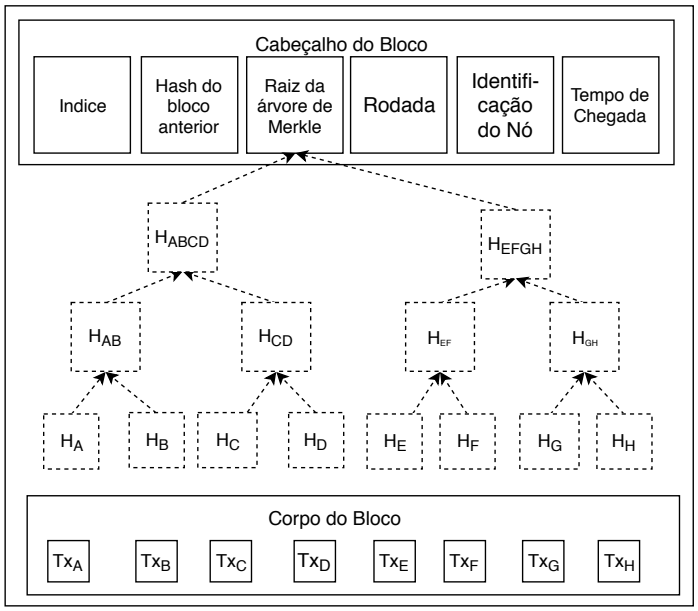


Figura 17 – Estrutura do bloco da proposta

Esta mudança da estrutura do bloco também deixa fora o campo do tempo de chegada, por ser um campo que, para um mesmo bloco, pode ser diferente em cada nó.

$$Hash\ do\ cabeçalho = H(Indice \| Hash\ bloco\ anterior \| Rodada \| Identificação\ nó) \quad (5.5)$$

Estas mudanças na estrutura dos blocos não devem afetar significativamente a propagação dos mesmos pela rede, já que mantêm aproximadamente o mesmo tamanho dos blocos de outras *blockchains*. No *Bitcoin*, por exemplo, este tamanho é de aproximadamente 500 KB o que, segundo Gervais et.al (GERVAIS *et al.*, 2016), acarreta em um tempo médio de propagação de 8.7s.

Mais adiante são fornecidos mais detalhes sobre o funcionamento do protocolo e sobre a criação e uso dos campos desta estrutura.

5.3 Controle de Forks

Diferentemente dos algoritmos baseados em prova de trabalho, onde trabalhar em duas ou mais cadeias traz alto custo, em algoritmos PoS esse custo é desprezível. Por isso, os algoritmos de consenso PoS permitem que um mesmo nó minerador crie blocos em diferentes cadeias e não apenas na cadeia mais longa.

Como consequência disso, ele pode contribuir, quase sem custos adicionais, para que uma cadeia secundária ultrapasse em um determinado momento a cadeia principal em ta-

manho, causando efeitos indesejáveis para a rede. Na literatura este problema é conhecido como *Nothing-at-Stake*, ou seja, um participante de forma natural não tem nada a perder em trabalhar ao mesmo tempo em várias cadeias diferentes.

O uso do *timeout* ajuda na redução de *forks*, pois, para que aconteça um *fork* é necessário que dois participantes criem dois blocos que atendam o desafio e que a rodada seja exatamente a mesma. Em outras palavras, a exigência de que seja na mesma rodada reduz a probabilidade de *forks*. Mas se o *timeout* não for calibrado corretamente, ele pode gerar um impacto no desempenho da *blockchain* e ainda não evitar que possam acontecer *forks* de alguma forma.

Segundo (BUTERIN; GRIFFITH, 2017) é necessário criar mecanismos para punir os participantes que desrespeitem a obrigação de gerar blocos para a cadeia mais longa. Ele propõe exatamente isso no algoritmo *Casper* descrito no Cap. 4.

No novo mecanismo proposto, a penalidade estará na menor probabilidade de sorteio para um participante que deseja criar um bloco em uma cadeia secundária. É permitido ao participante criar blocos em qualquer cadeia, porém a probabilidade de geração de blocos na cadeia principal é consideravelmente maior quando comparada com outras cadeias. Desde que essa diferenciação seja adequada é pouco provável que uma cadeia que perdeu a disputa em um determinado *fork* em um tempo passado, assuma a condição de cadeia principal.

A proposta para diferenciar a probabilidade de mineração por cadeia é trocar o objetivo da Eq. 3.2 por aquele dado pela Eq. 5.6.

$$objetivo = 2^{256-D-P} \tag{5.6}$$

Este novo objetivo também deve ser compatível com o *hash* de prova, isto é, deve ser representável por, no máximo, 256 *bits*. Os parâmetros D (dificuldade) e P (penalidade) irão controlar a dificuldade (com impactos na velocidade de convergência do algoritmo) e o nível da punição a ser aplicada aos nós que não seguirem as regras adequadamente.

- D: valor entre 0 e $(256 - P)$ que permite aumentar ou diminuir a dificuldade para a criação de um bloco de forma global para todas as cadeias.

- P: valor entre 0 e $(256 - D)$ que define a penalidade a ser aplicada a um nó por gerar um bloco em uma cadeia que não seja a principal (a mais longa). Caso seja criado um bloco na cadeia principal, adotamos $P = 0$.

Para exemplificar o conceito de probabilidades diferentes para cadeias diferentes, va-

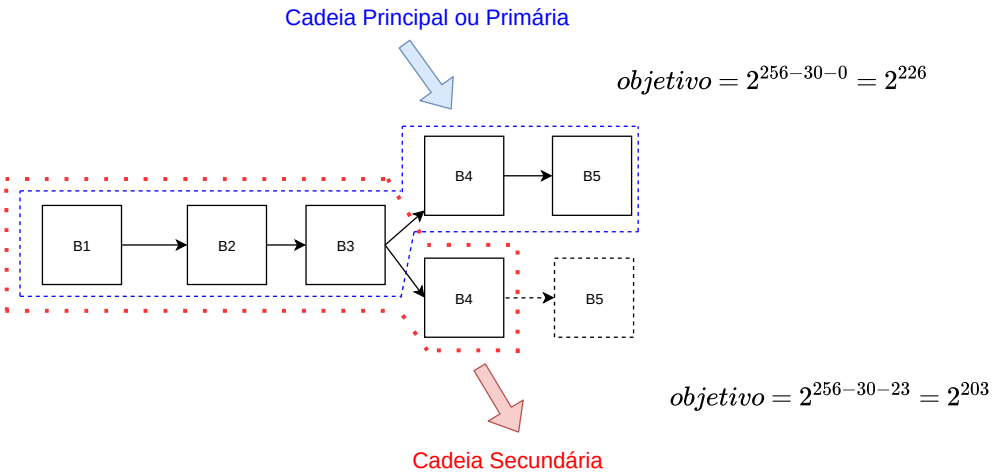


Figura 18 – Exemplo de penalidade

mos considerar duas cadeias pertencentes à *blockchain* local de um determinado nó participante, onde em uma existam 5 blocos e em outra 4 blocos.

Vamos supor também que as cadeias sejam idênticas até o bloco B_3, ou seja, ocorreu um *fork* no momento da geração do bloco B_4 como se pode observar na Fig. 18.

Desta forma, caso um participante queira gerar o bloco B_5 da cadeia secundária, considerando $P = 23$ e $D = 30$, temos:

$$hash\_de\_prova \ < \ 2^{256-30-23} = 2^{203} \tag{5.7}$$

Nestas condições particulares, a chance do nó satisfazer a condição acima é de 2^{203} possibilidades em 2^{256} combinações possíveis, ou seja, a probabilidade de criar um bloco na cadeia secundária é de 2^{-53}, valor muito baixo, mas que pode ser ajustado.

Se forem adotados estes valores, esta proposta de controle de *forks* equivale a praticamente proibir a geração de blocos em cadeias secundárias. Mas, por meio do parâmetro P, pode ser avaliado o impacto de diferentes penalizações no comportamento do protocolo.

5.4 Processos do protocolo

O algoritmo do protocolo de consenso proposto se divide principalmente em três processos, que são executados de forma concorrente.

- Processo de mineração de novos blocos
- Processo de escuta de outros nós para recebimento de novos blocos minerados por eles
- Processo de sincronização da cópia da *blockchain* no nó participante

5.4.1 Processo de mineração de novos blocos

O processo de mineração consiste em o nó participante verificar se foi sorteado em uma rodada para criar o novo bloco, como se pode observar no Alg. 5.1 e no Alg. 5.2 e no diagrama da Fig. 19. O processo de mineração espera que o evento *start* ocorra (isto é, seja ativado), para iniciar sua atividade; caso contrário ele fica em estado de espera. O processo é repetido enquanto o evento *kill* não ocorrer .

Caso o participante seja sorteado para criação do próximo bloco em alguma rodada, ele poderá criar o bloco e enviá-lo em *broadcast* para os outros participantes.

Algoritmo 5.1 Mineração de blocos

Entrada:
Saída:

```
1.  enquanto verdadeiro faça
2.    se evento start foi ativado então
3.      último_bloco ← blockchain.obter_último_bloco()
4.      id_nó ← obter_identificação()
5.      bloco ← Protocolo_PoS(último_bloco, id_nó)
6.      se bloco não é nulo  e   flag_escuta = 0 então
7.        blockchain ← blockchain||bloco
8.        Broadcast(bloco) // enviar por broadcast o bloco bloco para os nós
9.        sleep(TIMEOUT)
10.     senão
11.       flag_escuta ← 0
12.     fim se
13.   fim se
14. fim enquanto
```

5.4.2 Processo de escuta e recebimento de novos blocos

O processo de escuta é responsável por receber novos blocos criados por outros participantes e validá-los, buscando sua inserção na *blockchain* local. Este processo é mostrado no Alg. 5.3 e no diagrama da Fig. 20.

Algoritmo 5.2 Protocolo PoS

Entrada: *último_bloco*, *id_nó*
Saída: bloco *bloco*

1. $rodada \leftarrow$ *último_bloco.rodada* $+ \left\lfloor \frac{\texttt{TIME}() - \textit{último\_bloco.tempo\_chegada}}{TIMEOUT} \right\rfloor$
2. **enquanto verdadeiro faça**
3. $hash \leftarrow H\,(\textit{último\_bloco.hash} \| rodada \| \textit{id\_nó})$
4. $índice \leftarrow \textit{último\_bloco.índice} + 1$
5. **se** *hash_de_prova* $<$ *objetivo* **então**
6. $tempo\_chegada \leftarrow$ `TIME()` // tempo de criação de bloco
7. $bloco \leftarrow [índice, \textit{último\_bloco.hash}, rodada, \textit{id\_nó}, hash, tempo\_chegada, tx]$ // criar o *bloco*
8. **retorna** *bloco*
9. **fim se**
10. `sleep`(TIMEOUT)
11. $rodada \leftarrow rodada + 1$
12. **fim enquanto**
13. **retorna** nulo

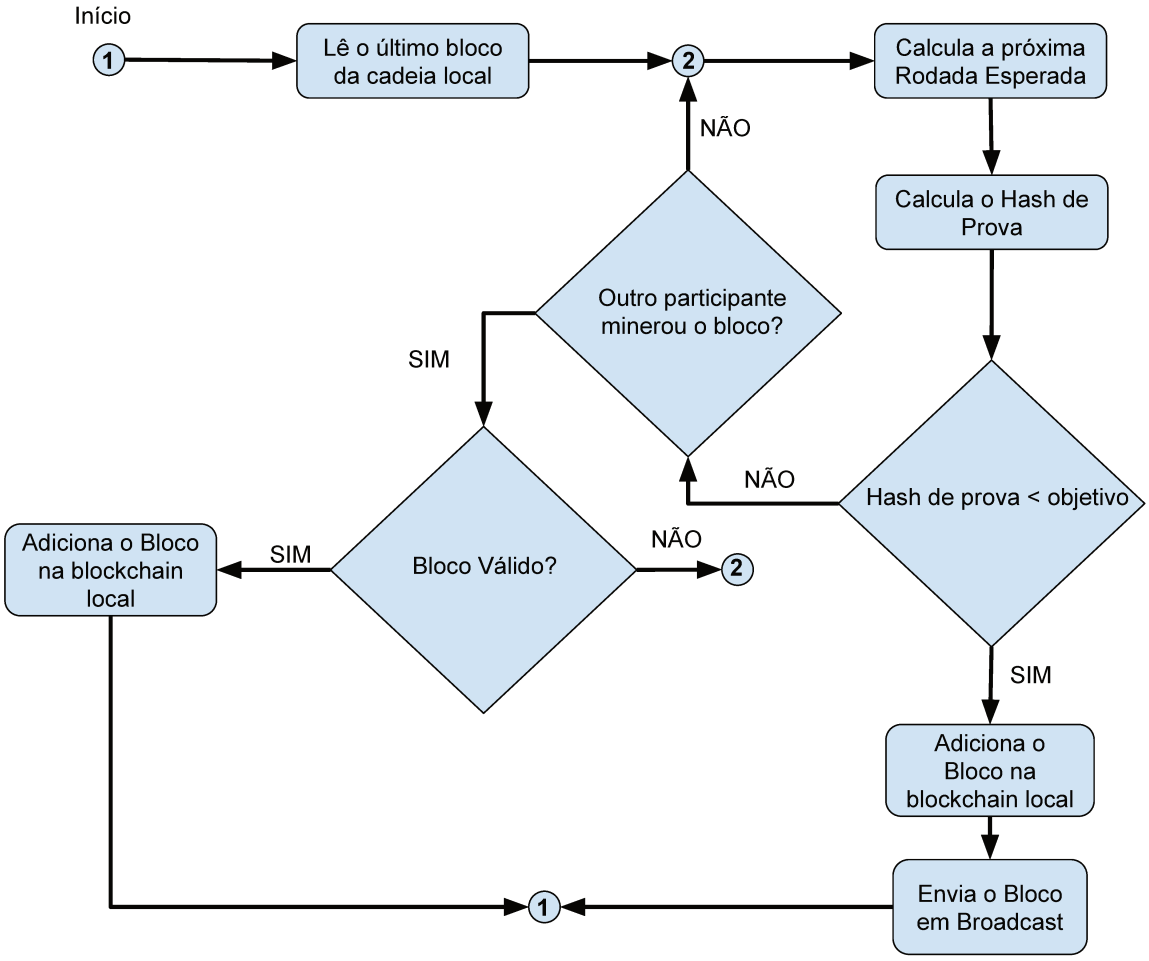


Figura 19 – Processo Mineração de Novos Blocos

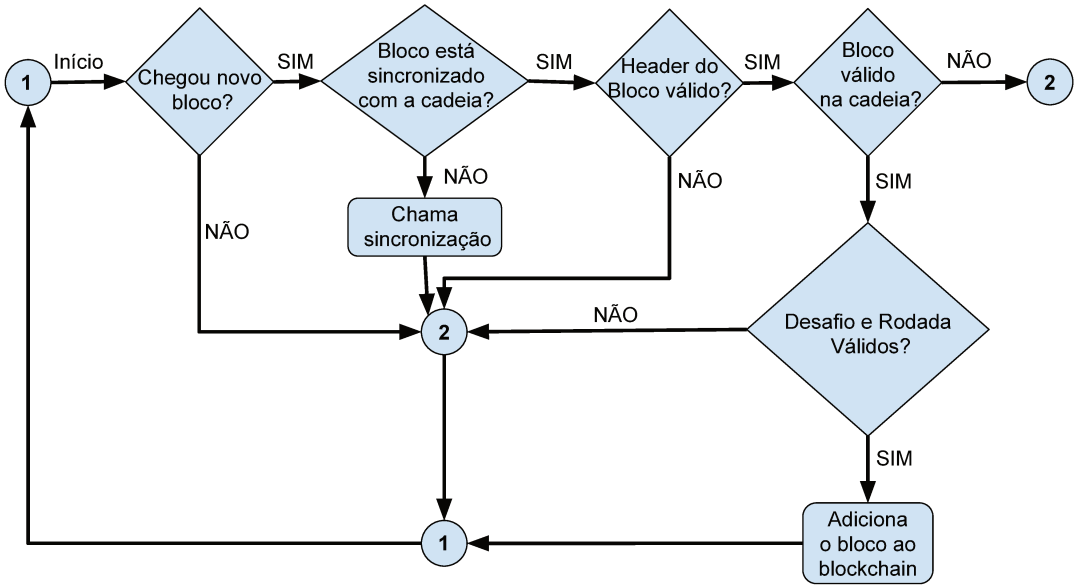


Figura 20 – Processo Escuta de Blocos

O processo de escuta fica esperando uma mensagem com um novo bloco o tempo todo. Quando a mensagem chega, primeiramente é feita uma validação do cabeçalho do bloco; depois é validada a consistência do bloco com a cadeia.

Logo depois, é validada a rodada esperada e, por último, se verifica se o bloco de fato passou no desafio do sorteio. Caso qualquer uma destas verificações falhe, o bloco é descartado e o processo volta ao início para esperar novos blocos.

Caso todas as validações estejam corretas, é verificado se o índice do bloco é maior que o último bloco da cadeia em apenas uma unidade. Caso isso ocorra, o bloco é inserido na *blockchain* local.

Se o índice do bloco superar o do último bloco da cadeia em mais de uma unidade, chama-se o processo de sincronização, que irá buscar os blocos faltantes e tratar casos de *forks*, se existirem.

5.4.3 Processo de sincronização da *blockchain* do nó participante.

O processo de sincronização consiste em buscar novos blocos de outros participantes para obter a cadeia principal com todos os blocos já gerados. O processo completo pode ser observado no Alg. 5.4 e na Fig. 21.

Ao ser chamado, o processo solicita o bloco mais recente dos *peers* e compara o índice do bloco recebido com o último bloco da cadeia local. Caso a diferença dos índices seja de uma unidade, faz-se a verificação do cabeçalho, consistência do bloco e de rodada. Caso as

Algoritmo 5.3 Escuta de blocos

Entrada: *blockchain* local do nó, *bloco* recebido
Saída:

```
1.  enquanto verdadeiro faça
2.    bloco ← bloco recebido do outro nó
3.    bloco.tempo_chegada ← TIME()
4.    se cabeçalho_valido(bloco) é verdadeiro então
5.      último_bloco ← blockchain.obter_último_bloco()
6.      se (bloco.índice − último_bloco.índice = 1 e
        bloco_valido(bloco, último_bloco) é verdadeiro e
        rodada_valida(último_bloco, blockchain) é verdadeiro e
        atende_desafio(b) é verdadeiro) então
7.        blockchain ← blockchain||bloco // inserir bloco na blockchain
8.      senão se bloco.índice − último_bloco.índice > 1 então
9.        sincronizar(bloco, último_bloco) // sincroniza a blockchain
10.     senão se bloco.índice = último_bloco.índice então
11.       se bloco.hash = último_bloco.hash então
12.         imprime 'retransmissão'
13.       senão
14.         imprime 'possivél fork'
15.         se (bloco_valido(bloco, último_bloco) é verdadeiro e
            rodada_valida(último_bloco, blockchain) é verdadeiro) então
16.           // armazena o bloco na base de dados local para resolver o fork posteriormente
17.         fim se
18.       fim se
19.     senão
20.       imprime 'bloco antigo'
21.     fim se
22.   senão
23.     imprime 'bloco inválido'
24.   fim se
25. fim enquanto
```

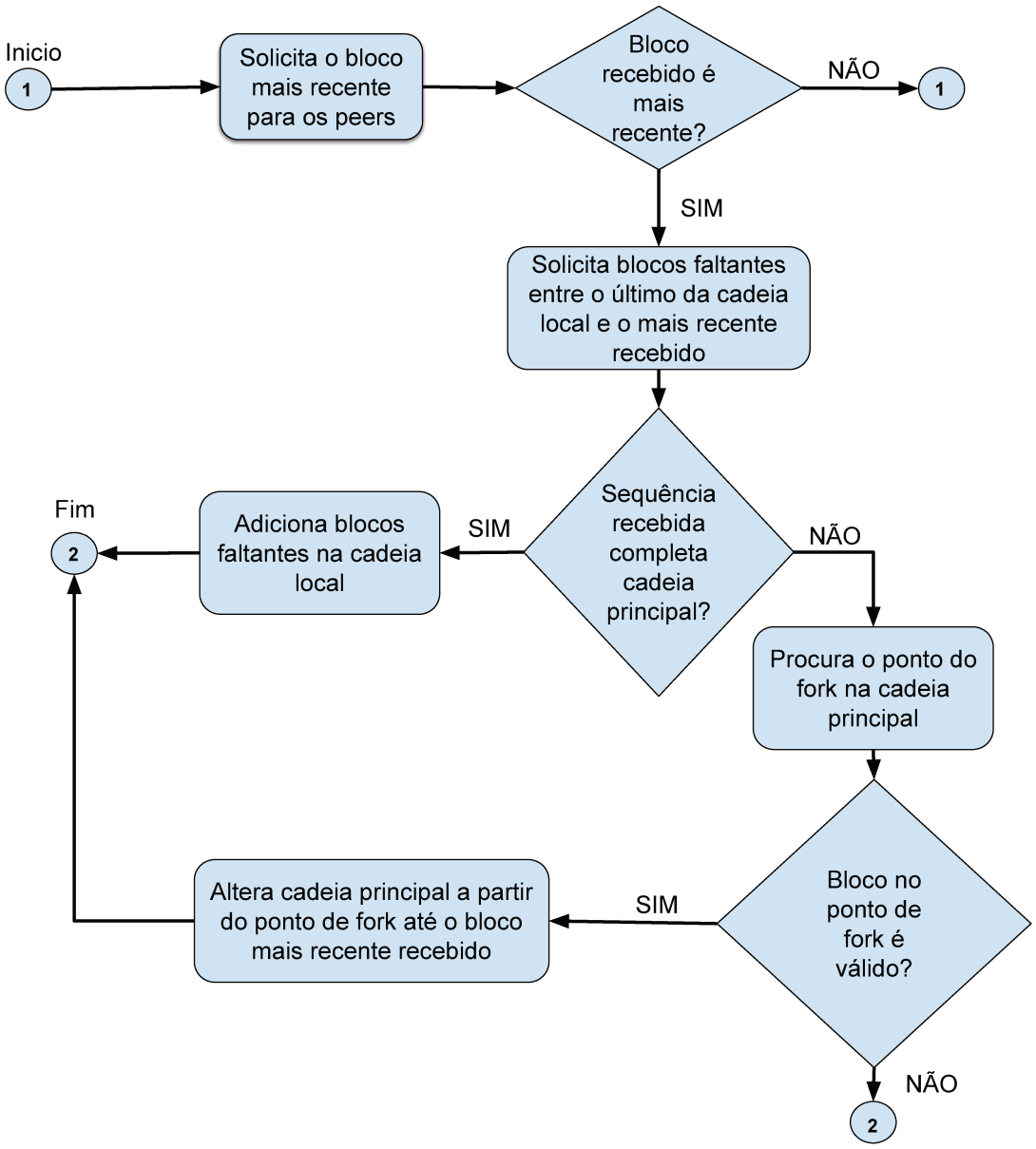


Figura 21 – Processo Sincronização de Nós

verificações sejam corretas, adiciona-se o bloco na cadeia local; caso a diferença dos índices seja maior que uma unidade, então o participante solicita todos os blocos faltantes. Depois de receber os blocos faz-se as verificações dos cabeçalhos, consistência dos blocos e das rodadas, para adicionar os blocos faltantes na *blockchain*.

5.5 Comparação com *Casper*

Esta seção fará algumas comparações qualitativas entre o *Casper* (descrito no Cap. 4) e o novo mecanismo proposto. Como se busca uma alternativa simplificada para um algoritmo

Algoritmo 5.4 Sincronização de blocos

Entrada: *bloco_pedido*, endereço *endereço_nó* do peer a fazer petições , *blockchain* local do nó
Saída:

```
1.  imprime 'sincronizando...'
2.  se bloco_pedido é nulo então
3.    bloco_pedido ← blockchain.obter_último_bloco()
      // limitar o número de petições aos peers
4.    para i ← 0 até 3 faça
5.      [bloco, ip_emissor] ← pedir_último_bloco()
6.      se bloco não é nulo e bloco.índice < bloco_pedido.índice então
7.        bloco_pedido = bloco
8.        endereço = ip_emissor
9.        interromper
10.     fim se
11.   fim para
12. fim se
13. último_bloco = obter_último_bloco()
14. se bloco_pedido.índice > último_bloco.índice então
15.   se bloco_pedido.índice − último_bloco.índice = 1 então
16.     se bloco_valido(bloco, último_bloco) é verdadeiro e
        cabeçalho_valido(bloco) é verdadeiro e
        atende_desafio(bloco) é verdadeiro então
17.       imprime 'bloco valido'
18.       blockchain ← blockchain||bloco // inserir bloco na blockchain
19.     fim se
20.   senão se bloco_pedido.índice − último_bloco.índice > 1 então
21.     cadeia ← pedir_blocos(último_bloco.índice + 1, bloco_pedido.índice, endereço)
22.     se cadeia não esta vazia então
23.       para i ← 0 até long(cadeia) faça
24.         se cabeçalho_valido(cadeia[i]) é verdadeiro e
            bloco_valido(bloco, ultimo_bloco) é verdadeiro e
            atende_desafio(b) é verdadeiro e
            rodada_valida(ultimo_bloco, blockchain) é verdadeiro então
25.           ultimo_bloco ← cadeia[i]
26.           blockchain ← blockchain||cadeia[i]
27.         fim se
28.       fim para
29.     fim se
30.   fim se
31. fim se
```

baseado em PoS, existem diferenças significativas quando comparamos nosso algoritmo com um modelo baseado no *Casper*.

O protocolo de tempo discreto proposto neste trabalho anula possíveis vantagens que participantes teriam em criar blocos mais rapidamente por meio de instalações com grande poder computacional. Tal abordagem pode trazer uma queda de desempenho no algoritmo (dependendo das configurações) pois, em muitos momentos, os participantes estarão esperando por uma nova oportunidade de criar os blocos.

Entretanto, esta potencial queda de desempenho poderá ser mitigada com o crescimento do número de participantes, pois quanto maior o número de nós maior será a probabilidade de pelo menos um deles ser sorteado em cada rodada, o que evitaria a situação indesejável de se ter uma rodada sem nenhum sorteio. Obviamente, o nível de dificuldade e o $TIMEOUT$ também podem ser ajustados para diminuir a probabilidade de uma rodada sem produção de pelo menos um bloco.

O *Casper* possui o conceito de *checkpoints* para determinar marcos na *blockchain* e assim lidar com os *forks*, procurando preservar apenas a cadeia de blocos mais longa. Estes *checkpoints* e os $forks$ acabam formando uma estrutura de *blockchain* complexa chamada de árvore de *checkpoints*.

No *Casper* o problema dos *forks* é tratado com apostas que um determinado participante faz. Existe um complexo sistema de punições chamado *slasher*, que garante a integridade do protocolo. Caso algum participante crie um bloco em uma cadeia secundária, desafiando as regras vigentes, isso será identificado e o participante será punido com a perda de moedas (perda da aposta) .

No mecanismo proposto tenta-se contornar o problema por meio do uso do tempo discreto para a mineração e pela adoção de parâmetros que diminuem sensivelmente a probabilidade de um nó participante gerar bloco em uma cadeia secundária.

O *Casper* utiliza um grupo de validadores, onde cada participante deste grupo realiza um investimento. Já no mecanismo proposto, qualquer pessoa pode participar e o sorteio ocorre de forma equiprovável para todos os participantes (na proposta inicial).

Com relação aos validadores do *Casper*, a garantia está nos enlaces majoritários, onde os participantes votam em uma determinada relação entre os *checkpoints*. No esquema proposto, o algoritmo tenta seguir a cadeia principal, permitindo, porém, a criação de blocos nas cadeias secundárias com menor probabilidade. Isso é importante, pois um determinado nó pode ter uma visão incompleta ou defasada da *blockchain* e achar que um ramo é secundário, quando na verdade é o principal. Acreditamos que permitir a criação de blocos em ramos secundários, ainda que com baixa probabilidade de sucesso, poderá ajudar a melhorar o

Tabela 4 – Comparação do mecanismo proposto com o Casper

	Casper	PoSTD
Forma de Lidar com *forks*	• uso de *checkpoints* • sistema de punições	• uso do tempo discreto para a mineração • adoção de parâmetros que diminuem a probabilidade
Participantes do mecanismo	grupo de validadores	qualquer nó pode participar
Garantia de mecanismo	está nos enlaces majoritários	tenta seguir a cadeia principal, mas é possível criar na cadeia secundária.
Como Lida com *Nothing at Stake*	uso de punições	uso de tempo discreto (rodadas)
Estrutura	simples	complexa (uso de Contratos Inteligentes)

desempenho do sistema como um todo.

O *Casper* lida com o problema do *Nothing-at-Stake* por meio da ameaça de punições. O mecanismo proposto lida com este problema usando o conceito de tempo discreto (rodadas), o que não permite que um nó faça mais que uma tentativa por rodada por ramo da *blockchain*.

Outro ponto positivo desta nova abordagem é sua simplicidade, já que não exige nenhum mecanismo adicional e complexo, como é o caso dos contratos inteligentes (*smart contracts*) do *Casper*. A Tab. 4 apresenta de forma resumida as principais comparações qualitativas.

5.6 Relação da proposta com os outros mecanismos

Uma pergunta possível é por que não foi escolhido algum outro mecanismos diferente de PoS como base para este novo protocolo, já que outros têm capacidades muito interessantes. Como explicado anteriormente a ideia é utilizar um mecanismo de baixo custo e, por isso, deve ser deixado de fora o mecanismo PoW. Assim também deve ser deixado de fora o PoA, já que também possui um consumo elevado, por ser um mecanismo híbrido e possuir uma parte PoW.

Por outro lado tanto no caso do PoA como em mecanismos como DPOS e *Algorand*,

sendo parte PoS ou baseados em PoS, existe uma certa complexidade estrutural adicional na forma de eleição de comitês e votações, algo não presente na estrutura mais simples aqui proposta para o PoSTD.

Apesar de o mecanismo PoB possuir um funcionamento simples, ele também apresenta uma desvantagem forte: o investimento sucessivo de moedas pode tornar inviável seu uso no longo prazo.

5.7 Análise de Segurança

5.7.1 Segurança na *blockchain* e no mecanismo de consenso

A segurança na *blockchain* esta atrelada ao uso de um esquema criptográfico de chave pública na forma de assinaturas digitais, as quais protegem as transações, garantindo sua integridade, autenticidade e irretratabilidade (WANG *et al.*, 2018). Este esquema garante a segurança das moedas de um usuário por meio do gerenciamento das chaves privada e pública em uma carteira. (ANTONOPOULOS, 2017)

O tipo de assinatura mais comum utilizada nas *blockchains*é o ECDSA, o qual está baseado em curvas elípticas. Como é comentado na literatura (PAAR; PELZL, 2010; STALLINGS, 2017), este esquema possui uma segurança satisfatória caso seja implementado de forma correta. No caso particular de *Bitcoin* é utilizada a curva secp256k1 (AGNER, 2016) a qual está dentro das curvas recomendadas pelo relatório da *Certicom* (BROWN, 2010), usando uma chave privada e grupo cíclico de comprimento 256 bits que dota a assinatura de uma segurança de 128 bits (PAAR; PELZL, 2010).

Outro elemento que protege as transações é a Árvore de *Merkle*, a qual esta baseada em funções *hash* de boa qualidade e criptograficamente seguras. Estas funções são utilizada para intertravamento de blocos e (em algumas blockchains) para o sorteio do nós autorizados a criar o novo bloco, provendo segurança, imutabilidade e verificabilidade dos blocos. Nas *blockchains* a função *hash* mais utilizada é o SHA-256, a qual também é utilizada por *Bitcoin* . Como se pode observar em Paar.et al (PAAR; PELZL, 2010), esta função oferece uma segurança de 128 bits, a qual pode ser considerada suficiente atualmente.

Os níveis de segurança apresentados podem ser considerados aceitáveis enquanto não surge um computador quântico viável na prática, já que o mesmo irá afetar a segurança de forma considerável e deve ser objeto de estudos futuros.

Na proposta PosTD, a *blockchain* possui uma segurança herdada em boa parte de outras *blockchains*, já que as mudanças realizadas não afetam a estrutura básica dos blocos.

Mesmo na mudança da forma de cálculo do cabeçalho do bloco, se procurou preservar a verificabilidade dos blocos por parte dos nós, o que garante ainda sua imutabilidade. Foram mantidos as demais características dos outros elementos do bloco com as mesmas caraterísticas que podem ser vistas em outras *blockchains* sem afetar sua segurança.

Por outro lado, como discutido por Lin et.al (LIN; LIAO, 2017), a proposta PoSTD também é vulnerável a ataques de 51%, já que um conluio de mais da metade da rede poderia controlar a criação de blocos falsos, de forma semelhante às *blockchains* baseadas em PoW,

As *blockchains* públicas com algoritmos de consenso baseados em *Proof-of-Stake* apresentam requisitos de segurança maiores que *Proof-of-Work*. Em arquiteturas PoS é possível que um participante crie blocos em qualquer cadeia e não apenas na principal, pois não existe esforço computacional associado a geração do bloco. Este problema acompanha as propostas para PoS desde sua primeira abordagem com a moeda *Peercoin*. Os participantes que exploram esta possibilidade podem atacar o protocolo através de uma reversão de longo alcance na cadeia principal, onde a partir de um ponto de *fork* é possível trabalhar em uma outra cadeia. Caso essa cadeia secundária se torne maior que a principal, os participantes não honestos poderão divulgá-la para o resto da rede tornando a mesma a nova cadeia a ser seguida por todos os participantes, com reversão de todas as transações incluídas no segmento principal onde ocorreu o ataque. Esta reversão de longo alcance foi tratada pela primeira vez por King (KING, 2013) no *PoS* da moeda *Peercoin*, onde este problema foi batizado de *Protection of History*.

A alternativa encontrada por King para proteção do histórico da *blockchain* principal foi desenvolver o conceito de *checkpoints*. Esses *checkpoints* são blocos múltiplos de um valor fixo que demarcam a *blockchain*. Assim, nenhum bloco que seja mais antigo que um *checkpoint* pode ser revertido. Na proposta de King, esses *checkpoints* são enviados através de um controlador de cadeias de forma centralizada e, por isso, possuem desvantagens em uma ambiente totalmente descentralizado. Em propostas baseadas em *proof-of-work* as reversões de longo alcance são mais difíceis de acontecerem na prática, pois o grande poder computacional exigido na mineração dificulta a competição em mais de uma cadeia.

Entretanto, em 2010 Nakamoto (KING, 2013) também fez uma proposta para controlar as reversões de longo alcance na cadeia principal para aumentar a segurança do *Bitcoin*. Atualmente os *checkpoints* do *Bitcoin* são enviados por mensagens de alerta em *Broadcast* para a rede.

Além dos *checkpoints*, algumas propostas de vinculação do *stake* de um participante à sua quantidade de moedas também foram utilizadas. Na moeda *Peercoin*, ao associar o *stake* de um participante à quantidade de moedas mantidas por um período, parâmetro

conhecido como *coin age* (explicado no Cap. 3), obtém-se uma redução nas chances de um participante contribuir para um ataque de longo alcance, pois ele poderia ser prejudicado pela desvalorização da moeda no mercado e seu *coin age* seria fortemente afetado em uma reversão de longo alcance.

5.7.2 Reversão de Longo Alcance no PoSTD

No consenso PoSTD proposto neste trabalho não foi implementado um controle para reversões de longo alcance baseado em *checkpoints*. Consideramos que o protocolo proposto oferece robustez contra este tipo de ataque pela sua própria forma de construção.

Alguns parâmetros importantes são avaliados quando um bloco é proposto, mesmo que este bloco seja enviado de forma atrasada, como acontece neste tipo de ataque. Caso um bloco b'_i seja proposto em algum momento como possível substituto do bloco b_i na cadeia principal, será necessário que esse bloco atenda alguns requisitos. Primeiramente a rodada deste bloco deve ser menor que a utilizada no bloco b_i, pois blocos criados em rodadas menores possuem prioridade. Além disso, o bloco b'_i deve possuir uma rodada maior que a do bloco b_{i-1} em pelo menos uma unidade. Assim, restariam apenas um conjunto limitado a algumas rodadas para que o bloco b'_i possa atingir seu objetivo. Esse conceito pode ser estendido para todos os blocos pertencentes à cadeia proposta pelo atacante, desde o ponto do *fork* até o bloco final da cadeia.

Como veremos nos resultados do Cap. 6, o número esperado de rodadas até um sorteio bem sucedido está associado com a dificuldade do objetivo. Espera-se que, quanto menor a dificuldade, mais rodadas de sucesso ocorram e, consequentemente, mais blocos possam ser propostos em menos tempo. Assim, um bloco estará mais próximo de seu antecessor em número de rodadas, restringindo o espaço disponível entre blocos para um ataque. Caso a dificuldade seja grande, espera-se que o número de rodadas entre blocos aumente e, consequentemente, existirão mais rodadas disponíveis entre blocos para um ataque de longo alcance. Porém seriam necessários mais participantes desonestos para combinar a criação de um bloco com uma rodada menor visando o ataque, pois a dificuldade alta dificultaria o sucesso em um sorteio.

Além da dificuldade de atender o desafio em uma rodada menor, para reversão de um bloco existe também a dificuldade associada à própria definição do ponto de *fork* para início de uma nova cadeia. Como premissa do protocolo, um *fork* ocorre apenas quando dois blocos, digamos b'_k e b_k são criados exatamente na mesma rodada. Essa premissa reduz o número de ocorrências de *forks* e consequentemente a possibilidade de geração de novas cadeias secundárias. Atualmente o controle da dificuldade é manual; porém, em versões futuras

do protocolo, mecanismos automatizados deverão ser adotados para ajustar a dificuldade conforme a variação do número de participantes.

5.7.3 Definição do *stake* no PoSTD

Em todas as propostas atuais de *stake* para os consensos *PoS* existem vantagens e desvantagens. Nas abordagens propostas pela moeda *Peercoin* (KING, 2013) e também em *Blackcoin* (VASIN, 2013), a principal vantagem está na associação do *stake* à quantidade e/ou idade de moedas de um participante. Isso traz responsabilidade financeira aos participantes, já que ações desonestas com o protocolo trazem consequências práticas imediatas.

Entretanto, essa associação provoca uma concentração em torno de quem possui mais recursos facilitando a criação de pequenos grupos de participantes ricos que podem decidir o que acontece com o futuro da *blockchain*.

O protocolo PoSTD proposto utiliza atualmente *stake* de uma unidade para cada participante, o que em termos práticos garante distribuição igualitária de probabilidade para todos os participantes da rede. Esta é a abordagem que oferece maior distribuição possível entre os participantes evitando a formação de pequenos grupos que controlam a rede. Porém, ela permite que um participante possa investir em todas as cadeias existentes sem nenhum tipo de perda associada, como ocorre no *Peercoin* ou *Blackcoin*. Além da necessidade clara de definirmos qual será o *stake* do protocolo nas versões futuras, existe a necessidade de implantação de um esquema de punição para participantes que não sigam o protocolo.

5.8 Conclusões

O novo mecanismo de consenso proposto neste capítulo traz uma solução ao problema de desigualdade de oportunidades na definição do criador de um bloco pois, através da necessidade de espera pela próxima rodada, um nó que possui alto poder computacional não consegue vantagens significativas no protocolo. Além disso, ele propõe um controle de *forks* usando uma estrutura simples.

Apesar do uso de tempo discreto reduzir o desempenho, esse fato pode ser minimizado através do ajuste do parâmetro *TIMEOUT* com base no monitoramento da latência na rede. Além disso, considera-se que possíveis reduções de desempenho serão compensadas pela capacidade do protocolo em garantir uma melhor distribuição da probabilidade de sorteio entre os nós.

6 Testes e avaliações do novo mecanismo

6.1 Resultados teóricos

A Eq. 6.1 apresenta o cálculo da probabilidade de ocorrência de um evento S_1, onde S_1 é o sorteio bem sucedido de um nó, isto é, S_1 é o evento de um nó conseguir atingir o objetivo por meio do *hash* de prova. Neste caso está sendo adotada uma penalização P nula, já que se está considerando a criação no ramo principal da *blockchain*.

$$p[S_1] = \frac{2^{256-D-P}}{2^{256}} \tag{6.1}$$

Usando o resultado da Eq. 6.1 podemos encontrar número esperado de rodadas até ocorrer sucesso no sorteio em um nó (Eq. 6.2).

$$Nr_{S_1} = \left\lceil \frac{1}{p[S_1]} \right\rceil \tag{6.2}$$

Para obter o valor esperado para múltiplos nós, temos que obter probabilidade de sucesso S_n, onde S_n é o sorteio bem sucedido em pelo menos um dos n nós (Eq. 6.3).

$$p[S_n] = 1 - \prod_{i=1}^{n} (1 - p[S_1]) = 1 - (1 - p[S_1])^n \tag{6.3}$$

De forma idêntica pode-se encontrar o número esperado de rodadas para se ter algum sorteio (*hash*) bem sucedido quando se tem mais de um nó (Eq. 6.4).

$$Nr_{S_n} = \left\lceil \frac{1}{p[S_n]} \right\rceil \tag{6.4}$$

Deve ser notado que, devido ao fato de estarmos trabalhando com tempos discretos, onde a menor unidade é o tempo de uma rodada, os valores produzidos pelas Eqs. 6.3 e 6.4 fornecem o número inteiro de rodadas imediatamente superior ao valor teórico (e muitas vezes fracionário) calculado a partir das probabilidades.

Na prática é bem provável que se encontre valores de rodadas menores que o limite superior aqui calculado. Entretanto, espera-se que estas diferenças entre valores estimados e medidos sejam cada vez menores à medida que se aumenta o número de rodadas necessárias para se conseguir um sucesso em sorteios. Para sorteios bem sucedidos em poucas rodadas, é

natural que o erro relativo entre os valores estimados e medidos seja maior. Mas isso não deve trazer nenhum tipo de dificuldade, pois sendo um erro de no máximo uma rodada, pouco efeito prático ele terá.

Com base nestas expressões foram calculados vários parâmetros para avaliar, de forma teórica, o comportamento de uma *blockchain* controlada pelo novo consenso PoS simplificado aqui proposto. As figuras a seguir apresentam os principais resultados teóricos com suas respectivas análises.

Na seção seguinte são apresentados resultados práticos e são feitas as devidas comparações com os resultados teóricos.

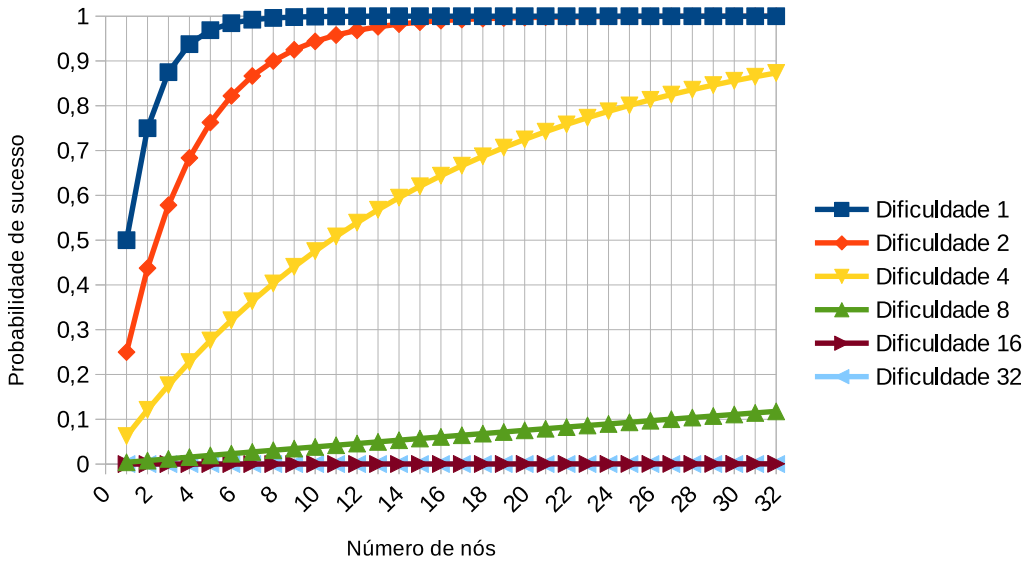


Figura 22 – Probabilidade de sucesso no sorteio em função do número de nós

Como pode ser visto nas Figs. 22, 23 e 24, a probabilidade de sucesso em uma determinada rodada aumenta com o número de nós participantes na *blockchain*, mas diminui muito rapidamente quando cresce o nível de dificuldade.

Na prática, os níveis de dificuldade devem ser ajustados de maneira a não permitir que seja muito provável qualquer nó ter sucesso no sorteio (aumentaria o número de $forks$), mas também não pode tornar esta probabilidade muito pequena, pois isso resultaria em atrasos muito grandes no processo de mineração, como mostram as Figs. de 25 a 30.

É possível notar (Fig. 23) que, mesmo para uma rede similar em tamanho à rede do *Bitcoin* (cerca de 10.000 nós), os níveis de dificuldade 16 e 32 parecem ser muito fortes, pois não permitem que a probabilidade de sucesso no sorteio de algum nó atinja 15%. Pode parecer pouco, mas 15% de 10.000 são 1.500 nós tendo sucesso no sorteio, o que pode resultar

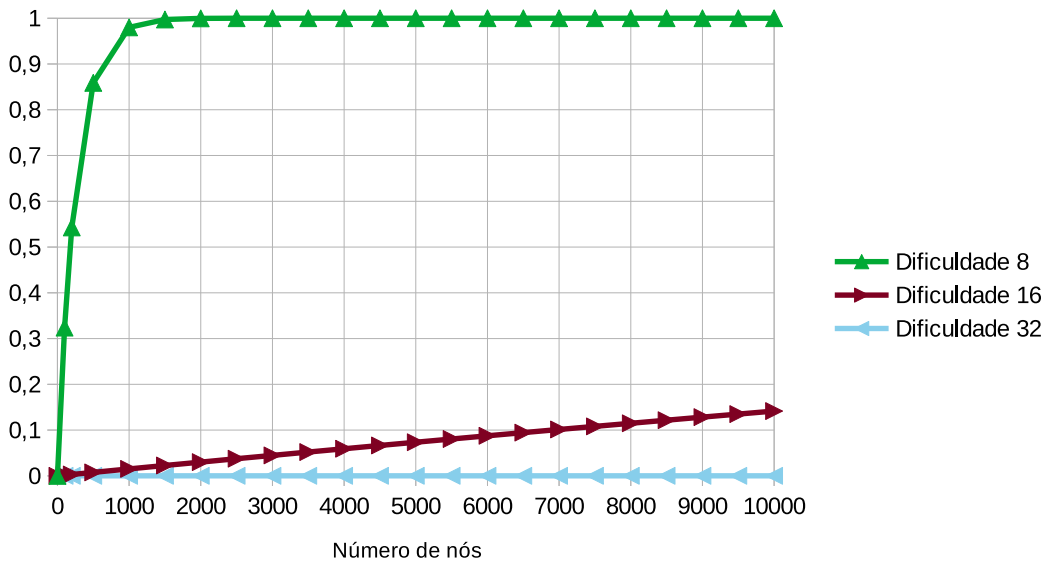


Figura 23 – Probabilidade de sucesso no sorteio em função do número de nós para 8, 16 e 32 nós

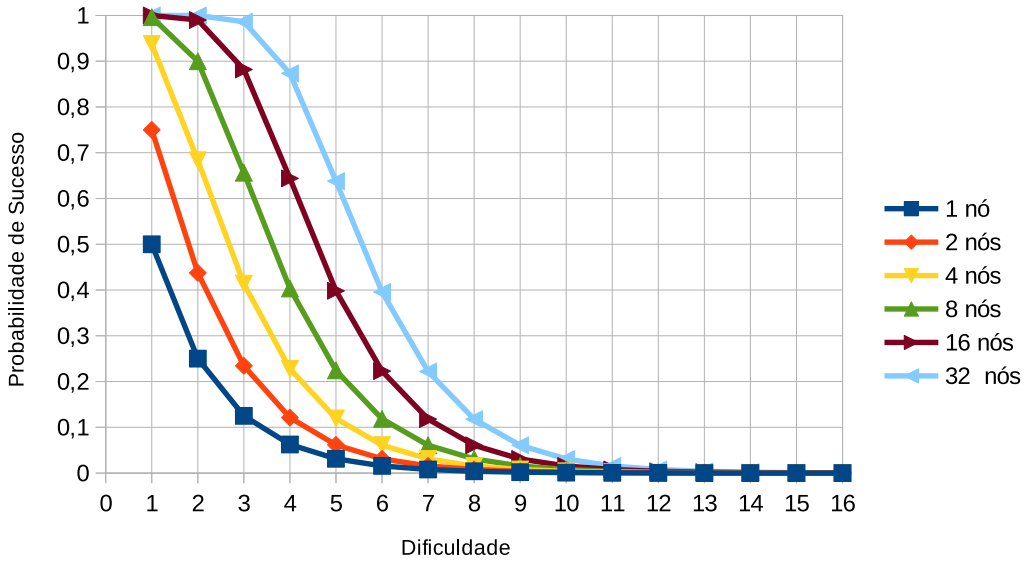


Figura 24 – Probabilidade de sucesso no sorteio em função da dificuldade

em 1.500 *forks* a serem resolvidos. É por este motivo que se faz necessário um controle bem rigoroso do nível de dificuldade a fim de evitar que o número de *forks* aumente.

Assim como ocorre em algumas *blockchains* (como a do *Bitcoin* , por exemplo), neste protocolo também deve ser adotado um esquema automático de ajuste de nível de dificuldade, de maneira a preservar o tempo médio de criação de blocos dentro de um intervalo pré-definido.

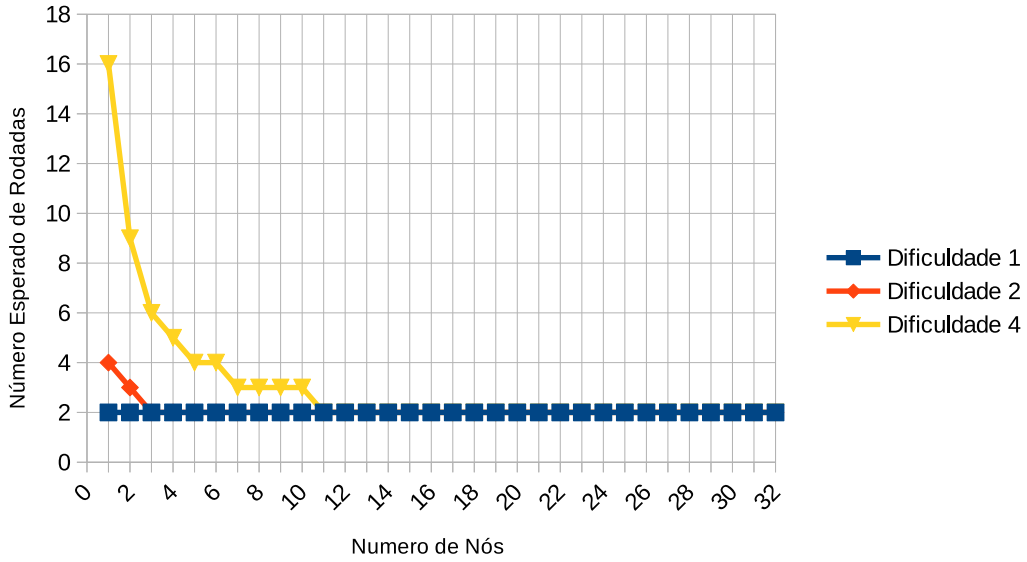


Figura 25 – Número esperado de rodadas até um sucesso em sorteio em função do número de nós para dificuldades de 1 a 4 bits

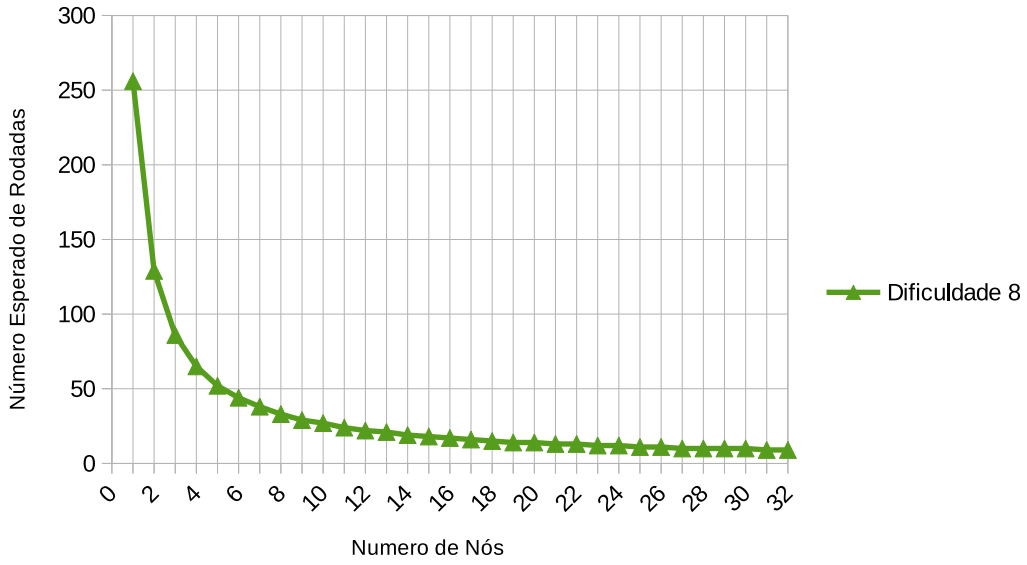


Figura 26 – Número esperado de rodadas até um sucesso em sorteio em função do número de nós para uma dificuldade de 8 bits

Como o tempo necessário para a rede de nós participantes criar pelo menos um bloco é diretamente proporcional ao *timeout* e ao número esperado de rodadas, os gráficos de tempo necessário para criação de um bloco são praticamente idênticos aos de número de rodadas, mudando apenas por um fator de escala igual a *timeout*.

Feitas as análises teóricas e conhecendo-se melhor o comportamento do mecanismo de

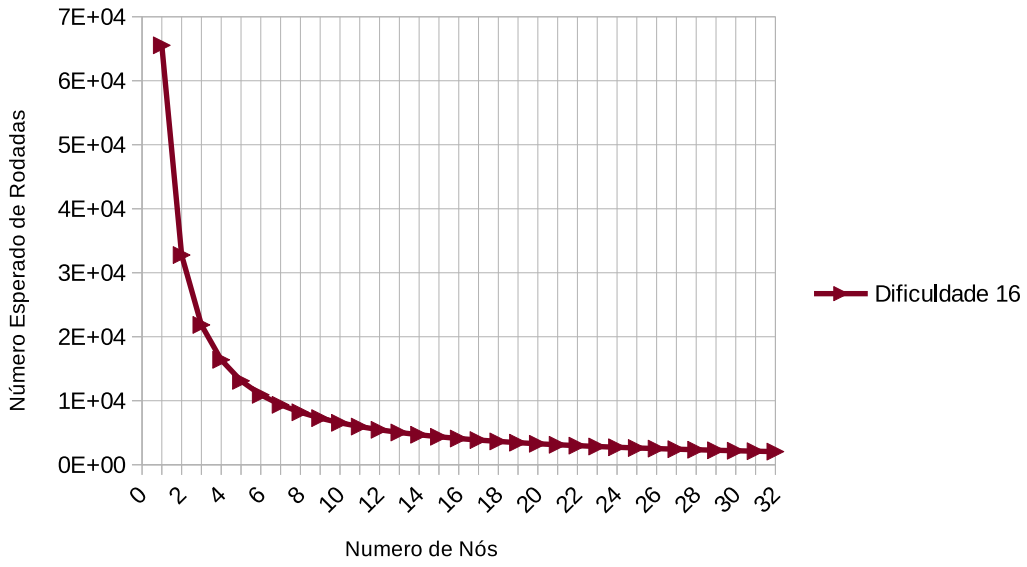


Figura 27 – Número esperado de rodadas até um sucesso em sorteio em função do número de nós para uma dificuldade de 16 bits

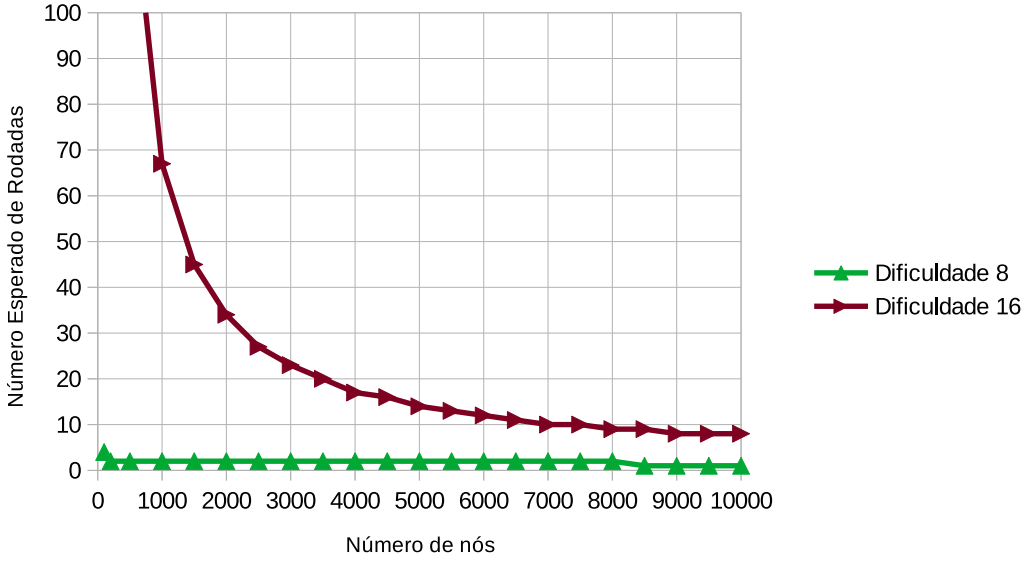


Figura 28 – Número esperado de rodadas até um sucesso em sorteio em função do número de nós para uma dificuldade de 8 e 16 bits

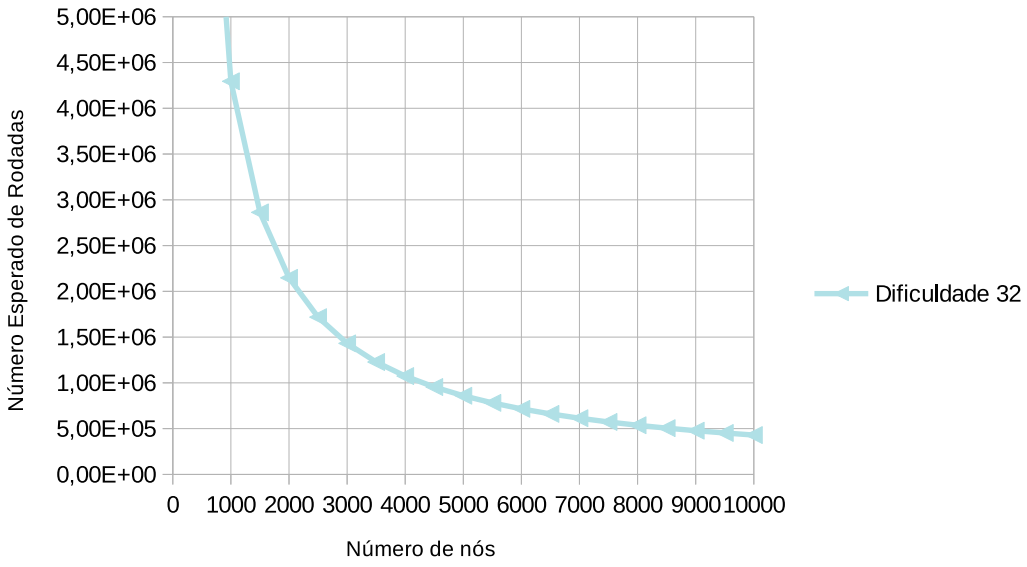


Figura 29 – Número esperado de rodadas até um sucesso em sorteio em função do número de nós para uma dificuldade de 32 bits

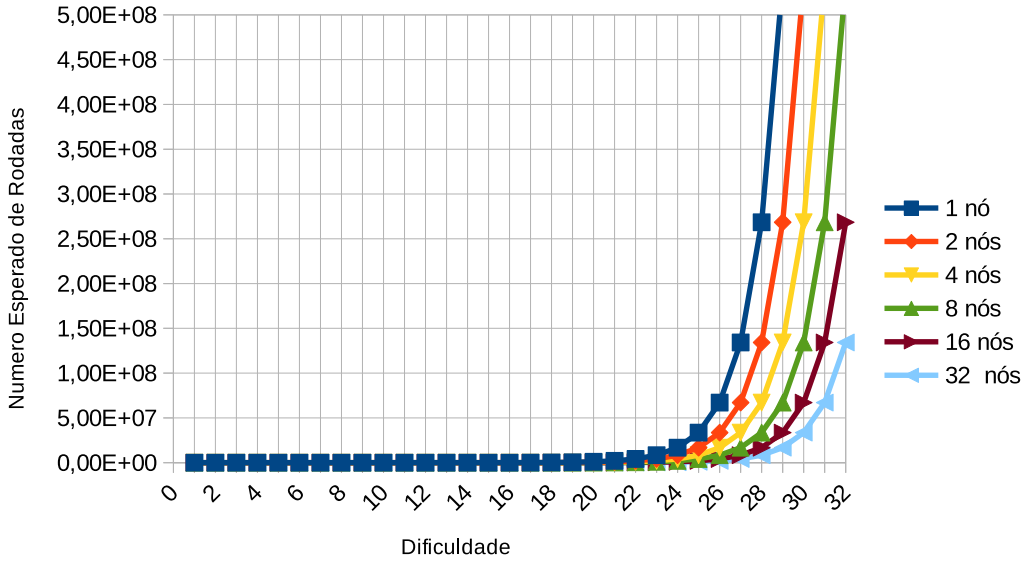


Figura 30 – Número esperado de rodadas até um sucesso em sorteio em função do nível de dificuldade

consenso proposto, passaremos à análise dos aspectos práticos, com base na implementação e execução de um protótipo.

6.2 Resultados práticos

Uma prova de conceito foi desenvolvida em *Python* e a primeira versão do mecanismo proposto foi disponibilizada em (`https://github.com/regras/bc_pos`). A escolha do *Python* se deu por sua sintaxe simples e a ampla disponibilidade de bibliotecas para funções *hash*, criptografia, entre outras. Com base nesta implementação, foram feitos vários testes e colhidos parâmetros de desempenho que estão sendo úteis para melhor avaliar e aprimorar o mecanismo de consenso proposto.

A seguir são apresentados os resultados práticos obtidos até o momento. Os testes foram executados em um servidor de 16 *GB* de memória *RAM* e processador *Intel Xeon* de 2,4 *GHz* com um core. Os nós mineradores foram criados no simulador de redes *Mininet* em sua versão 2.3.0. Na simulação, todos os nós foram criados no mesmo servidor, não sendo utilizado link externo para comunicação com nós remotos.

Devido ao tempo necessário para cada teste e à dificuldade de gerenciar um número grande de nós que formam a *blockchain*, optamos por limitar o tamanho da rede em 10 nós e o *timeout* em 8 segundos nesta avaliação inicial. Avaliações com outras configurações também estão sendo efetuadas e os resultados continuam mostrando-se coerentes com o esperado.

Para um primeiro teste foram medidos os tempos de criação de 100 blocos em cinco execuções por meio de um *script*. A partir dos valores obtidos foi calculado o tempo médio de cada execução. A Tab. 5 detalha este tempo médio para criação de 100 blocos para *timeout* variando de 1 a 8 segundos e redes com 1 a 10 nós.

Nestes testes, foi adotado um nível de dificuldade de dois bits ($D = 2$) e penalidade nula ($P = 0$). O gráfico da Fig. 31 permite avaliar melhor estes tempos e notar que há um crescimento praticamente linear com o aumento do *timeout*, como já era esperado.

Tabela 5 – Tempo médio (s) para criação de 100 blocos após 5 execuções do protocolo para dificuldade $D = 2$ e penalidade $P = 0$.

Timeout (s)	Número de nós							
	1	2	4	6	8	10	100	200
0,5	149,98	111,23	78,17	57,71	41,83	31,24	65,94	55,04
1	413,34	218,73	120,49	78,74	63,79	54,34	115,72	84,98
2	817,71	384,14	211,99	126,48	104,07	76,04	330,93	158,36
4	1579,97	771,71	441,96	254,98	182,10	121,82	543,17	379,37
6	2417,94	1193,73	548,07	368,17	274,78	223,82	967,92	623,44
8	3265,38	1614,75	698,32	530,30	391,19	302,12	1309,08	835,62

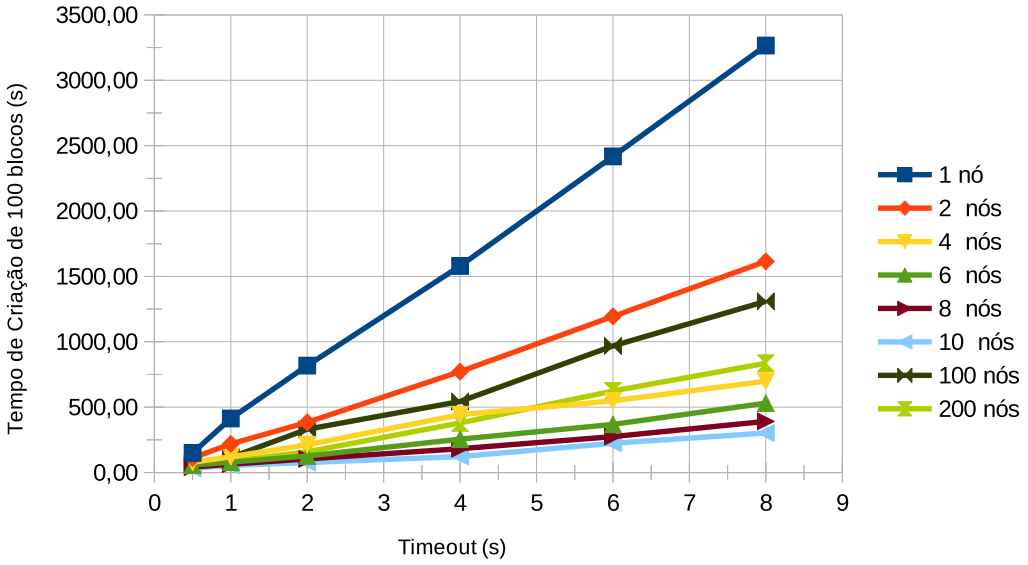


Figura 31 – Tempo médio de criação de 100 blocos em função do *timeout* para $D = 2$ e $P = 0$.

Já a Fig. 32 mostra os mesmos dados de outro ângulo, deixando claro como vai ficando mais fácil (mais rápido) criar 100 blocos à medida que se aumenta o número de nós. A diminuição do tempo de criação não é totalmente vantajosa, já que é preciso estar atento para o aumento da probabilidade de *forks* com o aumento do número de nós.

Porém, como é possível observar nas Figs. 31 e 33 para uma quantidade alta de nós, o tempo de criação é comparável ao de casos com uma quantidade pequena de nós. Isto é devido a que, quando muitos nós estão envolvidos na criação de blocos, o sistema utiliza um tempo adicional com sincronizações.

Na Fig. 33 fazemos uma comparação entre os tempos estimados e a média dos medidos para a criação de 100 blocos. É possível perceber uma diferença maior entre os valores à

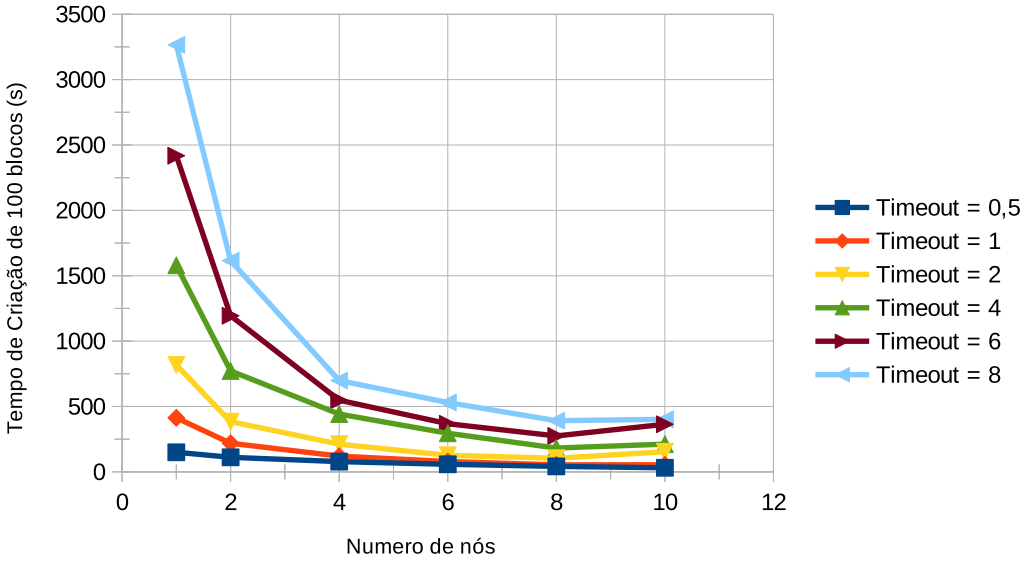


Figura 32 – Tempo médio de criação de 100 blocos em função do número de nós para dificuldade 2 e P=0

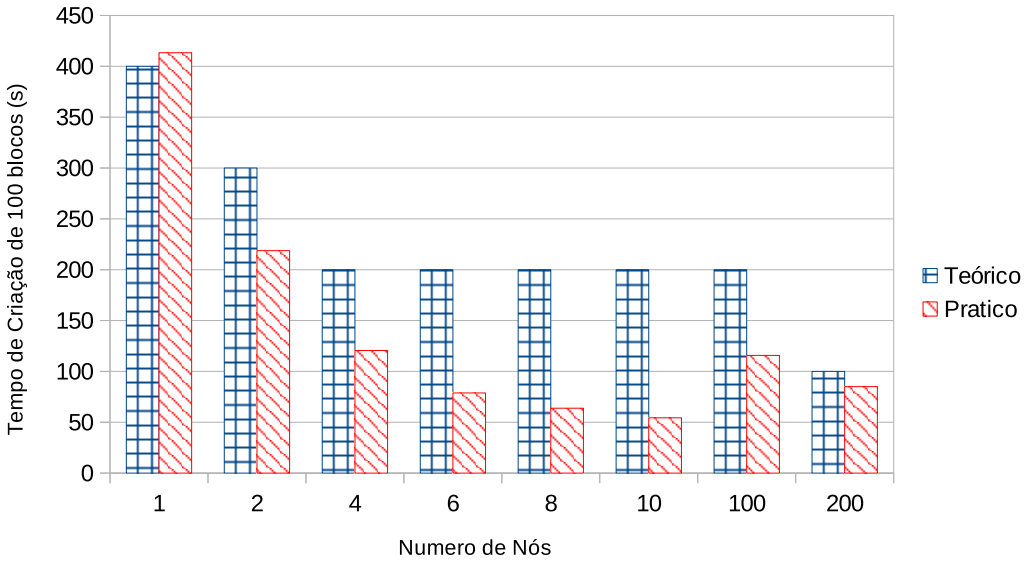


Figura 33 – Comparação entre os tempos estimados e medidos para criação de 100 blocos com dificuldade 2 e *timeout* = 1

medida que aumenta o número de nós e esta diferença se deve principalmente ao fato de que o valor estimado pela Eq. 6.4 para o número de rodadas está arrendondado para o primeiro inteiro acima.

Por um lado, isto nos traz um erro maior quanto menos rodadas forem necessárias para se ter sucesso em um sorteio. Por outro lado, quanto mais nós estiverem participando

dos sorteios, maior é a chance de que algum deles consiga obter sucesso antes do tempo estimado, já que se trata de uma estimativa baseada em probabilidades.

Apesar de que para uma quantidade de 100 e 200 nós o tempo é mais alto do que era esperado, este ainda recai dentro da faixa de valores teóricos.

Portanto, o aumento do número de nós tem um duplo efeito na diferença entre os valores teórico e prático:

- mais nós → mais chance de algum deles conseguir sucesso no sorteio mais rapidamente que o valor esperado teórico;
- mais nós → menos rodadas são necessárias para se obter sucesso, o que proporcionalmente destaca mais as diferenças causadas pelo arredondamento para a próxima rodada.

A Tab. 6 detalha o tempo médio em segundos de cinco execuções do mecanismo de consenso para criação de 100 blocos para *timeout* variando de 1 a 8 segundos e redes com 1 a 10 nós. Nestes testes, foi adotado um nível de dificuldade $D = 4$ bits e penalidade nula ($P = 0$).

Tabela 6 – Tempo médio (s) para criação de 100 blocos após 5 execuções do protocolo para $D = 4$ e $P = 0$

Timeout (s)	Número de nós							
	1	2	4	6	8	10	100	200
0,5	801,78	298,46	189,46	113,12	110,71	137,53	98,62	48,63
1	1582,94	827,96	454,30	259,23	191,93	169,15	188,26	168,88
2	3193,26	1653,83	782,75	549,99	426,12	341,60	329,80	199,70
4	6510,48	3217,43	1561,14	1127,00	775,53	491,90	814,98	354,15
6	9719,18	4760,56	2769,50	1503,04	1238,66	916,83	1281,20	591,01
8	12815,12	6510,16	3148,68	2339,98	1747,60	1588,29	1540,52	745,15

Os dados desta tabela também podem ser melhor entendidos e avaliados por meio dos gráficos nas Figs. 34 e 35, as quais mostram a variação do tempo médio para criação de 100 blocos em função do *timeout* e do número de nós, respectivamente. Como a dificuldade foi dobrada, tornou-se mais difícil a ocorrência de um evento de sucesso no sorteio para mineração de um bloco.

Portanto, mais rodadas e mais tempo se fizeram necessários até que tal evento ocorresse, o que modificou apenas a escala, mas não o formato dos gráficos em relação aos anteriores (dificuldade 2).

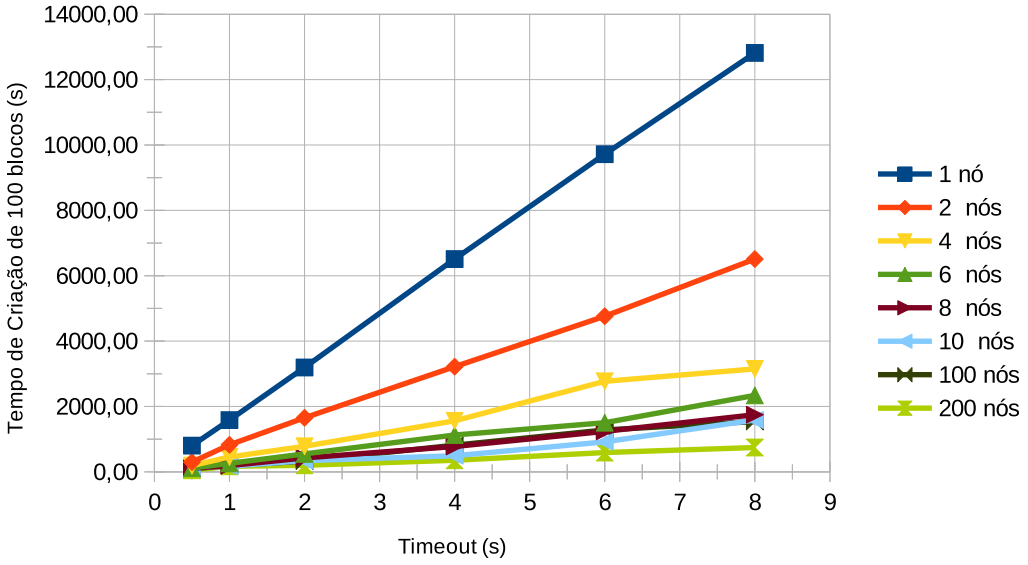


Figura 34 – Tempo médio de criação de 100 blocos em função do *timeout* para $D = 4$ e $P = 0$

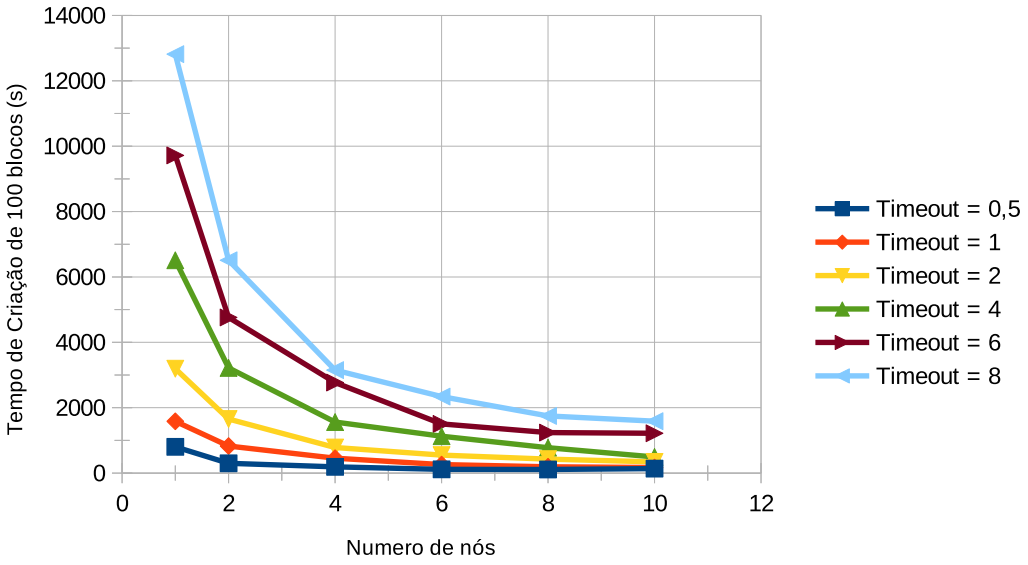


Figura 35 – Tempo médio de criação de 100 blocos em função do número de nós para $D = 4$ e $P = 0$

É interessante notar as mudanças que este aumento de dificuldade trouxe ao comparativo entre tempos estimados e medidos para a criação de 100 blocos. Devido à maior dificuldade no presente caso, o número de rodadas foi aumentado significativamente, o que resultou em um erro proporcional menor entre os valores, como pode ser visto na Fig. 36.

Além disso, é possível também perceber nesta figura, um aumento nas diferenças entre

valores teóricos e práticos à medida que o tempo (e o número de rodadas necessárias) diminui, como já discutido anteriormente.

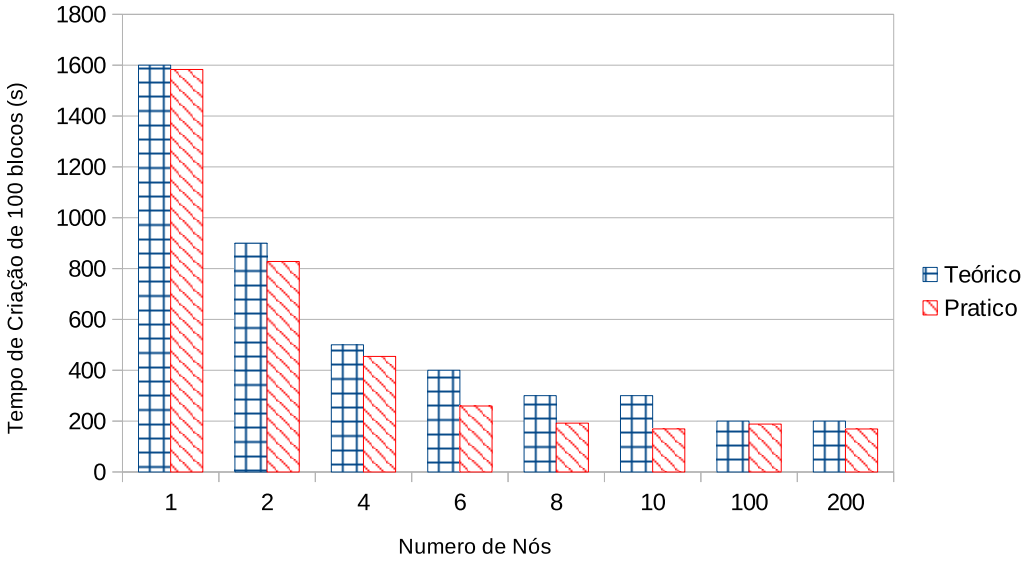


Figura 36 – Comparação entre os tempos estimados e medidos para criação de 100 blocos com dificuldade 4 e $timeout = 1$

As Figs. 37 e 38 permitem uma comparação melhor dos efeitos práticos da mudança de dificuldade de 2 para 4, ao reunir todos os dados medidos em um só local. Foram realizados outros testes práticos, entre eles, um de sincronização de 100 blocos a partir de um nó com todos os outros nós da rede. Os resultados são mostrados na Tab. 7 e na Fig. 39.

Tabela 7 – Tempo médio de sincronização de 100 blocos entre todos os nós da *blockchain*

Núm. de nós	Tempo de sincronização (s)
2	22,16
4	57,18
6	66,37
8	91,48

Nota-se que à medida que a quantidade de nós participantes da rede aumenta, o tempo para que todos estejam sincronizados (com a mesma visão da *blockchain*) também aumenta.

Este é um resultado esperado. Porém, na prática é pouco provável que se tenha a entrada simultânea de muitos nós na rede e, portanto, o processo de sincronização dos blocos que formam a *blockchain* deverá ocorrer de forma distribuída no tempo, já que são distribuídos os eventos de chegadas de novos nós participantes.

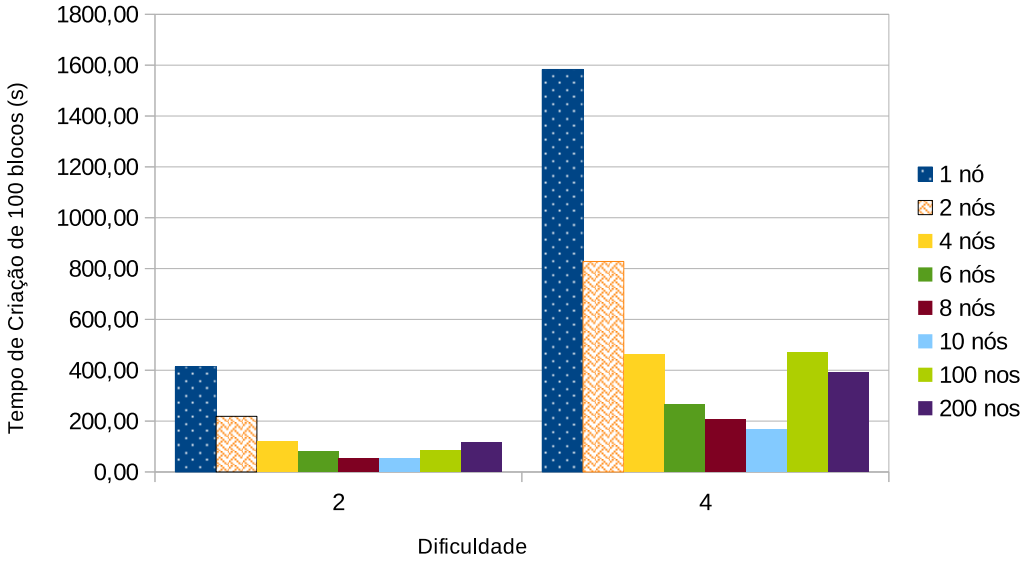


Figura 37 – Tempo médio de criação de 100 blocos em função da dificuldade com $timeout = 1$

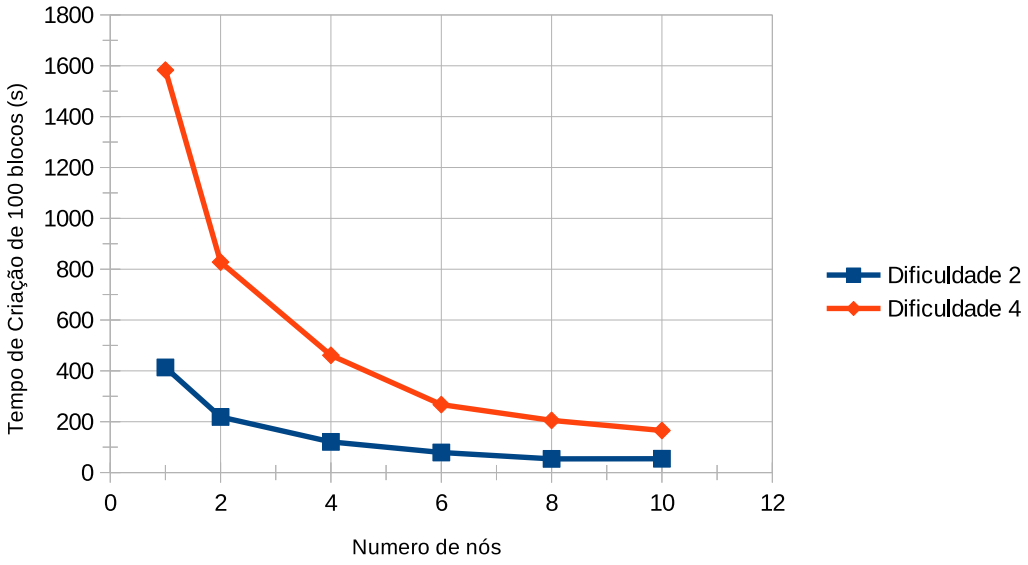


Figura 38 – Tempo médio de criação de 100 blocos em função do número de nós com $timeout = 1$

6.3 Conclusões

A análise teórica e os resultados práticos se mostraram bem consistentes com o que era previsto durante o projeto deste novo mecanismo de consenso. Foi possível mostrar a influência do controle de dificuldade no desempenho do sistema, bem como o impacto que o número de nós traz á velocidade de criação de novos blocos. Deve ser destacado também o

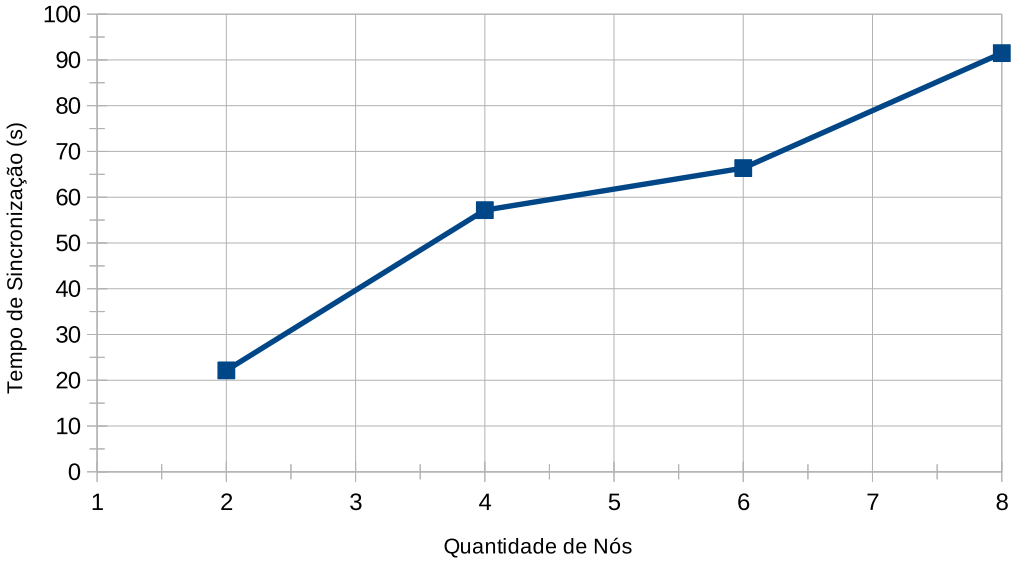


Figura 39 – Tempo médio sincronização de 100 blocos em função do número de nós

fato de haver uma razoável concordância entre os valores teóricos (estimados) e práticos de tempo de criação de um conjunto de blocos.

7 Conclusões

Os mecanismos de consenso são uma parte importante para o funcionamento de uma *blockchain* já que é o processo que permite o acordo entre usuários (possivelmente adversários) para determinar o criador do bloco e prover as demais características que garantem a continuidade e a confiabilidade da *blockchain*. Cada um dos mecanismos avaliados propõe alguma forma de lidar com o problema do consenso.

Como são muitos mecanismos e muitas características distintas, iniciamos uma comparação detalhada entre eles, registrando cada características em uma tabela de consolidação. Assim foi possível destacar melhor os pontos fortes e fracos de cada mecanismo, o que nos trouxe algumas premissas que deveriam ser atendidas em um novo mecanismo de consenso, como o que foi proposto neste trabalho.

Apresenta-se assim um novo mecanismo de consenso baseado em PoS, onde a escolha dos nós é realizada de forma equiprovável durante intervalos de tempo chamados de rodada. Isso permite uma criação controlada de blocos, sem privilegiar os nós que porventura tenham um grande poder de processamento.

Cada nó tenta atingir o objetivo utilizando um *hash* de prova, criado a partir de três informações de entrada que não podem ser alteradas com facilidade, o que dificulta fraudes para passar o desafio.

O protocolo propõe um mecanismo de punição para controlar os *forks*, o qual incentiva a criação de blocos na cadeia principal (com uma maior probabilidade), mas não proíbe a criação em uma outra cadeia que o nó vê como secundária, num determinado momento. Como as visões da *blockchain* e seus *forks* podem ser diferentes em diferentes momentos e em diferentes pontos da rede, é possível que um ramo visto como um *fork* por um determinado nó seja na verdade o ramo principal da *blockchain*. Assim, ao permitir que os nós possam criar blocos para um ramo secundário (ainda que com uma menor probabilidade de sucesso no sorteio), o mecanismo proposto poderá ajudar na melhoria de desempenho de todo o sistema, caso o ramo secundário seja na verdade o principal quando visto por toda a rede.

Foi feito um estudo detalhado do futuro mecanismo PoS do *Ethereum* (*Casper*) e uma comparação dele com o mecanismo proposto. Apesar dos resultados ainda serem preliminares, podemos perceber que o mecanismo proposto é mais simples e poderá trazer vantagens de desempenho e de segurança no futuro.

Foi feita uma implementação simplificada do mecanismo proposto (disponibilizada no

github) e vários testes de desempenho foram feitos com a mesma. Observamos uma boa concordância entre os valores teóricos e práticos de alguns dos parâmetros utilizados para avaliar a implementação e acreditamos estar no caminho certo para alcançar um novo mecanismo de consenso que permita a criação de uma *blockchain* que tenha requisitos mais simples para a inclusão de novos nós na mesma. Isso viabilizará a participação de um número muito maior de interessados no processo de mineração e criação de novos blocos, aumentando a segurança e a tolerância a falhas e a ataques de todo o sistema.

Acreditamos que o principal potencial deste novo mecanismo está na capacidade do mesmo de minimizar a concentração de poder de mineração que se percebe claramente hoje em grandes *blockchains* baseadas em mecanismos PoW (*Proof-of-Work*).

Trabalhos Futuros

Um importante ponto é o desenvolvimento de um esquema automático de ajuste de nível de dificuldade para poder preservar o tempo de criação de blocos dentro de um intervalo previsível. Em um ambiente aberto como as *blockchains* públicas, nota-se a necessidade de um controle automatizado da dificuldade, pois a quantidade de participantes pode variar com o tempo, mudando assim o tempo de criação.

O sistema de controle de *forks*, poderá implementar um sistema de punição adaptativo, onde esta punição dependeria basicamente do tamanho da diferença entre a cadeia principal e a cadeia secundária em questão. Em uma rede globalmente distribuída os atrasos são de difícil previsão e, com isso, punir sempre da mesma forma uma determinada cadeia secundária pode trazer consequências desfavoráveis para uma evolução mais eficiente da *blockchain*.

A construção de um sistema para formação de comitês para controlar a inserção de blocos na *blockchain* poderia trazer benefícios para a rede, como já proposto em vários outros trabalhos relacionados. Entretanto, seguindo os princípios adotados no mecanismo de consenso aqui proposto, a escolha dos membros do grupo deveria ser equiprovável e deveria haver uma certa rotatividade entre tais participantes, o que poderia facilitar o aumento da taxa de criação de blocos sem aumento dos *forks*, já que um nó só teria que verificar as assinaturas dos membros do comitê para verificar a validade de um bloco.

Acreditamos que também poderiam ser avaliadas possibilidades de utilização de alguma variante de PoS dinâmico, mas de tal forma que o uso do valor de *stake* não afetasse de forma significativa as chances de escolha de um nó participante. Isto poderia trazer alguns benefícios ao sistema, por não privilegiar demais aqueles nós que não estão efetivamente comprometidos com o sucesso do mesmo, mas sem contudo prejudicar a premissa básica de

facilitar a participação de todos os que estejam interessados em contribuir com a construção e manutenção da *blockchain* baseada no PoS simplificado com rodadas aqui proposto. Outro benefício seria a minimização de ocorrências de blocos incorretos, já que um nó que atuasse de forma errada seria penalizado com a perdas de suas moedas.

Referências

AGNER, M. *Bitcoin para Programadores*. 1. ed. [S.l.]: Instituto de Tecnologia & Sociedade de Rio, 2016. Citado na página 65.

AKKOYUNLU, E. A.; EKANADHAM, K.; HUBER, R. V. Some constraints and tradeoffs in the design of network communications. *SOSP '75 Proceedings of the fifth ACM symposium on Operating systems principles*, p. 67–74, November 1975. Citado na página 92.

ANTONOPOULOS, A. M. *Mastering Bitcoin: Programming the Open Blockchain*. 2. ed. [S.l.]: O'Reilly Media, Inc., 2017. Citado na página 65.

ARKCREW. *ARK Whitepaper - A Platform for Consumer Adoption*. 2018. [Online]. <https://ark.io/Whitepaper.pdf>. Whitepaper. Citado na página 33.

BACK, A. *HashCash*. 1997. <http://www.cypherspace.org/hashcash/>. (acessado em 14/03/2018). Citado na página 25.

BANO, S.; SONNINO, A.; AL-BASSAM, M.; AZOUVI, S.; MCCORRY, P.; MEIKLEJOHN, S.; DANEZIS, G. Sok: Consensus in the age of blockchains. *ArXiv e-prints*, 2017. Citado 3 vezes nas páginas 16, 28 e 31.

BASHIR, I. *Mastering Blockchain*. 1. ed. [S.l.]: Packt Publishing Ltd., 2017. Citado 2 vezes nas páginas 16 e 23.

BENTOV, I.; LEE, C.; MIZRAHI, A.; ROSENFELD, M. Proof of activity: Extending bitcoin's proof of work via proof of stake. *SIGMETRICS Perform. Eval. Rev.*, v. 42, n. 3, p. 34–37, December 2014. Citado na página 35.

BITSHARE. *Delegated Proof-of-Stake Consensus A robust and flexible consensus protocol*. 2016. <https://bitshares.org/technology/delegated-proof-of-stake-consensus/>. (acessado em 10/09/2017). Citado na página 32.

BITSHARETEAM. *Delegated Proof Of Stake*. 2016. <http://docs.bitshares.org/bitshares/dpos.html>. (acessado em 26/03/2018). Citado na página 33.

BLOCKGEEKS. *What is Ethereum Casper Protocol? Crash Course*. 2018. <https://blockgeeks.com/guides/ethereum-casper/>. (acessado em 08/01/2019). Citado 2 vezes nas páginas 10 e 27.

BLOCK.ONE. *EOS.IO Technical White Paper v2*. 2018. [Online]. <https://github.com/EOSIO/Documentation/blob/master/TechnicalWhitePaper.md>. Whitepaper. Citado na página 33.

BROWN, D. R. L. *SEC 2: Recommended Elliptic Curve Domain Parameters*. 2010. [Online]. <http://www.secg.org/sec2-v2.pdf>. Citado na página 65.

BRYSON, D.; PENNY, D.; GOLDENBERG, D. C.; SERRAO, G. *Blockchain Technology for Government*. 2017. Citado na página 20.

BUCHMANN, J.; DAHMEN, E.; SZYDLO, M. Hash-based digital signature schemes. *First International Workshop on Post-Quantum Cryptography*, 2008. Citado na página 50.

BUTERIN, V. *A Next-Generation Smart Contract and Decentralized Application Platform*. 2013. [Online]. <https://github.com/ethereum/wiki/wiki/White-Paper>. Whitepaper. Citado na página 28.

BUTERIN, V. *Slasher: A Punitive Proof-of-Stake Algorithm*. 2014. <https://blog.ethereum.org/2014/01/15/slasher-a-punitive-proof-of-stake-algorithm/>. (acessado em 27/09/2018). Citado 3 vezes nas páginas 32, 43 e 50.

BUTERIN, V.; GRIFFITH, V. Casper the friendly finality gadget. *ArXiv e-prints*, 2017. Citado 3 vezes nas páginas 32, 43 e 55.

CACHIN, C.; VUKOLIć, M. Blockchain consensus protocols in the wild. *31st International Symposium on Distributed Computing (DISC 2017)*, v. 91, p. 1–16, 2017. Citado na página 49.

CASTRO, M.; LISKOV, B. Practical byzantine fault tolerance. *OSDI '99 Proceedings of the third symposium on Operating systems design and implementation*, p. 173–186, 1999. Citado na página 96.

CHEN, J.; MICALI, S. Algorand. *ArXiv e-prints*, 2017. Citado na página 37.

COBLEE. *Litecoin - a lite version of Bitcoin. Launched!* 2011. [Online]. <https://bitcointalk.org/index.php?topic=47417.0>. Whitepaper. Citado na página 28.

COMMUNITY, N. *Nxt WhitePaper*. 2014. [Online]. <https://bravenewcoin.com/assets/Whitepapers/NxtWhitepaper-v122-rev4.pdf>. Whitepaper. Citado 3 vezes nas páginas 29, 31 e 32.

COULORIS, G.; DOLLIMORE, J.; KINDBERG, T. *Sistemas Distribuídos: Conceitos e Projetos*. 4. ed. [S.l.]: Bookman Editora Ltda, 2007. Citado na página 24.

COULORIS, G.; DOLLIMORE, J.; KINDBERG, T.; BLAIR, G. *Sistemas Distribuídos: Conceitos e Projetos*. 5. ed. [S.l.]: Bookman Editora Ltda, 2013. Citado na página 18.

DWORK, C.; NAOR, M. Pricing via processing or combatting junk mail. *CRYPTO 1992: Advances in Cryptology - CRYPTO' 92*, v. 740, p. 139–147, 1992. Citado na página 25.

FAN, L.; ZHOU, H.-S. A scalable proof-of-stake blockchain in the open setting* (or, how to mimic nakamoto's design via proof-of-stake). *IACR e-print*, 2017. Citado na página 52.

GERVAIS, A.; KARAME, G. O.; WüST, K.; GLYKANTZIS, V.; RITZDORF, H.; CAPKUN, S. On the security and performance of proof of work blockchains. *23rd ACM Conference on Computer and Communications Security*, 2016. Citado na página 54.

GRAMOLI, V. From blockchain consensus back to byzantine consensus. *Future Generation Computer Systems*, 2017. Citado na página 25.

GREENFIELDIV, R. *Explaining How Proof of Stake, Proof of Work, Hashing and Blockchain Work Together*. 2017. <https://medium.com/@robertgreenfieldiv/explaining-proof-of-stake-f1eae6feb26f>. (acessado em 18/02/2018). Citado na página 31.

GREVE, F.; SAMPAIO, L.; ABIJAUDE, J.; COUTINHO, A.; VALCY Ítalo; QUEIROZ, S. Blockchain e a revolução do consenso sob demanda. *XXXVI Simpósio Brasileiro de Redes de Computadores e Sistemas Distribuídos*, Minicurso, 2018. Citado 3 vezes nas páginas 18, 19 e 21.

HARPER, C. *Could Proof of Stake Eliminate Bitcoin's Energy Costs?* 2017. [Online]. <https://coincentral.com/could-proof-of-stake-mend-bitcoins-energy-costs/>. (acessado em 26/04/2019). Citado na página 50.

HOLTHAUS, E. *Bitcoin Mining Guzzles Energy and its Carbon Footprint Just Keeps Growing*. 2018. <https://www.wired.com/story/bitcoin-mining-guzzles-energyand-its-carbon-footprint-just-keeps-growing/>. (acessado em 07/02/2019). Citado na página 50.

JEPSON, C. *DTB001: Decred Technical Brief*. 2015. [Online]. <https://cryptorating.eu/whitepapers/Decred/decred.pdf>. Whitepaper. Citado na página 37.

JOSHI, A. P.; HAN, M.; WANG, Y. A survey on security and privacy issues of blockchain technology. *Mathematical Foundations of Computing*, 2018. Citado na página 49.

KIAYIAS, A.; RUSSELL, A.; DAVID, B.; OLIYNYKOV, R. Ouroboros: A provably secure proof-of-stake blockchain protocol. *Advances in Cryptology – CRYPTO 2017*, 2017. Citado na página 32.

KING, S. *Primecoin: Cryptocurrency with Prime Number Proof-of-Work*. 2013. [Online]. <http://primecoin.io/bin/primecoin-paper.pdf>. Whitepaper. Citado 5 vezes nas páginas 28, 30, 32, 66 e 68.

KING, S.; NADAL, S. *PPCoin: Peer-to-Peer Crypto-Currency with Proof-of-Stake*. 2012. [Online]. <https://peercoin.net/assets/paper/peercoin-paper.pdf>. Whitepaper. Citado 3 vezes nas páginas 28, 34 e 43.

KORDEK, M.; BEDDOWS, O. *Lisk Whitepaper*. 2016. [Online]. <https://github.com/slasheks/lisk-whitepaper/blob/development/LiskWhitepaper.md>. Whitepaper. Citado na página 33.

LAMPORT, L.; SHOSTAK, R.; PEASE, M. The byzantine generals problem. *ACM Transactions on Programming Languages and Systems (TOPLAS)*, v. 4, n. 3, p. 382–401, July 1982. Citado 3 vezes nas páginas 92, 93 e 95.

LARIMER, D. *Delegated Proof-of-Stake (DPOS)*. 2014. <https://steemit.com/bitshares/@testz/bitshares-history-delegated-proof-of-stake-dpos>. (acessado em 09/03/2018). Citado 2 vezes nas páginas 32 e 33.

LI, X.; JIANG, P.; CHEN, T.; LUO, X.; WEN, Q. A survey on the security of blockchain systems. *ArXiv e-prints*, 2018. Citado na página 49.

LIN, I.-C.; LIAO, T.-C. A survey of blockchain security issues and challenges. *International Journal of Network Security*, vol 19, n. num 5, p. pp.653–659, September 2017. Citado 4 vezes nas páginas 19, 22, 49 e 66.

LUU, L.; TEUTSCH, J.; KULKARNI, R.; SAXENA, P. Demystifying incentives in the consensus computer. *Conference on Computer and Communications Security'15 Proceedings of the 22nd ACM SIGSAC*, 2015. Citado na página 50.

MICALI, S. Byzantine agreement, made trivial. *ArXiv e-prints*, 2017. Citado na página 37.

MORA, C.; ROLLINS, R. L.; TALADAY, K.; KANTAR, M. B.; CHOCK, M. K.; SHIMADA, M.; FRANKLIN, E. C. Bitcoin emissions alone could push global warming above 2 c. *Nature Climate Change*, p. 924–936, 2018. Citado na página 50.

NAKAMOTO, S. *Bitcoin: A Peer-to-Peer Electronic Cash System*. 2009. [Online]. <https://bitcoin.org/bitcoin.pdf>. Whitepaper. Citado 2 vezes nas páginas 25 e 28.

NAQVI, S. J. *Delegated PoS vs Proof of Work*. 2017. <https://medium.com/karachain/delegated-pos-vs-proof-of-work-b433be6aef60>. (acessado em 26/03/2018). Citado na página 32.

P4TITAN. *Slimcoin A Peer-to-Peer Crypto-Currency with Proof-of-Burn "Mining without Powerful Hardware"*. 2014. [Online]. <https://github.com/slimcoin-project/slimcoin-project.github.io/raw/master/whitepaperSLM.pdf>. Whitepaper. Citado 2 vezes nas páginas 34 e 35.

PAAR, C.; PELZL, J. *Understanding Cryptography: A Textbook for Students and Practitioners*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2010. Citado na página 65.

PEREZ-SOLA, C.; HERRERA-JOANCOMARTI, J. Bitcoins y el problema de los generales bizantinos. *XIII RECSI*, 2014. Citado 2 vezes nas páginas 18 e 20.

QUANTUMMECHANIC. *Merkle Signature Schemes, Merkle Trees and Their Cryptanalysis*. 2011. <https://bitcointalk.org/index.php?topic=27787.0>. (acessado em 24/03/2018). Citado na página 28.

ROSIC, A. *Proof-of-Work Vs Proof-of-Stake: Basic Mining Guide*. 2017. <https://blockgeeks.com/guides/proof-of-work-vs-proof-of-stake/>. (acessado em 13/11/2017). Citado na página 26.

SALEH, F. Blockchain without waste: Proof-of-stake. *SSRN Electronic Journal*, 2018. Citado na página 50.

SANKAR, L. S.; SINDHU, M.; SETHUMADHAVAN, M. Survey of consensus protocols on blockchain applications. *4th International Conference on Advanced Computing and Communication Systems (ICACCS)*, p. 1–5, 2017. Citado 2 vezes nas páginas 19 e 49.

SCOTT, N. *Steem - An incentivized, blockchain-based, public content platform*. 2017. [Online]. <https://steem.com/SteemWhitePaper.pdf>. Whitepaper. Citado na página 33.

STALLINGS, W. *Cryptography and Network Security: Principles and Practice*. 7. ed. [S.l.]: Pearson, 2017. Citado na página 65.

SZABO, N. *Bitcoin, what took ye so long?* 2011. <http://unenumerated.blogspot.com.br/2011/05/bitcoin-what-took-ye-so-long.html>. (acessado em 24/03/2018). Citado na página 28.

TANENBAUM, A. S.; STEEN, M. V. *Sistemas Distribuídos princípios e paradigmas*. 2. ed. [S.l.]: Pearson Education do Brasil, 2008. Citado na página 92.

TSCHORSCH, F.; SCHEUERMANN, B. Bitcoin and beyond: A technical survey on decentralized digital currencies. *IEEE Communications Surveys Tutorials*, v. 18, n. 3, p. 2084–2123, thirdquarter 2016. Citado na página 50.

VASIN, P. *BlackCoin's Proof-of-Stake Protocol v2*. 2013. [Online]. <https://blackcoin.co/blackcoin-pos-protocol-v2-whitepaper.pdf>. Whitepaper. Citado 4 vezes nas páginas 29, 31, 32 e 68.

WANG, H.; WANG, Y.; CAO, Z.; LI, Z.; XIONG, G. An overview of blockchain security analysis. *15th International Annual Conference, CNCERT 2018: China Cyber Security Annual Conference*, 2018. Citado na página 65.

WATTENHOFER, R. *The Science of the Blockchain*. 2. ed. [S.l.]: CreateSpace Independent Publishing Platform, 2016. Citado 2 vezes nas páginas 24 e 92.

ZHENG, Z.; XIE, S.; DAI, H.; CHEN, X.; WANG, H. An overview of blockchain technology: Architecture, consensus, and future trends. *2017 IEEE International Congress on Big Data (BigData Congress)*, p. 557–564, 2017. Citado 2 vezes nas páginas 35 e 97.

ZHENG, Z.; XIE, S.; DAI, H.-N.; CHEN, X.; WANG, H. Blockchain challenges and opportunities: A survey. *International Journal of Web and Grid Services (IJWGS)*, 2017. Citado 2 vezes nas páginas 16 e 18.

Anexos

ANEXO A – Problema dos Generais Bizantinos

É um problema teórico na área de sistemas distribuídos, no campo dos protocolos de consenso e sistemas tolerantes a falhas. Foi definido por Leslie Lamport, Marshall Pease e Robert Shostak (LAMPORT *et al.*, 1982) da seguinte forma:

Imaginemos que várias divisões do exército bizantino estão acampadas fora de uma cidade inimiga, cada divisão comandada por seu próprio general. Os generais podem se comunicar com outro só por um mensageiro. Depois, observando o inimigo, eles devem decidir sobre um plano comum de ação. Porém, alguns dos generais podem ser traidores, tentando evitar que os generais leais chegam ao acordo. Os generais devem ter um algoritmo que garanta que:

A. *todos os generais leais decidam o mesmo plano de ação: atacar ou não a cidade inimiga. Ou todos os generais leais atacam ou todos desistem do ataque;*

B. *um pequeno número de traidores não possam fazer com que os generais leais adotem um plano ruim, ou seja, só parte deles decida atacar.*

Wattenhoffer (WATTENHOFER, 2016) menciona que este problema é uma generalização do problema dos dois generais proposto por E. A. Akkoyunlu, K. Ekanadham e R. V. Hubert (AKKOYUNLU *et al.*, 1975) onde ele mostra que é impossível chegar a um consenso entre dois participantes.

Na Fig. 40 mostramos um exemplo com seis generais ($G1, ..., G6$) tentando chegar a um consenso para atacar um castelo.

Tanenbaum e Van Steen (TANENBAUM; STEEN, 2008) mencionam que o Lamport et.al limitam sua proposta a processos síncronos, a mensagens *unicast*, à consideração de que a ordenação é preservada e com atraso de comunicação limitado.

Os parâmetros utilizados para descrever as soluções foram idealizados por Lamport et.al (LAMPORT *et al.*, 1982), tal que N se define como o número de processos, i como o processo, k como o número de processos com Falha Bizantina e v_i como o valor fornecido pelo processo i para os outros processos.

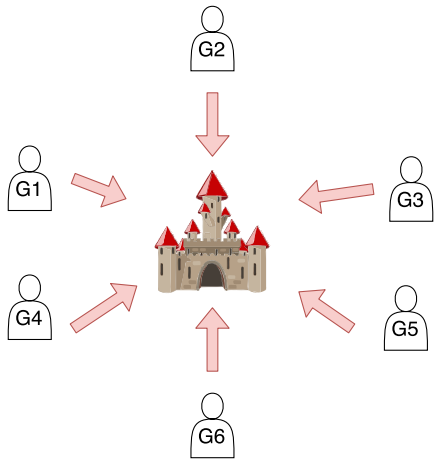


Figura 40 – Problema dos Generais Bizantinos

O objetivo é permitir que cada processo construa um vetor V de comprimento N tal que,se o processo i não for faltoso, $V[i] = v_i$ caso contrario. $V[i]$ é indefinido. Considera-se que há no máximo k processos faltosos.

A.1 Solução com Mensagens Orais

É a primeira proposta de solução descrita no artigo original (LAMPORT *et al.*, 1982), onde se propõe que cada general (processo) envie informações para outros de tal forma que a maioria chegue a um mesmo resultado.

A solução consiste nos seguintes passos: cada processo i fornece um valor v_i para os demais; com cada valor recebido, o processo cria um vetor; em seguida, cada processo compartilha seu vetor com os outros; finalmente se aplica a função de cálculo de maioria que consiste em fazer um análise de cada v_i nos vetores, buscando as coincidências ou a maioria de valores coincidentes.

Para que este algoritmo trabalhe de forma adequada, são necessárias as seguintes suposições:

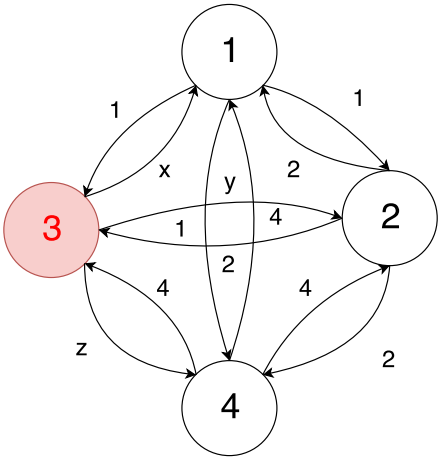
S1. toda mensagem enviada é entregue corretamente;

S2. o receptor de uma mensagem conhece quem a enviou;

S3. a ausência de uma mensagem pode ser detectada.

Exemplo: para $N = 4$ e $k = 1$

Cada processo fornece um valor v_i para os outros, que neste caso será seu número; porém o processo 3 mente e envia um valor x, y e z para os outros processos.



Logo com cada valor da etapa 1 cria-se um vetor com os valores que cada processo recebeu

$$\begin{aligned} 1 \text{ obtém } V_1 &= (1, 2, x, 4) \\ 2 \text{ obtém } V_2 &= (1, 2, y, 4) \\ 3 \text{ obtém } V_3 &= (1, 2, 3, 4) \\ 4 \text{ obtém } V_4 &= (1, 2, z, 4) \end{aligned}$$

Depois cada processo compartilha seu vetor com os outros. Porém, como o processo 3 é desonesto, ele compartilhará 3 vetores falsos com os seguintes valores a,b,c,d,e, f, g, h, i , j, k e l.

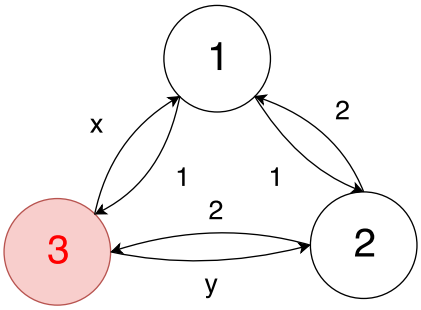
1 obtém	2 obtém	3 obtém
2 $(1, 2, y, 4)$	1 $(1, 2, x, 4)$	1 $(1, 2, x, 4)$
3 (a, b, c, d)	3 (e, f, g, h)	2 $(1, 2, y, 4)$
4 $(1, 2, z, 4)$	4 $(1, 2, z, 4)$	3 (i, j, k, l)

O processo examina o i-ésimo elemento de cada um dos vetores que acabou de receber e, se estiver na maioria, ele é colocado no vetor de resultado: $R = (1, 2, 4)$, o que significa que os processos 1, 2 e 4 chegaram a um acordo.

Esta solução possui um problema conhecido como impossibilidade de consenso com três nós, já que ela precisa de pelo menos $3k+1$ processos (generais), onde $k \geq 1$ é o número de processos falhos (generais traidores).

Exemplo: para $N = 3$ e $k = 1$

Cada processo fornece um valor v_i para os outros, que, neste caso, será seu número. Porém o processo 3 mente e envia um valor x e y para os outros processos.



Logo com cada valor da etapa 1 cria-se um vetor com os valores que que cada processo recebeu.

$$\begin{aligned} 1 \text{ obtém } V_1 &= (1, 2, x) \\ 2 \text{ obtém } V_2 &= (1, 2, y) \\ 3 \text{ obtém } V_3 &= (1, 2, 3) \end{aligned}$$

Depois cada processo compartilha seu vetor com os outros. Porém, como o processo 3 é desonesto, ele cria e compartilha 2 vetores com os seguintes valores a,b,c,d,e e f.

1 obtém	2 obtém
2 $(1, 2, y)$	1 $(1, 2, y)$
3 (a, b, c)	3 (d, e, f)

O processo examina o i-ésimo elemento de cada um dos vetores que acabou de receber e o elemento que estiver em maioria é colocado no vetor de resultado. Porém, neste caso, como nenhum elemento tem maioria, o vetor de resultado será: R = DESCONHECIDO, o que significa que não se conseguiu um acordo.

A.2 Solução com Mensagens Assinadas

Esta é a segunda solução também descrita no artigo original (LAMPORT *et al.*, 1982), que permite chegar a um consenso com apenas $3k$ nós, enviando mensagens assinadas e que não podem ser falsificadas.

O algoritmo consiste em um general enviando uma ordem assinada para os outros e cada um dos outros generais insere sua assinatura na ordem e a reenvia para outros generais.

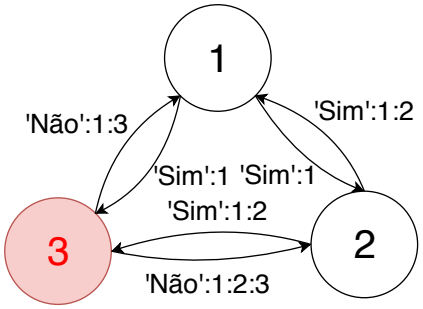
Para que o algoritmo trabalhe de forma adequada, é necessário fazer outras suposições além das anteriores.

S4. A assinatura de um general leal não pode ser forjada.

S5. Qualquer alteração do conteúdo da mensagem assinada pode ser detectada e a mensagem descartada.

S6. Qualquer um pode verificar a autenticidade da assinatura do general.

Exemplo: para $N = 3$ e $k = 1$



O processo 1 envia um valor para os outros processos como, por exemplo, o valor "Sim", junto com sua assinatura.

Os demais processos, depois de receber o valor, devem reenviá-lo, concatenando sua assinatura à assinatura do processo anterior.

$$\begin{aligned} 1 \text{ obtém } V_1 &= (\text{'}Sim\text{'},\text{'}N\tilde{a}o\text{'}) \\ 2 \text{ obtém } V_2 &= (\text{'}Sim\text{'},\text{'}N\tilde{a}o\text{'}) \\ 3 \text{ obtém } V_3 &= (\text{'}Sim\text{'},\text{'}Sim\text{'}) \end{aligned}$$

Em seguida os processos compartilham com os demais os dados recebidos.

1 obtém	2 obtém
$2\ \text{'}Sim\text{'} : 1 : 2$	$1\ \text{'}Sim\text{'} : 1$
$3\ \text{'}N\tilde{a}o\text{'} : 1 : 3$	$3\ \text{'}N\tilde{a}o\text{'} : 1 : 3$

Neste exemplo as assinaturas de 1 e 2 estão associadas aos mesmos valores corretos, mas a assinatura do processo 3 não. Como o valor incorreto aparece associado à assinatura 3, os demais processos detectam que este é o traidor.

A.3 *Practical Bizantine Fault Tolerance* (PBFT)

A primeira solução prática ao Problema dos Generais Bizantinos foi proposta por Miguel Castro e Barbara Liskov em 1999 (CASTRO; LISKOV, 1999). Tal solução considera

um máximo de 1/3 de nós maliciosos bizantinos e implementa uma variação da solução de mensagens assinadas.

Todo o processo está divido em três fases: pré-preparar, preparar, restabelecer (*commit*). Em cada fase um nó deve receber os votos de 2/3 de todos os nós para poder passar à fase seguinte (ZHENG *et al.*, 2017a).

Este algoritmo está feito para sistemas distribuídos com pelo menos um elemento central, já que depende de um processo líder e precisa conhecer todos os nós da rede. Por causa do controle centralizado e necessidade de conhecimento de toda a rede, algumas *blockchains* privadas utilizam esta solução (ou uma variação dela) como mecanismo de consenso.

ANEXO B – Algoritmos Auxiliares

B.1 Funções de Verificação

B.1.1 Função Atende Desafio

A função atende desafio verifica se o bloco superou o desafio, ou seja, teve sucesso no sorteio.

Algoritmo B.1 função atende_desafio

Entrada: *bloco*, *target*
Saída: verdadeiro, falso
1. **se** $bloco.hash < target$ **então**
2. **retorna verdadeiro**
3. **senão**
4. **retorna falso**
5. **fim se**

B.1.2 Função Bloco Válido

A função bloco válido verifica se o campo *hash do bloco anterior* é igual ao *hash* do último bloco.

Algoritmo B.2 função bloco_valido

Entrada: *bloco*, *ultimo_bloco*
Saída: verdadeiro, falso
1. **se** $bloco.hash\_anterior = ultimo\_bloco.hash$ **então**
2. **retorna verdadeiro**
3. **senão**
4. **retorna falso**
5. **fim se**

B.1.3 Função Rodada Esperada

A função rodada esperada permite verificar se a rodada de um bloco corresponde à rodada em que supostamente ele deveria estar.

Algoritmo B.3 função rodada_esperada

Entrada: *bloco*, *ultimo_bloco*
Saída: verdadeiro, falso
1. $rodadas\_transcorridas \leftarrow \left\lfloor \frac{bloco.tempo\_chegada - ultimo\_bloco.tempo\_chegada}{TIMEOUT} \right\rfloor$
2. $rodada\_esperada \leftarrow ultimo\_bloco.rodada + rodadas\_transcorridas$
3. **se** $rodada\_esperada = bloco.rodada$ **então**
4. **retorna verdadeiro**
5. **senão**
6. **retorna falso**
7. **fim se**

B.1.4 Função Validar Rodada

A função validar rodada permite verificar se a rodada de um bloco é consistente com a do último bloco da cadeia.

Algoritmo B.4 função rodada_esperada

Entrada: *bloco*, *blockchain*
Saída: verdadeiro, falso
1. $bloco_cadeia \leftarrow$ `obter_ultimo_bloco()`
2. **se** $bloco.indice > bloco\_cadeia.indice$ **e** $bloco.rodada > bloco\_cadeia.rodada$ **então**
3. **retorna verdadeiro**
4. **senão**
5. **retorna falso**
6. **fim se**

B.1.5 Função Cabeçalho Válido

A função cabeçalho válido permite verificar se o cabeçalho do bloco é igual ao calculado pelo *hash* a partir dos dados do bloco.

Algoritmo B.5 função cabeçalho_valido

Entrada: *bloco*
Saída: verdadeiro, falso
1. **se** $bloco.hash =$ *bloco*.`calcBlockhash()` **então**
2. **retorna verdadeiro**
3. **senão**
4. **retorna falso**
5. **fim se**

B.1.6 Função Validar Cadeia

A função validar cadeia permite validar se a cadeia possui algum erro em algum bloco da cadeia, com o objetivo de detectar algum problema existente.

Algoritmo B.6 função validar_cadeia

Entrada: *blockchain*, *cadeia*
Saída: *bloco_error*, *hash_error*

1. *ultimo_bloco* ← `obter_ultimo_bloco()`
2. **para** $i \leftarrow 0$ **até** `long(cadeia)` **faça**
3. **se** `cabeçalho_valido`(*cadeia*[*i*]) é **falso então**
4. **retorna** *cadeia*[*i*], **verdadeiro**
5. **fim se**
6. **se** `bloco_valido`(*bloco*, *ultimo_bloco*) é **verdadeiro então**
7. **se** `atende_desafio`(*b*) é **verdadeiro e** `rodada_valida`(*ultimo_bloco*, *blockchain*) é **verdadeiro então**
8. *ultimo_bloco* ← *cadeia*[*i*]
9. *blockchain* ← *blockchain*||*cadeia*[*i*]
10. **fim se**
11. **senão**
12. **retorna** *cadeia*[*i*], **falso**// Possivél Fork
13. **fim se**
14. **fim para**
15. **retorna** Nulo, **falso**

B.1.7 Função Validação Recursiva

A função validação recursiva permite validar o bloco ate achar o bloco que gera um fork.

Algoritmo B.7 função validação_recursiva

Entrada: *bloco_erro*, *blockchain*
Saída: *novo*

1. *índice* $\leftarrow$ *bloco_erro.índice* $-$ 1
2. *bloco_pedido* $\leftarrow$ *blockchain.cadeia*[*indice*]
3. *tentativas* $\leftarrow$ 3
4. **enquanto** *índice* $>$ 0 **e** *tentativas* $>$ 0 **faça**
5. **imprime** 'Validando índice ' *indice*
6. *novo* $\leftarrow$ `pedir_bloco(indice)`
7. **se** *novo* não é nulo **e** `cabeçalho_valido`(*novo*) é **verdadeiro então**
8. **se** `bloco_valido`(*bloco*, *ultimo_bloco*) é **verdadeiro e** `atende_desafio`(*b*) é **verdadeiro então**
9. **imprime** 'Retornando'
10. **retorna** *novo*
11. **senão**
12. *indice* $\leftarrow$ *indice* $-$ 1
13. *bloco_pedido* $\leftarrow$ *blockchain.cadeia*[*indice*]
14. **fim se**
15. **fim se**
16. **fim enquanto**
17. **retorna** *novo*

ANEXO C – Artigos Publicados

C.1 Artigos publicados derivados desta pesquisa

MAEHARA, Y.; LEAL, V. C.; LUCENA, A. U. de; HENRIQUES, M. A. A. Avaliação de mecanismos de consenso para blockchains em busca de nova estratégia mais eficiente e segura. In: *XVIII Simpósio Brasileiro de Segurança da Informação e de Sistemas Computacionais.* Natal - RN, 2018.

MAEHARA, Y.; MARTINS, D. F. G.; HENRIQUES, M. A. A. Proof-of-stake baseado em tempo discreto. In: *XXXVII Simpósio Brasileiro de Redes de Computadores e Sistemas Distribuídos.* Gramado - RS, 2019.