UNIVERSIDADE ESTADUAL DE CAMPINAS
Faculdade de Engenharia Elétrica e de Computação

Diego Fernandes Gonçalves Martins

Um novo mecanismo de consenso probabilístico para blockchains públicas

A new probabilistic consensus mechanism for public blockchains

Campinas

2021

UNIVERSIDADE ESTADUAL DE CAMPINAS
Faculdade de Engenharia Elétrica e de Computação

Diego Fernandes Gonçalves Martins

Um novo mecanismo de consenso probabilístico para blockchains públicas

A new probabilistic consensus mechanism for public blockchains

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 Diego Fernandes Gonçalves Martins e orientada pelo Prof. Dr. Marco Aurélio Amaral Henriques.

Campinas

2021

Ficha catalográfica
Universidade Estadual de Campinas
Biblioteca da Área de Engenharia e Arquitetura
Rose Meire da Silva - CRB 8/5974

M366n

Martins, Diego Fernandes Gonçalves, 1986-
Um novo mecanismo de consenso probabilístico para blockchains públicas / Diego Fernandes Gonçalves Martins. – Campinas, SP : [s.n.], 2021.

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

1. Cadeia de blocos. 2. Consenso distribuído (computação). 3. Sistemas distribuídos. 4. Bitcoin. 5. Redes de computadores - Protocolos. 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: A new probabilistic consensus mechanism for public blockchains
Palavras-chave em inglês:
Blockchain
Distributed consensus (Computation)
Distributed systems
Bitcoin
Computer networks - Protocols
Área de concentração: Engenharia de Computação
Titulação: Mestre em Engenharia Elétrica
Banca examinadora:
Marco Aurélio Amaral Henriques [Orientador]
Leonardo Barbosa e Oliveira
Raphael Vicente Rosa
Data de defesa: 24-02-2021
Programa de Pós-Graduação: Engenharia Elétrica

Identificação e informações acadêmicas do(a) aluno(a)
- ORCID do autor: https://orcid.org/0000-0003-2524-7846
- Currículo Lattes do autor: http://lattes.cnpq.br/8006833772459468

COMISSÃO JULGADORA - DISSERTAÇÃO DE MESTRADO

Candidato: Diego Fernandes Gonçalves Martins **RA:** 069921
Data da Defesa: 24 de fevereiro de 2021
Título: "Um novo mecanismo de consenso probabilístico para blockchains públicas"

Prof. Dr. Marco Aurélio Amaral Henriques (Presidente, FEEC/UNICAMP)
Prof. Dr. Leonardo Barbosa e Oliveira (DCC/UFMG)
Dr. Raphael Vicente Rosa (FEEC/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

Agradecimentos

- A Deus, pela sua presença constante na minha vida, mesmo nos momentos mais difíceis.
- A minha esposa Cecília e filha Mariana pela ajuda, apoio e muita compreensão. Amo vocês.
- A minha mãe Suzana e meu pai Leonel, principalmente pela ajuda e palavras positivas nas horas mais difíceis.
- A minha irmã Larissa, que mesmo distante colaborou muito com o meu trabalho.
- A minha sogra Maria Eni e meu sogro Nereu pelo apoio durante o trabalho.
- A meu orientador, professor Marco Aurélio Amaral Henriques, pelos importantes ensinamentos, pela paciência e apoio na elaboração deste projeto.
- Aos meus amigos Yoshitomi Eduardo Maehara Aliaga, Antônio Unias de Lucena, Vitor Hugo Galhardo Moia pelas suas contribuições e apoio neste projeto.
- A todos os membros do grupo de pesquisa ReGrAS que de alguma forma colaboraram com esse trabalho.
- Ao IFSP (Instituto Federal de Ciência e Tecnologia de São Paulo) que investiu na minha formação acadêmica, oferecendo o afastamento das minhas atividades como docente para que me dedica-se a este trabalho.
- A RNP (Rede Nacional de Ensino e Pesquisa) e ao projeto Cloud Labs, pela disponibilização de parte dos computadores e redes que foram fundamentais para a execução dos testes realizados durante o trabalho.

"Viva como se fosse morrer amanhã.
Aprenda como se fosse viver para sempre."
(Mahatma Gandhi)

Resumo

A blockchain, introduzida com a invenção da criptomoeda Bitcoin, é considerada uma tecnologia disruptiva, pela sua capacidade em oferecer um ecossistema descentralizado e seguro para que transações entre duas partes sejam realizadas sem que exista um terceiro confiável. Atualmente, as principais blockchains públicas (Bitcoin e Ethereum) utilizam um mecanismo de consenso baseado em PoW (Proof-of-Work ou Prova de Trabalho), o qual é dependente do cálculo de uma grande quantidade de funções hash com um grande gasto energético. Além disso, o PoW é um mecanismo que premia, por meio de recompensas financeiras, apenas o nó que primeiro calcula um hash menor que um valor pré-estabelecido, desperdiçando o esforço e gasto energético de todos os demais nós que tentaram atingir o mesmo objetivo. Estes fatos têm exigido altos investimentos em energia e hardware para aumentar o poder computacional, o que tem afastado os participantes menos privilegiados e, consequentemente, contribuído para uma centralização do mecanismo de consenso em torno de poucos nós que são capazes de realizar os investimentos necessários para se manterem competitivos. Tal centralização não é interessante sob o ponto de vista da segurança do mecanismo, já que ela facilita o conluio entre algumas partes que juntas detenham mais da metade do poder computacional total da rede. Outro problema do mecanismo PoW é que ele é um mecanismo de baixo desempenho no que tange a capacidade de confirmar as transações na blockchain. Estes pontos despertaram o interesse no desenvolvimento de novos mecanismos de consenso que sejam capazes de melhorar o desempenho do PoW, e promover a descentralização do consenso a partir de técnicas que não estejam alicerçadas no poder computacional dos participantes. Uma alternativa já conhecida ao PoW é o mecanismo PoS (Proof-of-Stake ou Prova de Posse), que permite a participação no consenso de qualquer nó que provar a posse de algum valor ou objeto especificado pelo mecanismo. Entretanto, ele ainda não é largamente utilizado por não haver uma forma de implementação que tenha conquistado a confiança dos usuários das blockchains públicas. A partir do estudo e comparação dos principais mecanismos de consenso para blockchains públicas, esse trabalho busca definir os requisitos desejáveis para a construção de um novo mecanismo de consenso baseado em PoS que seja seguro e eficiente. Neste sentido, o trabalho apresenta um novo mecanismo de consenso PoS que é probabilístico e não requer a formação de comitês para que novos blocos sejam confirmados. Apesar de existirem outros consensos baseados em PoS que não formam comitês, eles não são funcionais ou exigem algum tipo de grupo de consenso nos bastidores para suportar os blocos já confirmados e manter o consenso seguro. Assim como ocorre com outros consensos baseados em PoS, o mecanismo proposto também é capaz de reduzir o consumo de energia elétrica quando comparado ao PoW. Além da formalização do mecanismo, o texto apresenta resultados práticos de avaliação de segurança e de desempenho. Por

fim, ele compara o novo mecanismo proposto com alguns dos principais mecanismos de consenso utilizados na prática, de forma a demonstrar seu melhor desempenho, maior robustez e menor consumo de energia em relação aos mesmos.

Palavras-chave: mecanismo de consenso; blockchains; Prova de Posse; consenso probabilístico; Prova de Posse sem comitês.

Abstract

The blockchain, introduced with the invention of the Bitcoin cryptocurrency, is considered a disruptive technology. It offers a decentralized and secure environment for transactions between two parties, built without a trusted third party. The leading public blockchains (Bitcoin and Ethereum) use a consensus mechanism based on PoW (Proof-of-Work), which depends on calculating a large volume of hash functions that spend much energy. Also, PoW is a mechanism that rewards only the node that first calculates a hash smaller than a pre-established value, wasting the efforts and energy spent by all other nodes that tried to achieve the same goal. These facts have demanded high investments in energy and hardware to increase the computational power, which has hampered the participation of less privileged nodes and, consequently, contributed to a centralization of the consensus mechanism around the few nodes having the resources to stay competitive. This centralization is not interesting from the security point of view since it facilitates the collusion between some parties that hold together more than half of the network's total computational power. Another problem is the low performance of the PoW mechanism regarding its ability to confirm transactions on the blockchain. These points increased the interest in developing new consensus mechanisms capable of enhancing the performance of PoW while promoting the decentralization of consensus, using techniques not based on the participants' computational power. A known alternative to PoW is the PoS (Proof-of-Stake) mechanism, which allows the participation in the consensus of any node that proves the possession of some value or object specified by the mechanism. However, this approach is still not widely used because no form of implementation has won public blockchain users' trust. Based on the study and comparison of the main consensus mechanisms for public blockchains, this work defines the desirable requirements for developing a new consensus mechanism based on PoS that is safe and efficient. As a result, the work presents a new PoS consensus mechanism that is probabilistic and does not require a committee to confirm new blocks. Although there are other consensus proposals based on PoS that do not require committees, they are nonfunctional or require some kind of agreement group to support the confirmed blocks and achieve a safe consensus. Moreover, similar to other mechanisms based on PoS, the proposed consensus can reduce electricity consumption compared to PoW. The text shows the formalization of the mechanism and practical results regarding safety and performance evaluation. It then shows a comparison with some of the main consensus mechanisms used in practice in order to demonstrate its higher performance, stronger robustness, and lower power consumption relative to them.

Keywords: consensus mechanism; blockchains; Proof-of-Stake; probabilistic consensus; committeeless Proof-of-Stake.

Lista de ilustrações

Lista de tabelas

Sumário

1 Introdução

As blockchains públicas têm atraído muita atenção principalmente depois da popularização das criptomoedas. A blockchain pode ser vista como um livro razão distribuído entre todos os participantes da rede, onde as transações inseridas neste livro são imutáveis, após decorrido um tempo de confirmação. Uma das partes mais importantes de uma blockchain é o mecanismo de consenso utilizado, que permite que partes desconhecidas entre si cheguem a um acordo sobre o novo estado dos dados armazenados. O mecanismo de consenso é definido por Bashir (BASHIR, 2018) como um conjunto de passos que são dados pelos nós que compõem a rede para entrarem em acordo a respeito de um valor.

As blockchains públicas, ou seja, aquelas onde não há restrições sobre quem pode verificar transações ou criar blocos (LIN; LIAO, 2017), necessitam de mecanismos de consenso eficientes já que os desafios para esse tipo de estrutura são bastante complexos, principalmente os relacionados à presença de nós desonestos (NGUYEN; KIM, 2018). Além disso, esse tipo de rede é definido sobre uma infraestrutura totalmente assíncrona, que utiliza conexões tradicionais com a internet como forma de interconectar os diferentes *peers*. Fischer et al. (FISCHER; LYNCH; PATERSON, 1985) mostrou que em uma rede, onde não há limites temporais para a transmissão de mensagens (sistema assíncrono), o consenso não pode ser alcançado de maneira determinística se uma única falha ocasionada pela interrupção de algum nó ocorrer (mesmo que não existam nós desonestos). Assim, os mecanismos de consenso para blockchains públicas podem restringir a operação dos nós que compõem a rede, buscando um grau de sincronismo para que seja possível criar mecanismos de consenso determinísticos, ou trabalhar com consensos probabilísticos.

Os consensos determinísticos são propostas derivadas do trabalho de Lamport et al. (LAMPORT; SHOSTAK; PEASE, 1982) e buscam resolver o problema teórico dos Generais Byzantinos (Anexo D). Segundo Lamport et al., só é possível atingir o consenso se, no máximo, um terço dos nós são desonestos. Além disso, é necessário considerar que a rede possua algum tipo de sincronismo, onde seja possível admitir que os nós honestos recebam as mensagens dentro de um limite de tempo conhecido. Essa categoria de mecanismos resolve o problema dos generais Byzantinos com a possibilidade de ocorrerem falhas ou de existirem nós desonestos (*Fault Byzantine Tolerance* BFT). Deste modo, é comum que eles utilizem esquemas de votação que contam com diferentes tipos de implementações.

Por sua vez, os consensos probabilísticos não garantem que a rede tenha concordância sobre um valor de forma instantânea; porém mecanismos são definidos para que, ao longo do tempo, as decisões dos nós honestos sejam em direção a um único objetivo (GREVE et al., 2018). Os consensos probabilísticos também são amplamente utilizados

em blockchains públicas. O principal mecanismo desta categoria é o Proof-of-Work (PoW), onde os nós buscam criar o próximo bloco através dos seus esforços computacionais, que são utilizados para encontrar um valor de hash que atenda um desafio proposto (NAKAMOTO, 2009). Uma desvantagem do PoW é que ele está ligado ao desperdício de energia elétrica em sua operação, já que todos os nós precisam trabalhar com alto desempenho e apenas o trabalho de um deles será aproveitado (nó que primeiro encontrar o hash capaz de atender o desafio).

Outro ponto importante, é que o PoW acaba privilegiando os nós que possuem mais recursos computacionais disponíveis. Esses nós dispõem de recursos financeiros para investir em melhores hardwares, com desempenhos muito maiores quando comparados com processadores de propósito geral. Isso aumenta o poder de decisão destes nós no consenso, pois eles são capazes de decidir na maioria das vezes quais são os próximos blocos da blockchain. Desta forma, é possível, que estes nós criem um conluio e executem um ataque conhecido como Ataque dos 51%, onde nós abastados formam um grupo que detém a maioria do poder computacional e, assim, conseguem manipular alguns consensos de forma a tirar proveito de algum resultado de interesse do grupo (TSCHORSCH; SCHEUERMANN, 2016). Esses fatos despertaram o interesse por outros mecanismos de consenso e, entre as abordagens existentes destaca-se o Proof-of-stake (PoS), onde a prova de posse de um *stake* (valor ou objeto) é utilizada pelo nó para ganhar o direito de tentar criar novos blocos e receber recompensas pelo seu trabalho (KING; NADAL, 2012). A prova de posse do *stake* consiste em utilizar recursos do próprio nó, como por exemplo, a quantidade de moedas associada a um endereço de seu controle, como garantia de que ele realmente pode participar do processo de criação de novos blocos.

No presente trabalho são apresentados os principais mecanismos de consenso para blockchains públicas que implementam o consenso a partir de estratégias probabilísticas e determinísticas. Este estudo norteou o desenvolvimento de um novo mecanismo de consenso, que possui algumas características dos mecanismos baseados em *Proof-of-Stake*, além de uma nova abordagem probabilística, utilizada na confirmação de blocos sem que comitês de confirmação sejam necessários. Ao não requerer a utilização desses comitês, o mecanismo proposto torna-se mais simples no que tange a confirmação dos blocos, pois passa a não depender da convergência dos votos que são emitidos por esses comitês. Assim, esse mecanismo é inovador, já que outros consensos baseados em PoS que não implementam comitês são não funcionais ou acabam implementando algum tipo de comitê nos bastidores, para garantir a convergência de alguma de suas fases. Como ocorre no mecanismo *Peercoin*, que não utiliza comitês, mas implementa validações emitidas posteriormente pelos nós para garantir a persistência dos blocos já confirmados (KING; NADAL, 2012). A abordagem probabilística proposta oferece ainda parâmetros para que o novo mecanismo seja configurado, permitindo que o tempo para que um bloco seja confirmado e a segurança do consenso possam ser configurados por tais parâmetros.

Neste sentido, o texto da dissertação está organizado da seguinte forma: o Capítulo 2 descreve os principais conceitos relacionados à estrutura de uma blockchain típica. No Capítulo 3 são apresentados os principais mecanismos de consenso para blockchain públicas. Em seguida, no Capítulo 4 o novo mecanismo de consenso é apresentado, bem como uma discussão sobre suas principais características e formas de funcionamento. Os resultados obtidos por meio de experimentos práticos com o novo mecanismo são apresentados no Capítulo 6, mostrando como ele se comporta de acordo com mudanças de seus principais parâmetros. Já o Capítulo 7 compara o protocolo desenvolvido com os mecanismos estudados, apontando suas principais vantagens e desvantagens. Por fim, o Capítulo 8 discute os principais resultados deste trabalho e aponta algumas possíveis direções para trabalhos futuros.

2 Fundamentos de blockchains e mecanismos de consenso

2.1 Conceitos gerais de uma blockchain

Blockchain é uma tecnologia emergente que oferece suporte para que um livro razão seja replicado de forma consistente entre todos os participantes que compõem a rede. Este livro é preenchido com transações que são agrupadas em blocos, formando uma sequência sempre crescente (cadeia). Esta tecnologia é inovadora, pois cria um ecossistema confiável em um ambiente distribuído, sem exigir a presença de uma terceira parte confiável.

Segundo Greve et al. (GREVE et al., 2018), a blockchain implementa uma máquina de estados replicada que garante a manutenção de um estado global compartilhado, por meio da ação conjunta de pares que estão globalmente distribuídos na rede. Além disso, os passos que são tomados por estes pares para alterar o estado global do sistema são determinados pelo protocolo de consenso, que deve ser tolerante a ações de nós maliciosos. Abaixo são listadas as principais propriedades da blockchain, que colaboram para que ela seja de fato uma tecnologia desruptiva no contexto de sistemas distribuídos:

- descentralização: a blockchain não requer um controlador central. Assim, o estado atual da cadeia é acordado de maneira descentralizada, através de um mecanismo de consenso;
- transparência: todos os dados armazenados nos blocos são visíveis para todos os participantes da rede que os cria e mantém;
- imutabilidade: transações contidas em blocos confirmados são extremamente difíceis de serem modificadas sem que os participantes da rede percebam;
- privacidade: apesar de públicas, as transações não são facilmente rastreáveis, ou seja, dependendo do protocolo criptográfico utilizado, é difícil associar uma chave pública a um usuário;
- segurança: a autenticidade de uma transação é garantida a partir de uma assinatura digital gerada com uma chave privada. Estas assinaturas são facilmente verificadas com a chave pública associada e muito difíceis de serem forjadas.

A partir destas propriedades, muitos modelos de negócios podem ser criados, onde a confiabilidade dos mesmos depende das ações descentralizadas que são tomadas

pelos participantes da rede, na direção de um objetivo comum. Neste sentido, estão surgindo novas aplicações, a partir desta tecnologia, em diferentes áreas:

- internet das coisas (IoT): a blockchain pode ser utilizada em muitas redes de IoT, como por exemplo, em aplicações para redes veiculares (KANG et al., 2019);
- saúde: blockchains estão sendo adotadas por muitas empresas de saúde para aumentar a privacidade de pacientes, e para manter relatórios de procedimentos médicos imutáveis (BAXENDALE, 2016);
- militar: blockchain pode ser utilizada para manter bancos de dados seguros e descentralizados na área de inteligência militar (MCABEE; TUMMALA; MCEACHEN, 2019).

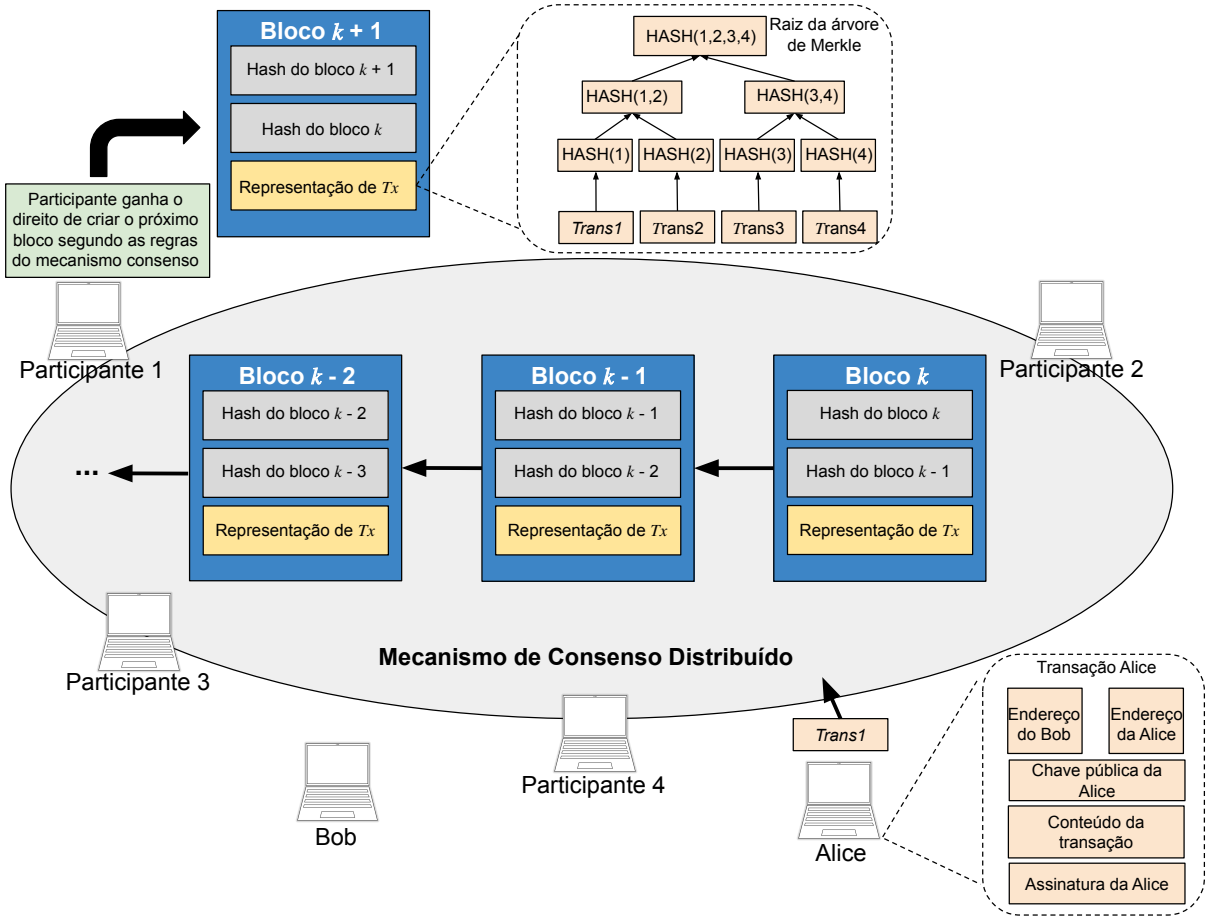

Figura 2.1 – Visão geral de uma blockchain. Adaptado do original disponível em (NGUYEN et al., 2019)

Baseada no trabalho de Nguyen et al. (NGUYEN et al., 2019), a Fig. 2.1 representa uma blockchain no nível de suas transações, onde é mostrado o ciclo de uma transação desde sua criação até o momento de sua entrada em um bloco. No cenário mostrado na Fig. 2.1, quando Alice quer enviar algum ativo a para Bob, ela cria uma

transação em seu dispositivo local com seu endereço exclusivo (Id_A), coloca o endereço de Bob (Id_B) como endereço de saída e assina esta transação com sua chave privada (Pr_A) para que todos possam verificar que a mesma foi criada por ela. Na transação, Alice também envia a sua chave pública Pu_A para que outros interessados sejam capazes de validar sua assinatura. É possível representar a criação da transação enviada por Alice a partir de uma notação formal, como a apresentada abaixo:

$$A \rightarrow B : \{trans(a, Id_A, Pu_A, Id_B)\}_{Pr_A}$$

Na notação, Alice envia para Bob ($A \rightarrow B$) um ativo a que está inicialmente associado ao seu endereço Id_A e a sua chave pública Pu_A. A posse do ativo é então transferida para o endereço de Bob Id_B, através de uma transação ($trans(a, Id_A, Pu_A, Id_B)$) que é assinada com a chave privada Pr_A de Alice ($\{trans(a, Id_A, Pu_A, Id_B)\}_{Pr_A}$).

Após a criação, Alice envia a transação via *broadcast* para a rede. Um ou mais participantes do mecanismo de consenso verificam se a transação é legítima e a inserem em um bloco junto às transações de outras pessoas. Se o bloco do participante atender os critérios de seleção estabelecidos pelo mecanismo de consenso, ele ganha o direito de divulgá-lo para os outros participantes. Caso a criação do bloco seja aprovada pelos outros participantes e ele seja o próximo bloco da cadeia (na Fig. 2.1 é esperado o bloco $k+1$), o mesmo é aceito e inserido na cadeia local de cada um dos participantes, como um novo bloco válido. Nota-se que o índice k associado ao bloco representa a sua posição ou altura na blockchain e seu valor é incrementado em uma unidade a cada novo bloco produzido. A partir deste momento, a transação realizada por Alice torna-se válida na blockchain, e Bob passa a ser o novo dono exclusivo do ativo a transferido, já que apenas ele conhece a chave Pr_B capaz de movimentá-lo em uma futura transação, ou seja, Pr_B é a única chave privada que está associada ao endereço Id_B e à chave pública Pu_B.

Todas as transações escolhidas pelo nó no momento da criação do bloco são inseridas no *payload* do bloco criado e uma árvore de *Merkle* é definida (Fig. 2.1), onde os hashes de cada transação formam os nós do primeiro nível desta árvore (folhas) (MERKLE, 1990). A construção da árvore segue para o segundo nível, onde cada valor deste nível será produzido por pares de hashes do nível anterior. No último nível da árvore apenas um hash é obtido, ou seja, a raiz da árvore capaz de representar todas as transações presentes no bloco. Esta raiz de Merkle é inserida no cabeçalho do bloco e, deste modo, é possível que qualquer participante possa verificar se uma determinada transação está no bloco a partir do hash da mesma e do caminho de Merkle que leva esta transação até a raiz da árvore. É desta forma que as diferentes partes que compõem o ecossistema de uma blockchain podem consultar se uma determinada transação está em um determinado bloco.

Qualquer bloco tem entre seus campos de cabeçalho o valor do hash do conteúdo completo do bloco anterior, i.e., tem o resultado da aplicação de uma função hash sobre todo o conteúdo do bloco anterior. Por isso, diz-se normalmente que um bloco sempre

aponta para o bloco anterior, já que seu conteúdo depende do anterior, que, por sua vez, depende do seu anterior e assim sucessivamente, criando uma cadeia de blocos dependentes uns dos outros.

2.2 Tipos de blockchains

Blockchains são classificadas em públicas, privadas e permissionadas. As blockchains públicas (*permissionless*), são assim chamadas porque a participação é aberta a qualquer um que deseja contribuir para a sua evolução, ou seja, as participações não são controladas por uma entidade centralizada ou um grupo específico. Deste modo, qualquer participante pode realizar transações, criar novos blocos ou participar do processo de consenso neste tipo de blockchain (ANTONOPOULOS, 2017). Os principais mecanismos de consenso para blockchains públicas serão apresentados em detalhes no próximo capítulo. Os outros tipos de blockchain, que não serão foco deste trabalho, são fechados à participação externa (privadas) ou exigem permissão de alguém ou alguma entidade para que um nó participe de suas ações (permissionadas).

Hyperledger Entre as propostas para blockchains privadas, destaca-se o *Hyperledger* (HYPERLEDGER, 2020), que é um projeto de código aberto mantido pela Linux Foundation. Esse projeto implementa um *framework* para a criação de blockchains privadas (CACHIN, 2016), e seus objetivos principais são disseminar o conceito de blockchain dentro da comunidade *open source*, além de criar um sistema modular capaz de auxiliar no atendimento da demanda de novos modelos de negócio. *Hyperledger* é subdivido em diferentes projetos, onde cada um utiliza um mecanismo de consenso diferente.

Neste sentido, o Hyperledger Fabric utiliza o consenso Kafka (FOUNDATION, 2017), que busca ordenar as transações a partir da eleição de um líder. Este consenso é baseado em um esquema de votação e não é tolerante à falhas Bizantinas (Anexo D), o que limita bastante seu escopo de atuação. Já o projeto Hyperledger Indy utiliza um mecanismo tolerante a falhas, chamado RBFT (Redundant Byzantine Fault Tolerance) (AUBLIN; MOKHTAR; QUEMA, 2013), que pode operar na presença de nós com comportamento Bizantino e também utiliza esquemas de votação. No projeto Iroha, utiliza-se o consenso Sumeragi (STRUCKHOFF, 2016), o qual é baseado na reputação de servidores que podem ser distribuídos entre diferentes *clusters*. O consenso proposto por tal projeto é tolerante a falhas Bizantinas e ocorre em intervalos de poucos segundos. Por fim, o projeto Hyperledger Sawtooth, utiliza o mecanismo de consenso PoET (*Proof of Elapsed Time*) (CORPORATION, 2013), que é uma estratégia associada a sorteios garantidos pelo *hardware* seguro SGX (*Software Guard Extensions*) produzido pela Intel.

2.3 Princípios básicos de segurança nas blockchains

A segurança das blockchains é fornecida por uma série de mecanismos e protocolos que, quando combinados, garantem as propriedades listadas na seção 2.1. Vale ressaltar, que as blockchains públicas operam em uma infraestrutura em que não existe relação de confiança entre os diferentes participantes (*peers*) que compõem a rede. Assim, nesta seção são destacados os principais agentes que garantem a segurança de uma blockchain.

2.3.1 Criptografia assimétrica

Nas blockchains públicas não existe uma entidade central ou federação capaz de autenticar os participantes. Logo um nó é identificado apenas por sua chave pública, que é associada a uma chave privada, por meio de um algoritmo de criptografia assimétrica (ou de chave pública). Desta forma, qualquer agente ativo que participe do ecossistema de uma blockchain, seja realizando transações ou colaborando com a produção de blocos, deve antes de qualquer ação, ser capaz de produzir seu par de chaves pública/privada, por meio do algoritmo de criptografia assimétrica adotado pela blockchain em questão.

No Bitcoin e em outras criptomoedas, a criptografia assimétrica é utilizada para associar um determinado recurso financeiro (moedas) a uma chave pública, onde apenas o dono da chave privada correspondente é capaz de movimentar este recurso através de uma nova transação na blockchain (TSCHORSCH; SCHEUERMANN, 2016). Essa nova transação deve ser assinada com a chave privada em questão, o que comprova para o restante da rede que de fato o dono do recurso quis fazer a transação e que ela é válida. É comum o uso de esquemas baseados em *Elliptic Curve Digital Signature Algorithm* (ECDSA) (NIST, 2009), parametrizado pela curva secp256k1 (Certicom Research, 2010), que é capaz de oferecer um nível de segurança de 128 bits com uma chave pública de 256 bits.

2.3.2 Resumos criptográficos

As funções hash são também chamadas de funções de espalhamento ou resumos criptográficos e são excelentes alternativas para a produção de saídas de tamanho fixo a partir de entradas de tamanho variável. Uma importante característica das funções hash é a sua capacidade em garantir a integridade de dados, pois uma mudança em qualquer bit ou bits na entrada resulta, com alta probabilidade, em uma mudança no código de hash (William Stallings, 2013). Nas blockchains são utilizadas funções hash criptográficas que, além da propriedade acima, são capazes de garantir (William Stallings, 2013):

- **resistência à pré-imagem:** para qualquer saída y, é computacionalmente muito difícil encontrar x, de modo que $H(x) = y$;

- **resistência à segunda pré-imagem (colisão fraca):** para qualquer entrada x, é computacionalmente muito difícil encontrar $z \neq x$ com $H(x) = H(z)$;
- **resistência à colisão forte:** é computacionalmente muito difícil encontrar qualquer par (x, z), de modo que $H(x) = H(z)$.

Nas blockchains, a resistência à segunda pré-imagem e colisão forte são propriedades fundamentais, já que garante que blocos com o mesmo *hash* não sejam facilmente produzidos, o que evita problemas de segurança ligados a tentativas de substituição de blocos (e transações) já validados e inseridos na cadeia. Como nas blockchains são utilizadas boas funções hashes criptográficas (SHA 2, por exemplo), pode-se assumir que cada bloco terá um *hash* exclusivo na blockchain.

O *hash* do bloco anterior é um dos parâmetros de entrada para o cálculo do *hash* do próximo bloco e isto faz com que seja improvável que modificações no conteúdo do bloco anterior possam ocorrer sem que o *hash* de seu sucessor na blockchain precise ser modificado. A amarração entre os *hashes* de cada bloco é o que permite que a blockchain seja comumente conhecida como corrente de blocos, onde cada bloco da corrente é fortemente conectado ao anterior, o que garante que o conteúdo de um bloco aceito não possa ser modificado sem que se modifique também todos os seus blocos sucessores.

2.4 Mecanismos de consenso

Bashir (BASHIR, 2018) define um mecanismo de consenso como um conjunto finito de passos que são dados pela maioria dos nós para que eles concordem sobre um valor ou estado do sistema. Assim, um mecanismo de consenso é avaliado pela sua capacidade de produzir altas taxas de transações confirmadas por segundo, ao mesmo tempo em que pode garantir segurança aos blocos confirmados através de regras que colaboram para que esses blocos não sejam cancelados ou removidos da blockchain. Em outras palavras, o consenso tem papel fundamental na segurança dos sistemas distribuídos, pois é ele que define os passos que os nós participantes devem seguir para alterar o estado global da blockchain, em um ambiente sucetível a falhas Bizantinas (Anexo D). Por este motivo, eles são conhecidos na literatura como consensos Bizantinos.

Neste tipo de consenso, cada processo p_i pode propor um valor v_i, e todos os processos corretos (ou honestos) devem decidir por um único valor v_i entre todos os propostos, dentro de um intervalo de tempo definido. Segundo Greve et al. (GREVE et al., 2018) os consensos Bizantinos devem atender um conjunto de três propriedades:

- Acordo: dois ou mais processos corretos decidem pelo mesmo valor;

- Terminação: todo processo correto decide por algum valor dentro de um intervalo de tempo pré-definido;
- Validade: se um processo correto decidiu por um valor v_i, então v_i foi proposto por algum processo.

No contexto das blockchains, os processos são os nós representados por suas chaves públicas, enquanto que os valores são os blocos produzidos por estes nós. Gilbert et al. (GILBERT; LYNCH, 2012) provou um teorema conhecido como CAP (*Consistency, Availabilty e Partition*), que define que os consensos Bizantinos não podem oferecer as três propriedades definidas abaixo ao mesmo tempo:

- Consistência: todas as requisições ao sistema obtêm os dados mais recentes;
- Disponibilidade: todas as requisições ao sistema obtêm uma resposta;
- Tolerância a partições: o sistema pode operar normalmente, mesmo que partições momentâneas possam ocorrer devido a falhas.

Além do resultado deste teorema, existe a incapacidade dos mecanismos alcançarem o consenso determinístico em uma rede totalmente assíncrona, como mostrado por Fischer et al. (FISCHER; LYNCH; PATERSON, 1985). Assim, os consensos Bizantinos escolhem duas propriedades do teorema CAP para satisfazer em detrimento da terceira e definem critérios para contornar a impossibilidade de obter o consenso determinístico em redes assíncronas. Neste sentido, a categoria de mecanismos que busca resolver o consenso de maneira determinística define algum grau de sincronismo na rede, para privilegiar o acordo em detrimento da evolução da blockchain, o que oferece alta consistência ao sistema, mas pode comprometer a evolução da blockchain (menor disponibilidade), quando as falhas Bizantinas ocorrem.

Uma outra forma de contornar o problema mostrado por Fischer et al. (FISCHER; LYNCH; PATERSON, 1985) é baseada no enfraquecimento do acordo, reduzindo assim a consistência momentânea do sistema frente às falhas Bizantinas, mas privilegiando a evolução da blockchain (disponibilidade), mesmo quando ainda não exista a garantia do consenso. Uma forma encontrada por esta categoria de mecanismos para enfraquecer o problema consiste em criar condições para que o acordo possa convergir probabilisticamente ao longo de algumas rodadas de execução.

2.4.1 Mecanismos de consenso probabilísticos e determinísticos

As condições expostas acima são suficientes para que os consensos possam ser classificados em dois grupos: probabilísticos ou determinísticos. Os consensos determinísticos são conhecidos por implementar mecanismos para resolver o problema teórico

dos gerenerais Bizantinos proposto por Lamport et al. (LAMPORT; SHOSTAK; PEASE, 1982), a partir da premissa de que algum grau de sincronismo possa ser considerado, e também pela formação de comitês de validação distribuídos, que podem ser implementados de diferentes maneiras. Os mecanismos podem ser do tipo atrasado, quando o consenso a respeito de um bloco ocorre após a chegada de alguns de seus sucessores, ou imediato, que ocorre no momento em que um novo bloco é proposto.

Por outro lado, a partir do enfraquecimento das propriedades do acordo, Ben-Or (BEN-OR, 1983) propôs um novo modelo capaz de garantir o consenso probabilístico, com base na utilização de números aleatórios, garantindo que ele ocorra mesmo quando o adversário tem informações completas a respeito do sistema. Neste tipo de consenso, o acordo não ocorre no momento em que um novo bloco é recebido. Ao invés disso, os participantes necessitam conhecer a evolução da blockchain para então concordarem sobre quais blocos criados no passado devem ser confirmados. Assim, o consenso ocorre baseado no aumento gradativo da confiança que os nós possuem em um bloco passado, à medida que novos sucessores do mesmo na blockchain são recebidos.

Como nesta categoria o consenso é atingido probabilisticamente e de maneira atrasada, podem ocorrer bifurcações (*forks*) na blockchain que levam o mecanismo a momentos de instabilidade, como mostrado na Fig. 2.2. De acordo com a figura, após o bloco B_4, a blockchain se dividiu em duas, já que existem dois sucessores possíveis para o bloco B_4 segundo as políticas definidas pelo mecanismo de consenso em questão. Por este motivo, os mecanismos de consenso probabilísticos devem definir critérios, que sejam capazes de conduzir os nós na direção de uma das duas cadeias, além de oferecer condições para que eles eliminem a cadeia não escolhida.

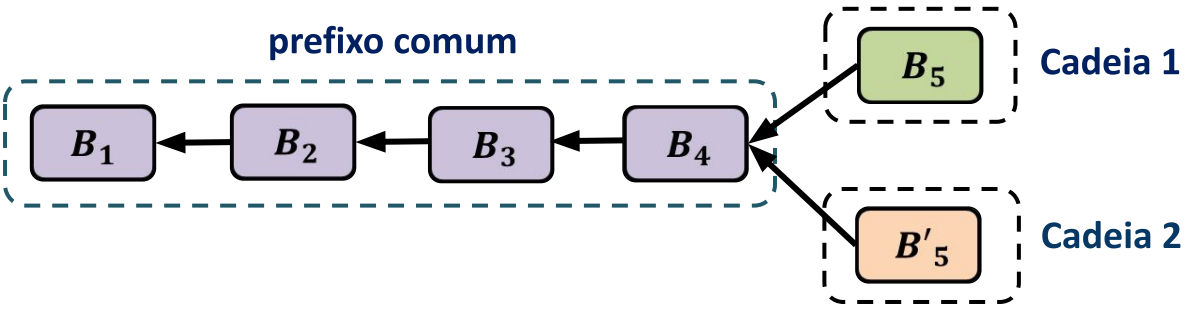

Figura 2.2 – A convergência atrasada e probabilística do consenso pode produzir *forks* na blockchain

Segundo Deirmentzoglou et al. (DEIRMENTZOGLOU; PAPAKYRIAKOPOULOS; PATSAKIS, 2019), estes critérios são definidos de acordo com propriedades das próprias cadeias que estão envolvidas no *fork*, o que é chamado de decisão baseada em cadeia (*Chain-based Decision*). O critério mais comumente utilizado pelos mecanismos consiste em seguir a cadeia que se tornar a mais longa em relação ao número de blocos.

Como a produção de sucessivos blocos nas duas cadeias, mostradas na Fig. 2.2, não é sustentada por muitas rodadas devido aos critérios de produção definidos por esses mecanismos, é esperado que após algumas rodadas uma das duas cadeias seja de fato a mais longa e, portanto, a que deve ser seguida por todos os nós honestos.

2.4.2 Consenso baseado em prova

Nas blockchains públicas, não existe uma forma de autenticar um participante por meio de alguma identidade controlada por ele. De fato, um nó pode criar sem custos e de forma anônima quantos pares de chaves assimétricas (identidades) desejar. Cada ação realizada por este nó passa a ser reconhecida por toda a rede, a partir apenas do vínculo que existe entre esta ação e uma das chaves públicas controladas pelo nó, garantindo ao sistema a propriedade de pseudonimização do dono da identidade (ANTONOPOULOS, 2017). Estas propriedades tornam inviável a possibilidade de limitar ou mensurar as ações de um participante, a partir do número de identidades que o mesmo controla na rede.

Assim, os mecanismos de consenso para blockchains públicas empregam o conceito de prova, onde algum recurso que possa ser associado direta ou indiretamente à identidade do nó passa a limitar a abrangência de suas ações durante o consenso (SONNINO; DANEZIS, 2019). O processo de associar o recurso de um participante com seu poder de decisão na rede é importante, pois evita que ele possa produzir e controlar novas identidades sem custo associado, buscando com isso aumentar seu poder de decisão no consenso ou sua probabilidade de produzir novos blocos. Esta produção irrestrita de identidades é a chave para um ataque comum em mecanismos distribuídos, conhecido como *Sybil Attack* (JYOTHI; JANAKIRAM, 2011). A prova deste processo ocorre quando todos os outros nós da rede são capazes de verificar facilmente que de fato o recurso utilizado durante a ação estava, de alguma forma, associado à identidade do nó que a executou.

Nos mecanismos baseados em *PoW*, este recurso é o poder computacional que cada participante dispõe para calcular valores de funções pseudoaleatórias (funções *hash*) (SONNINO; DANEZIS, 2019). De maneira indireta, o recurso computacional investido pelo nó para produzir um bloco está associado a uma ou várias identidades controladas por ele, já que eventuais recompensas derivadas da mineração são emitidas pelo próprio criador do bloco, através de uma transação que informa como saída um endereço de destino (identidade) controlado por ele. Assim, quanto maiores são os recursos disponíveis, maior é a probabilidade de o participante gerar o próximo bloco antes dos outros. Porém, o nó não obtém nenhuma vantagem ao criar novas identidades, já que o poder computacional disponível determina a chance do nó produzir o próximo bloco, independentemente de como é a sua distribuição entre as identidades controladas pelo nó.

Outro importante grupo de mecanismos são os que associam os recursos do nó à quantidade limitada de *stake* que este possui. Estes mecanismos são chamados de

Proof-of-Stake (PoS ou Prova de Posse) e sua primeira implementação prática foi idealizada por Sunny King e Scott Nadal (KING, 2013) e implementada como parte da criptomoeda *Peercoin*. Nele, cada participante utiliza a prova de posse de uma quantidade de recursos como uma forma de mensurar e limitar sua influência nas decisões tomadas durante o consenso no que tange a definição de qual será o próximo bloco da cadeia entre todos os que foram criados, ou para propor o próximo bloco. A prova de posse pode ser, por exemplo, a quantidade de moedas que o participante controla por meio de um endereço público na rede (DEIRMENTZOGLOU; PAPAKYRIAKOPOULOS; PATSAKIS, 2019). Algumas características dos mecanismos de consenso baseado em PoS listadas no trabalho de Maehara et al. (MAEHARA et al., 2018), são apresentadas abaixo:

- este mecanismo diminuiu o gasto de energia em comparação com o PoW, já que não incentiva e não depende de poder computacional elevado;
- é mais difícil garantir que apenas uma cadeia evolua ao longo do tempo como ocorre com o PoW;
- pode existir 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 maximizar seus ganhos individuais que podem ocorrer em qualquer cadeia que venha a vencer;
- o PoS permite aumentar uma eventual concentração de riquezas, já que quem possui mais moedas tem uma maior chance de ser escolhido para criar novos blocos e ganhar com as taxas atreladas às transações.

Além destas duas abordagens apresentadas, existem outros mecanismos de consenso propostos na literatura que deram origem a outros tipos de provas, tais como Proof-of-Activity (PoA) (BENTOV et al., 2014), Proof of Elapsed Time (PoeT) (CHEN et al., 2017), Proof-of-Retrievability (PoR) (MILLER et al., 2014), entre outros. Essas e algumas outras abordagens são apresentadas no próximo capítulo deste trabalho.

2.4.3 Incentivo à participação no consenso

Outro aspecto importante é a política de incentivo adotada pelo mecanismo de consenso. Uma boa política de incentivo é capaz de motivar os nós a seguirem as regras definidas pelo consenso, já que em geral as recompensas recebidas são mais vantajosas que uma possível tentativa de ataque ao mecanismo (ANTONOPOULOS, 2017). A utilização de incentivos para direcionar as ações de participantes desconhecidos e distribuídos em favor da evolução do consenso é mais um dos grandes avanços propostos pela blockchain do Bitcoin, por meio do seu consenso baseado em *Proof-of-Work* (NAKAMOTO, 2009).

Apesar de seu pionerismo, atualmente sabe-se que a política de incentivo adotada, por exemplo, pelo PoW do Bitcoin, não é adequada para atingir o objetivo de motivar os participantes com menos recursos financeiros, já que eles na maioria das vezes não são recompensados financeiramente. Isso porque a atividade ligada à mineração é a única forma existente de angariar recompensas na rede, onde apenas o nó capaz de produzir um bloco é remunerado, ou seja, na maioria dos casos, apenas aquele que dispõe de maiores recursos computacionais (NAKAMOTO, 2009).

Neste sentido, no próximo capítulo são apresentados mecanismos de consensos com políticas mais inovadoras em relação à capacidade de distribuir o incentivo entre os participantes, seguindo um critério bem estabelecido. Essas políticas que promovem a distribuição de recompensas são capazes de contribuir para que o consenso tenha maior descentralização por meio da entrada de mais participantes.

A descentralização é importante para que o consenso obtido seja seguro, ou seja, para que ele possa ser alcançado pela grande maioria dos nós honestos e que seja robusto contra tentativas de reversões, isto é, cancelamentos de blocos já inseridos. A descentralização é uma das características desejáveis em um mecanismo de consenso distribuído, fato que será mostrado com mais detalhes no Capítulo 4.

2.5 Conclusões

Este capítulo apresentou a blockchain como um livro razão distribuído, onde as ações conjuntas e descentralizadas dos diferentes nós que compõem a rede colaboram para que esse sistema possa mudar seu estado. Trata-se de uma tecnologia inovadora, pois cria um sistema seguro para que transações entre duas partes ocorra sem que exista uma terceira parte confiável, como um banco, cartório ou governo.

O mecanismo de consenso é parte fundamental da blockchain, já que ele define o conjunto de regras que os nós devem aplicar na sua visão replicada da blockchain, para que o estado global compartilhado seja mantido, enquanto o sistema evolui a partir de um acordo sobre cada novo bloco. Este mecanismo em geral opera em infraestruturas Bizantinas, onde nós podem falhar e a presença de nós desonestos podem dificultar as ações que levam ao consenso.

Além disso, os mecanismos de consenso para blockchains públicas devem implementar técnicas baseadas em algum tipo de prova como forma de controlar a influência de um participante nas ações executadas durante o consenso. É possível afirmar que este controle orientado à prova contorna a ausência de uma infraestrutura para autenticação dos participantes, o que colabora para que este tipo de blockchain tenha uma estratégia eficiente no controle das *Sybil* identidades.

Além do mecanismo de consenso, outros protocolos e premissas são combinados para que a blockchain seja segura. Dentre eles destaca-se a criptográfia assimétrica, os resumos criptográficos (hashes) e os incentivos à participação no consenso, que devem buscar a descentralização da rede, a partir de políticas que promovam a distribuição das recompensas angariadas. No próximo capítulo serão apresentados os mecanismos de consenso mais usados em blockchains públicas, destacando as principais características que nortearam o desenvolvimento deste trabalho.

3 Revisão do estado da arte em consenso distribuído

Neste capítulo são apresentados alguns dos principais mecanismos de consenso para blockchains públicas existentes na literatura. A partir do conceito de consenso baseado em prova mostrado no capítulo anterior, os mecanismos apresentados neste capítulo são classificados tomando como base a forma como provam suas ações durante o consenso.

Entre os consensos probabilísticos, é apresentado um breve resumo sobre o Proof-of-Work, apontando suas principais limitações. Além disso, o *Fast Probabilistic Consensus* (FPC) é mostrado como uma alternativa probabilística capaz de atingir o consenso sobre uma transação em um intervalo de tempo menor. Após uma explicação mais geral, é feita uma comparação mais aprofundada entre as estratégias que se baseiam no Proof-of-Stake como forma de controlar as identidades dos nós, já que os estudos apontaram vantagens desta categoria frente a outras aqui apresentadas.

3.1 Proof-of-Work

A primeira blockchain foi desenvolvida a partir do mecanismo de consenso *Proof-of-Work* (PoW) (NAKAMOTO, 2009). Segundo Nguyen et al. (NGUYEN et al., 2019), os nós encontram o consenso em uma blockchain baseada em PoW por meio de um processo de procura, onde cada nó avalia *nonces* (numbers used once) repetidamente com o objetivo de verificar se são capazes de atender um desafio estabelecido. O *nonce* e o *hash* do bloco anterior (entre outras informações fixas) são utilizados como entradas de uma função *hash* de 256 bits e a saída desta função deve ser menor do que um valor estabelecido (desafio). O nó que primeiro encontrar o *nonce* que gere um hash abaixo do limite ganha o direito de divulgar um novo bloco na rede para ser indexado à blockchain, como ilustrado na Fig. 3.1.

No PoW, cada nó compete com os outros para ser o primeiro a encontrar um *nonce* capaz de atender o desafio. Neste sentido, este processo depende dos recursos computacionais de cada nó, medido em função da taxa de *hashes* calculados por segundo. Segundo Nguyen et al. (NGUYEN et al., 2019), a probabilidade de um nó i encontrar o *nonce* primeiro e, com isso, ser selecionado é dada por

$$\frac{c_i}{\sum_{j=1}^{N} c_j} \tag{3.1}$$

onde c_i é a taxa de *hashes* por segundo do nó i e N é o número total de nós. Toda

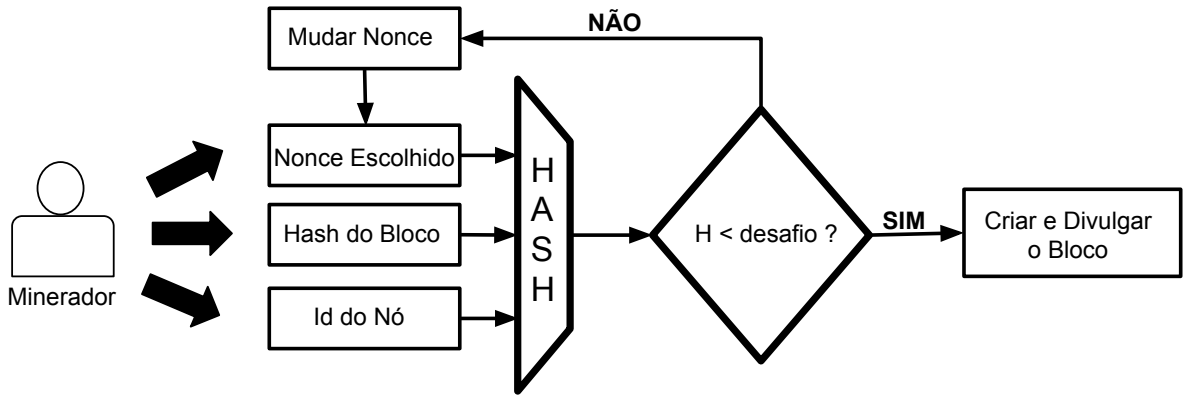

Figura 3.1 – No PoW os *nonces* são trocados até que um deles atenda o desafio estabelecido

esta computação leva a um grande desperdício de energia elétrica, já que todos os nós trabalham no mais alto desempenho computacional possível e apenas o trabalho de um deles é aproveitado ao final do ciclo de produção.

Desta forma, os custos energéticos envolvidos no processo definem os locais em que as grandes empresas ligadas à mineração são instaladas, o que colabora para que grande parte do poder de mineração esteja centralizado em poucos locais que oferecem energia elétrica de baixo custo. Por este motivo, a China lidera o número de mineradores do Bitcoin, enquanto aproxidamente 2/3 de sua matriz energética ainda é derivada de fontes não renováveis, como o carvão (ARATANI, 2021). Assim, como o único parâmetro de escolha dos mineradores é o custo energético e não existem formas para regularizar ou rastrear as fontes energéticas utilizadas, pode-se dizer que o processo de mineração contribui com o aumento das emissões de dióxido de carbono.

É comum que os participantes com baixo poder computacional se associem em grandes *pools* de mineração, aumentando suas chances individuais de receber recompensas, já que estas são divididas entre os participantes do *pool* de maneira proporcional. Como cada *pool* é controlado por uma única organização, essas associações contribuem para que o poder de mineração da rede seja dominado por poucos participantes, aumentando a centralização das ações do consenso. Assim, os grandes *pools* de mineração controlam a produção de novos blocos no PoW. Como exemplo, os 5 maiores *pools* em 2020 controlavam mais de 70% da taxa de mineração do Bitcoin (BLOCKCHAIN.INFO, 2020). Como consequência, as blockchains baseadas em PoW têm problemas relacionados à centralização, pois em geral a influência agregada que poucos *pools* exercem permite que eles controlem o mecanismo de consenso.

Por fim, PoW tem problemas relacionados ao tempo de confirmação. Um bloco é considerado confirmado apenas quando uma quantidade suficiente de sucessores são produzidos. No Bitcoin, por exemplo, é recomendado aguardar a criação de pelo menos seis blocos sucessores antes de considerar um dado bloco como confirmado de forma irreversível

(TSCHORSCH; SCHEUERMANN, 2016). Isso é necessário para que a probabilidade da cadeia principal ser revertida (cancelada e trocada por outra), a partir de um ataque conhecido como Ataque dos 51%, seja reduzida. Desta forma, o mecanismo é considerado mais seguro; porém a confirmação de um bloco leva pelo menos o tempo necessário para a criação de seus sucessores. Este tema será discutido com mais detalhes no Capítulo 7.

3.1.1 Bitcoin-NG

Com o objetivo de melhorar o desempenho do PoW, foram desenvolvidas algumas alternativas. Dentre elas destaca-se a proposta de Eyal et al. (EYAL et al., 2016), chamada Bitcoin-NG. A ideia principal desta técnica, consiste em continuar a criação de blocos no intervalo de tempo entre a produção de dois blocos produzidos pelo mecanismo PoW tradicional. Os blocos produzidos neste intervalo são chamados de *Microblocks*, e estão associados ao último bloco produzido através do PoW tradicional. Neste sentido, o processo de mineração tradicional é utilizado apenas para produzir os chamados *key Blocks*, e a partir deste momento, todos os *Microblocks* criados são assinados pelo dono da chave privada que está associada à chave pública do último *key block* produzido pelo PoW, como mostrado na Fig. 3.2.

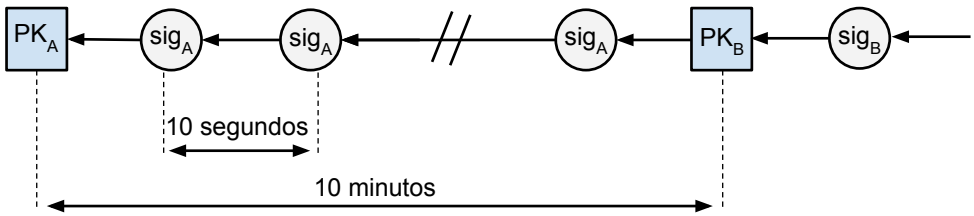

Figura 3.2 – Os *Microblocks* são assinados pela chave privada associada à chave pública do último *key Block* criado.

Quando um nó cria um novo *key block* através do PoW tradicional, ele se torna líder e ganha o direto de produzir *microblocks* seguindo uma taxa de produção máxima definida pelo mecanismo. O intervalo de produção deve ser muito menor que o tempo para se produzir um novo *Key Block*, para que o objetivo de melhorar o desempenho seja alcançado. Devido ao elevado valor da taxa de produção de *microblocks*, é esperado que sejam criados pequenos *forks* na transição entre dois *key blocks* sucessivos, como mostrado na Fig. 3.3.

Como o nó criador do novo *key block* estará produzindo os *microblocks* apenas em seu ramo do *fork*, é esperado que sua cadeia se torne a mais longa rapidamente, o que é capaz de resolver o *fork* produzido. Apesar de aumentar o número de blocos produzidos por rodada quando comparado ao PoW, o Bitcoin-NG é um mecanismo que ainda depende do PoW tradicional para produzir os *key blocks*, o que mantém neste mecanismo quase todos os problemas relacionados ao PoW.

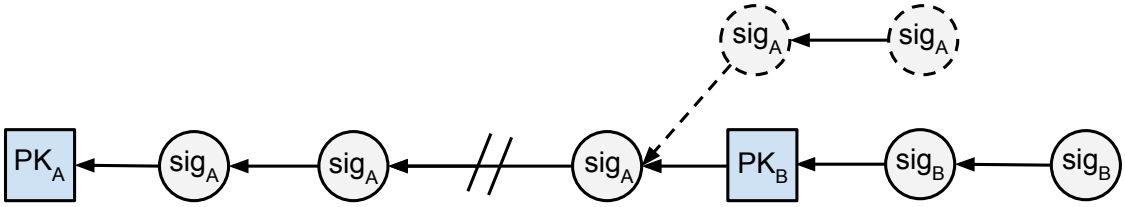

Figura 3.3 – *Forks* são esperados na transição entre dois *key blocks*.

3.1.2 EtHash

EtHash é uma proposta idealizada por Wood (WOOD, 2014) e tem como objetivo a redução da influência do poder de processamento na mineração de novos blocos, a partir da concepção de uma nova prova de trabalho.

Essa proposta busca criar um grande conjunto de dados aleatórios organizados em um DAG (Directed Acyclic Graph). Neste sentido, antes de iniciar a mineração, o participante constrói o DAG (inicialmente com 2 GB), a partir dos dados de um cache de menor tamanho. Os dados do DAG são atualizados a cada 30000 blocos (5 dias) e seu tamanho cresce linearmente a cada reconstrução. Os pontos principais do mecanismo são mostrados na Fig. 3.4. De acordo com a figura, no passo 1 o nó escolhe um nonce e, juntamente com os dados do cabeçalho do último bloco aceito, o primeiro *Mix* é calculado. O *Mix* é um valor pseudoaleatório e seu papel é endereçar uma parte do DAG (128 bytes) que deve ser lido da memória. No passo 2, o *Mix* 1 é calculado utilizando o *Mix* anterior e os dados do DAG lidos da memória. Esse novo *Mix* é utilizado como endereço pseudoaleatório para que uma nova parte do DAG seja lida da memória. O passo 3 mostra que o processo de definição do *Mix* e sua utilização para buscar valores em memória, é repetido até que 64 partes aleatórias do DAG sejam obtidas. Por fim, no passo 4, o último *Mix* (64) é utilizado para o cálculo de um novo hash que, por sua vez, é comparado com o desafio. Diz-se que o nó pode divulgar o próximo bloco, quando ele encontra um *nonce* menor que o desafio estabelecido ao final deste processo. Qualquer nó pode atestar a validade do *nonce* utilizado sem que seja necessário criar todo o DAG.

Esta nova abordagem, busca explorar a condição de que os acessos à memória não são otimizados nos hardwares especilizados (Application-specific integrated circuit - ASIC), contruídos para a mineração tradicional no PoW. Isso é possível porque os diferentes acessos ao DAG são pseudoaleatórios, contribuindo para uma grande quantidade de *cache misses*, o que aumenta a probabilidade de acessos à memória. Esta proposta foi adotada pela criptomoeda Ethereum em sua versão 1.0, mas não foi capaz de resolver complementamente o problema da baixa descentralização do PoW tradicional, já que ao longo do tempo novos ASICS foram propostos para otimizar esse novo tipo de prova de trabalho, obtendo ganhos significativos quando comparadas àquelas que utilizam hardwares tradicionais.

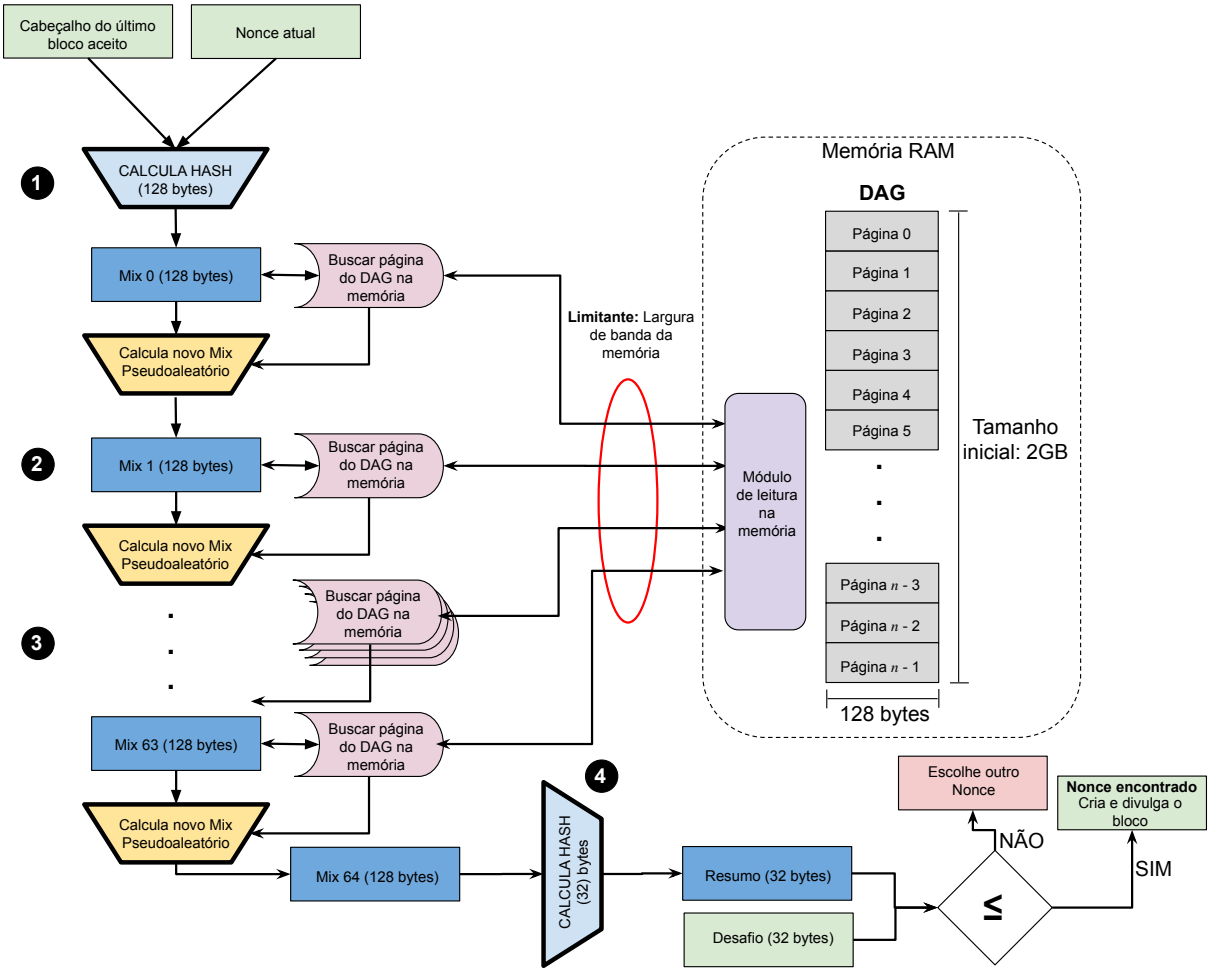

Figura 3.4 – EtHash busca limitar a influência dos ASICS a partir de uma intensa utilização da memória. Construído a partir do original proposto por Pradeep (PRADEEP, 2017).

3.2 Fast Probabilistic Consensus (FPC)

O FPC é um mecanismo de consenso probabilístico que permite baixa complexidade na comunicação entre os *peers*, por meio de um consenso a respeito de um único bit (POPOV; BUCHANAN, 2021). Neste mecanismo, um nó j obtém o consenso por meio de consultas realizadas a um conjunto de k nós, durante um intervalo de rodadas. É assumido que k deve ser relativamente grande ($k \geq 50$), mas muito menor que o número de nós presentes na rede. Essas consultas têm como objetivo identificar as diferentes opiniões destes nós a respeito do valor de um bit e, a partir dessas opiniões, o nó j é capaz de definir a sua própria opinião. O mecanismo depende de valores aleatórios calculados a partir de uma fonte confiável, além dos três parâmetros apresentados abaixo:

- $1/2 < a \leq b < 1$: limites para a primeira rodada do mecanismo;
- $\beta \in (0, 1/2)$: limites para as rodadas subsequentes;

- l : número de rodadas em que a opinião deve se manter estável para que a confirmação ocorra.

No início da primeira rodada cada nó decide o valor inicial de um bit, seguindo os seguintes passos:

- cada nó honesto j aleatoriamente consulta outros k nós e armazena o número de opiniões iguais a 1 ($\eta_1(j)$) recebidas;
- Depois disso, o mecanismo fornece aos nós um número aleatório X_1 uniformemente distribuído no intervalo $[a, b]$;
- Por fim, o nó j define sua opinião como 1 se $k^{-1}\eta_1(j) \geq X_1$, caso contrário ele adota o valor 0.

Nas próximas rodadas, o procedimento é quase o mesmo, onde a única mudança é em relação ao intervalo do número aleatório fornecido pelo mecanismo:

- Na rodada $m \geq 2$, cada nó honesto j aleatoriamente consulta outros k nós e armazena o número de opiniões iguais a 1 ($\eta_m(j)$);
- Depois disso, o mecanismo fornece aos nós um número aleatório X_m uniformemente distribuído no intervalo $[\beta, 1 - \beta]$;
- Por fim, o nó j, caso ainda não tenha obtido o consenso final, define sua opinião como 1 se $k^{-1}\eta_m(j) \geq X_m$, caso contrário ele adota o valor 0.

Caso um nó permaneça l rodadas consecutivas com a mesma opinião, esta é então definitiva e representa o consenso em relação ao bit.

Encontrar o consenso em relação a um bit é uma forma de validar transações de maneira individual, ou seja, não incorporadas a um bloco, como ocorre na maioria dos consensos para blockchain. Assim, por meio do estado do bit, é possível definir se uma transação é válida no que tange a utilização de valores não conflitantes com os de outras transações já realizadas. Como os valores aleatórios X_m são definidos por um fonte honesta, o consenso é seguro mesmo que um conjunto de participantes desonestos ataque o sistema a partir da redefinição de suas opiniões, com base nas respostas obtidas a cada rodada, com o objetivo de aumentar o tempo para que o consenso seja alcançado.

Segundo Popov et al. (POPOV; BUCHANAN, 2021), os adversários capazes de realizar este tipo de ataque são do tipo *Berserkey*, porém a troca de opinião dentro de uma rodada pode ser detectada pelos participantes honestos, após algumas iterações do mecanismo de consulta. A segurança é garantida mesmo na presença deste tipo de

adversário, já que eles não podem prever o valor aleatório X_m de cada rodada, mesmo conhecendo as respostas dos nós honestos. Entretanto, a utilização de uma fonte centralizada para criação de números aleatórios é contrária à ideia de descentralização das decisões de uma blockchain pública.

Outras alternativas para descentralizar a produção dos números aleatórios são possíveis, entre elas, segundo o autor, existe a possibilidade de criar um pequeno comitê de validação para definir o próximo número, porém não são explicadas quais são as premissas básicas para a formação do mesmo. Além disso, caso essa estratégia seja adotada, o mecanismo se tornará dependente de uma fase destinada à formação do comitê, que passará a anteceder todo o processo de consenso a cada rodada. Essa nova fase pode aumentar o tempo para que uma transação seja finalizada, causando impactos diretos no tempo de confirmação do mecanismo.

O FPC passou a ser considerado pelos autores como uma nova alternativa de consenso descentralizado para o sistema IOTA (POPOV et al., 2020). O IOTA é uma nova estrutura idealizada para ser utilizada em aplicações IoT, através da concepção de um sistema seguro para transmissão dos dados produzidos pelos dispositivos conectados à rede. A nova estrutura permite que as transações que contêm os dados transmitidos sejam confirmadas individualmente, ou seja, fora de uma estrutura organizada em cadeias de blocos. As transações são organizadas em um DAG (Directed Acyclic Graph) chamado Tangle (POPOV, 2018), que as mantém fortemente vínculadas umas as outras. Tangle é uma estrutura que oferece um maior desempenho em relação ao número de transações confirmadas por segundo quando comparado às blockchains tradicionais. Entretanto, em suas versões iniciais, o esquema de confirmação do IOTA não conta com mecanismos que favorecem a descentralização do consenso, fato que pode ser resolvido com a implementação do FPC, desde que sejam avaliados métodos descentralizados para a geração dos números pseudoaleatórios.

Por fim, o mecanismo FPC oferece um consenso probabilístico, já que as decisões são tomadas a partir da confiança que um nó tem em torno de uma opinião, após ela permanecer imutável por l rodadas seguidas. Como o consenso é alcançado algumas rodadas após a criação da transação, ele é do tipo atrasado.

3.3 Proof-of-Stake

O Proof-of-Stake (PoS) tem se destacado como uma alternativa ao alto consumo de energia elétrica dos mecanismos baseado em Proof-of-Work (PoW). No PoS, o nó que poderá criar um bloco é escolhido com base na posse de algum recurso limitado, normalmente o número de moedas em seu poder. 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. Neste sentido, a prova pode envolver o envio de moedas a si mesmo ou a algum outro elemento da rede, em caráter temporário ou permanente.

É comum que o tempo de confirmação e a segurança dos mecanismos baseados em PoS sejam associados à capacidade do mecanismo em garantir sincronismo na rede (NGUYEN et al., 2019). Deste modo, um novo requisito é que as mensagens sejam entregues dentro de um limite de tempo aceitável, garantindo que de fato sejam recebidas e consideradas válidas pelos participantes. Ainda assim, estes mecanismos podem ser tolerantes às falhas e à ausência parcial de sincronismo na rede. Os principais representantes desta categoria são os mecanismos que implementam comitês de validação. Cada mecanismo implementa o comitê de maneira diferente e sua principal função é confirmar os blocos produzidos a partir dos votos emitidos a cada rodada de validação. Em geral, o funcionamento destes comitês estão baseados no dilema dos generais Bizantinos, proposto por Lamport (LAMPORT; SHOSTAK; PEASE, 1982). Segundo Lamport, é possível atingir o consenso em um sistema distribuído se no mínimo 2/3 dos participantes forem honestos.

3.3.1 Ouroboros

O Ouroboros é um mecanismo de consenso baseado puramente em PoS (KIAYIAS et al., 2017). Ele é considerado síncrono e suas ações são organizadas em *slots* de tempo definidos em um período. No início de cada período, um comitê é definido através de um sorteio entre os participantes cuja probabilidade de sucesso é proporcional à quantidade de *stake* de cada participante. Após a formação do comitê, os participantes são encarregados de eleger aleatoriamente os novos líderes, que por sua vez, devem produzir os blocos. Além de eleger os participantes que produzem os blocos em cada *slot*, os membros do comitê são responsáveis por definir quais são os futuros membros do comitê com início no próximo período. As recompensas pela produção dos blocos são divididas entre os membros do comitê e os líderes selecionados pelo comitê, para incentivar a participação. O Ouroboros é seguro para uma rede parcialmente síncrona, onde os participantes desonestos controlam menos de 51% do *stake*.

3.3.2 Algorand

Algorand proposto por Gilad et al. (GILAD et al., 2017) é um mecanismo que implementa o conceito de comitê de validação de blocos por meio do uso de funções aleatórias verificáveis. Essas funções permitem que qualquer participante da rede possa verificar se foi sorteado para criar o próximo bloco em uma rodada, além de gerar uma prova do sorteio para que os outros nós possam atestar a veracidade do mesmo. Em cada rodada de produção, um ou mais blocos podem ser gerados e o mecanismo define qual será o vencedor a partir da formação de um comitê de validação dinâmico e aleatoriamente

definido, como ilustrado na Fig. 3.5. De acordo com a figura, a cada nova rodada os nós

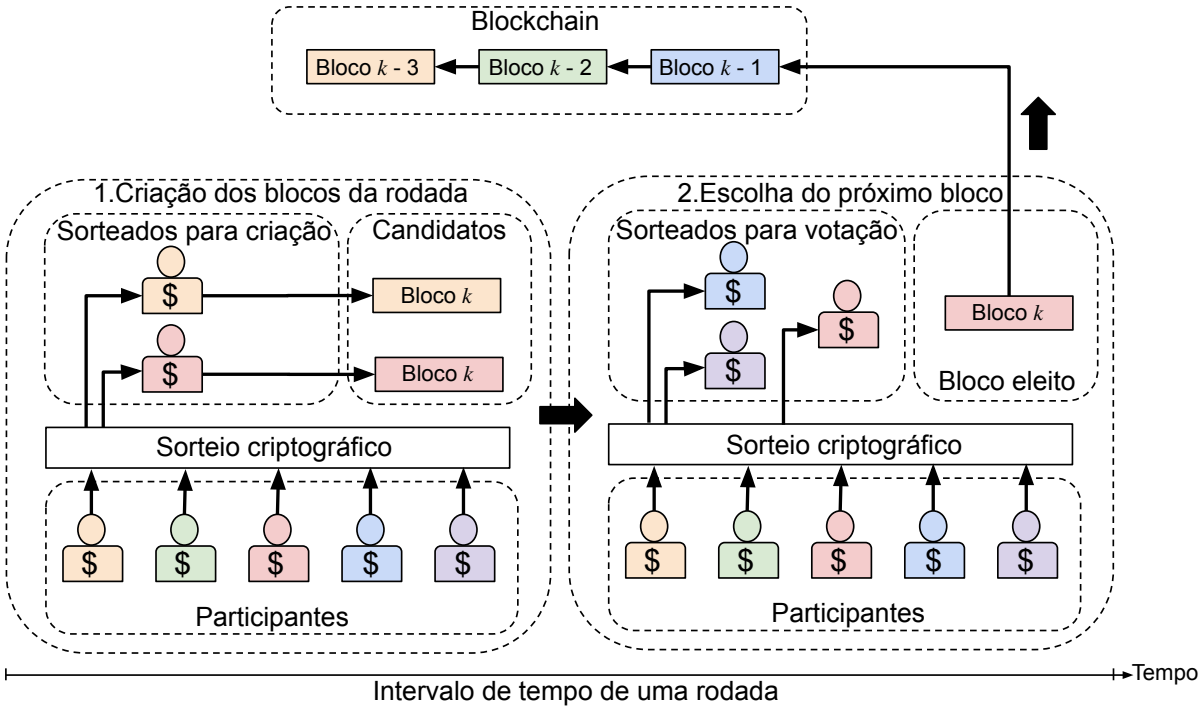

Figura 3.5 – Produção e validação de blocos no Algorand

participam de um sorteio criptográfico distribuído, com o objetivo de verificar se estão aptos a criar e divulgar um novo bloco na rede, ou seja, se seus parâmetros de entrada são capazes de serem bem sucedidos em um sorteio.

As propriedades do sorteio criptográfico garantem duas importantes características ao processo: (*i*) as identidades dos nós sorteados em uma rodada qualquer não são reveladas até que os respectivos blocos sejam produzidos e enviados na rede, o que evita que esses nós sejam de alguma forma subornados ou impedidos de produzir seus blocos; (*ii*) o subconjunto formado pelos nós sorteados é pequeno quando comparado ao conjunto total de participantes e, como o processo de sorteio é aleatório (apesar de ponderado pelo *stake* de cada nó), espera-se que esse subconjunto seja diferente a cada rodada.

As funções aleatórias verificáveis permitem que os nós que criaram blocos possam provar para o restante da rede que eles de fato foram sorteados através de parâmetros de entrada válidos. Assim, qualquer participante que recebe um bloco produzido é capaz de atestar se o mesmo foi criado a partir de um nó realmente sorteado.

Após a fase de criação dos blocos, o mecanismo de consenso segue para a fase 2, onde apenas um bloco entre todos os que foram propostos deve ser escolhido para ser inserido na blockchain. Assim, os nós participam novamente do sorteio criptográfico, porém agora com o objetivo de verificar se são considerados sorteados para compor o comitê de validação, o qual deve eleger o bloco a ser indexado à blockchain a partir do envio de votos para a rede *peer-to-peer*. Esse comitê é formado por meio de um processo semelhante ao

utilizado para criação dos blocos, o que fornece a ele as mesmas características explicadas anteriormente. Por fim, quando 2/3 dos votos do comitê são na direção de um mesmo bloco, ele é confirmado e inserido na blockchain.

As chances de um nó ser escolhido para produzir um bloco ou compor o comitê de validação são proporcionais à quantidade de *stake* que eles controlam através de suas identidades. O consenso obtido pelo Algorand é imediato, pois em cada ciclo de produção é formado um comitê com o objetivo de confirmar o bloco da rodada atual antes que outra possa ser iniciada. Como o mecanismo confirma de fato o bloco por meio dos votos de um comitê, este mecanismo é considerado determinístico e, portanto, o acordo é priorizado em relação à evolução da blockchain. O Algorand dispõe da melhor taxa de transações confirmadas por segundo entre os mecanismo avaliados, alcançada quando a rede está fortemente sincronizada, ou seja, quando as mensagens são recebidas pela grande maioria dos nós (95%) dentro do limite de tempo estabelecido.

A política de incentivo do Algorand distribui as recompensas entre os participantes que foram sorteados em qualquer uma das fases, de maneira proporcional à quantidade de *stake* de cada nó. Essa política tem a desvantagem de beneficiar os nós com maiores recursos financeiros, já que estes serão os nós sorteados com maior frequência a cada rodada.

3.3.3 Tendermint

O mecanismo Tendermint emprega um comitê de confirmação para validar os blocos propostos (KWON, 2014). Os nós interessados em participar do consenso na condição de validadores devem depositar uma quantidade de moedas a partir da criação de uma transação do tipo *unbound*. Essas transações indicam para os participantes qual é o *stake* de cada validador e, portanto, qualquer interessado pode obter o número total de *stake* disponível a partir da soma de todas as transações deste tipo que ainda sejam consideradas válidas. Diferentemente do Algorand, esse protocolo não utiliza mecanismos de sorteio criptográfico para formar um novo comitê de validação em cada nova rodada. Ao invés disso, esse comitê é formado a partir de um esquema de seleção baseado em *round robin* determinístico. Além disso, o criador do bloco também é escolhido a partir desse mesmo esquema, porém esta escolha é feita somente entre os participantes do comitê.

Os validadores participam do consenso através do envio de votos assinados, que são utilizados para confirmar os blocos. O mecanismo tem três tipos de votos: *prevote*, *precommit* e *commit*, como mostrado no diagrama da Fig. 3.6 Neste esquema, um bloco é confirmado quando recebe ao menos 2/3 dos votos do tipo *commit*. No início do processo o *proposer*, ou seja, o nó eleito para ser o criador do bloco, envia o bloco criado para a rede através de seus *peers*. Após receber o bloco proposto na rodada atual, os validadores realizam uma das duas ações possíveis: (*i*) se o validador já confirmou o bloco da rodada

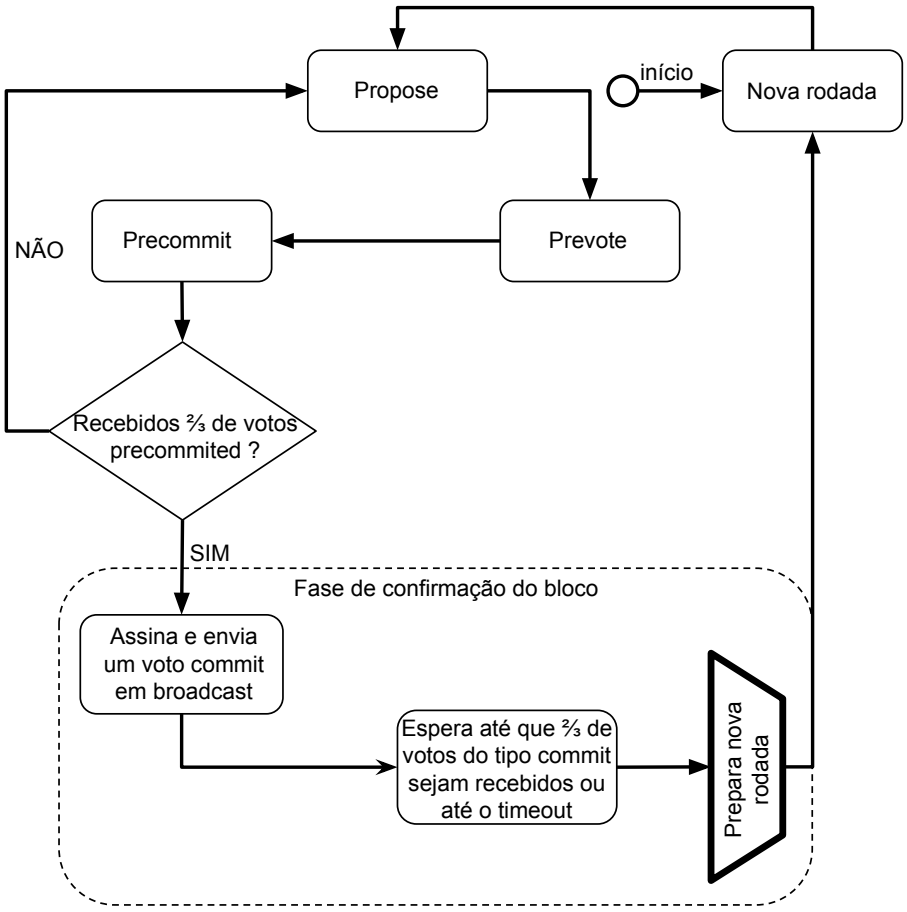

Figura 3.6 – Passos para o consenso no Tendermint

anterior, ele envia um *prevote* na direção deste novo bloco; (*ii*) caso exista um bloco não confirmado, que o validador já tenha votado na rodada passada, o *prevote* é enviado na direção do bloco anterior.

Na fase seguinte, caso o validador receba 2/3 dos votos na direção de um determinado bloco, ele cria e assina um voto do tipo *precommit* na direção deste bloco. Ele também víncula o seu endereço a este bloco, e destrava qualquer vínculo que tenha realizado em rodadas passadas com qualquer outro bloco, já que um validador pode estar vinculado a no máximo um bloco. Se ao final desta fase o nó receber 2/3 de votos do tipo *precommit* na direção do bloco em que está vinculado, ele parte para a fase de confirmação do bloco. Caso contrário, ele retorna para a fase em que um novo *proposer* será escolhido.

Na fase de confirmação, o validador realiza duas ações. Primeiramente ele aguarda o recebimento do bloco que recebeu 2/3 dos votos do tipo *precommit* na fase anterior, caso ainda não tenha o bloco. Uma vez recebido pelo validador, o bloco em questão é assinado e enviado via broadcast. A segunda ação consiste em aguardar até que o bloco receba uma quantidade de assinaturas que corresponda a 2/3 dos validadores. Após estes passos, o bloco é finalizado e uma nova rodada é iniciada. Uma diferença do Tendermint quando comparado com outros mecanismos de consenso baseados em comitês

de validação, está no fato que os validadores atestam o bloco que deve ser confirmado, a partir de assinaturas que permanecem no cabeçalho do próprio bloco. É desta forma, que os outros validadores comprovam se aquele determinado bloco já recebeu ao menos 2/3 dos votos necessários para ser confirmado. O Tendermint implementa um sistema de punições destinado aos nós que descumpram as regras estabelecidas, o que garante a destruição das moedas depositadas pelos validadores envolvidos nas atividades ilícitas.

O Algorand pode ser considerado mais robusto na escolha de quem vai propor o próximo bloco, quando comparado com o Tendermint. Isso ocorre, porque qualquer participante da rede pode ser escolhido e não somente os membros do comitê. O Tendermint exige que a rede tenha grandes restrições de comunicação entre os nós, pois é necessário que as mensagens com os respectivos votos, sejam recebidas dentro de um intervalo de tempo conhecido. Esse mecanismo garante que apenas um bloco é criado dentro de uma rodada o que otimiza consideravalmente o tráfego de rede. Assim, como ocorre no Algorand, o mecanismo é livre de *forks*. Como o mecanismo de seleção por meio do *round robin* não é bem explicado no artigo, não é possível realizar uma comparação mais aprofundada deste esquema com o sorteio criptográfico utilizado no Algorand.

3.3.4 Casper

O Casper é o mecanismo de consenso baseado em PoS que está em desenvolvimento para ser utilizado em uma nova versão da plataforma *Ethereum*. De acordo com Buterin et al. (BUTERIN; GRIFFITH, 2017), este mecanismo trabalha com o conceito de validadores, que possuem como objetivo principal a finalização de *checkpoints* por meio de seus votos. Os *checkpoints* são blocos cuja posição na blockchain é um múltiplo de uma quantidade pré-estabelecida (100 blocos na versão original do Casper) e marca o início de um novo período, ou seja, de um novo ciclo de produção de blocos.

Nas primeiras versões do Casper, a escolha sobre qual validador deveria produzir o próximo bloco era baseada no PoW, neste sentido, aqueles nós com maiores recursos computacionais eram selecionados com maior probabilidade. Deste modo, o mecanismo Casper era utilizado apenas em um segundo momento, para finalizar o bloco proposto. Nas abordagens mais recentes, o Casper é utilizado para propor e finalizar blocos utilizando apenas o PoS, onde os blocos são produzidos a cada rodada, por um conjunto de validadores escolhidos de maneira pseudoaleatória, em um esquema baseado em *round-robin* com probabilidade de seleção proporcional à quantidade de *stake* depositado. Os *checkpoints* são finalizados por meio da obtenção de 2/3 dos votos emitidos pelos validadores, garantindo que todos os blocos antecessores a ele sejam confirmados, como mostrado na Fig. 3.7.

Os círculos em verde indicam checkpoints finalizados e os cinzas, que estão acima do mais alto círculo verde, são checkpoints que ainda poderão ser finalizados pelos nós. Os checkpoints que estão fora do ramo compreendido entre o mais baixo e o mais alto

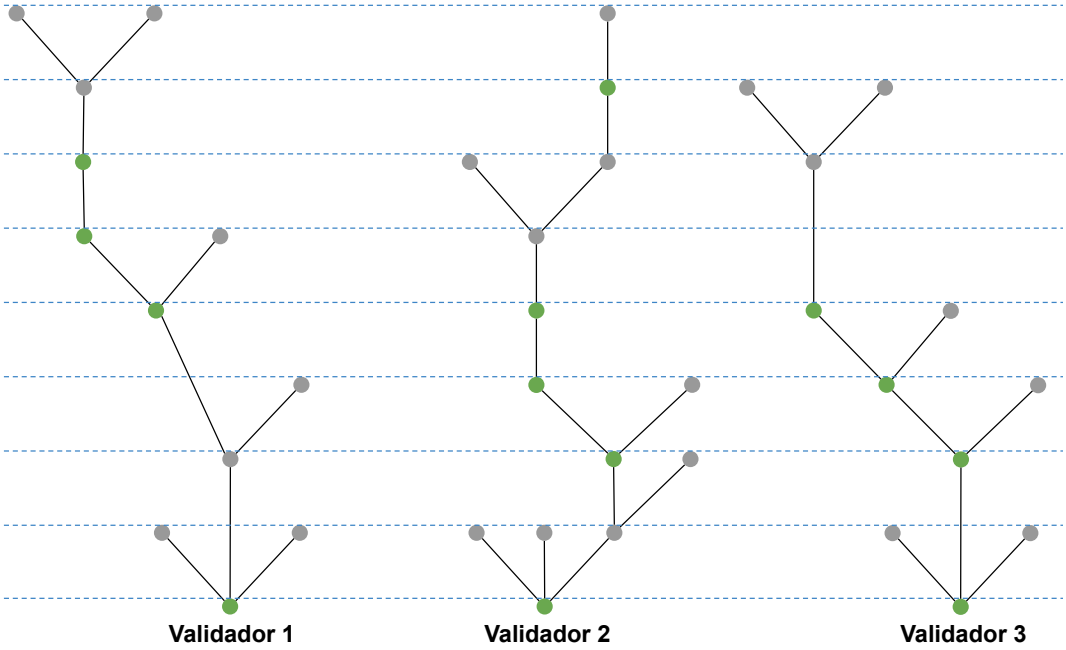

Figura 3.7 – Validadores com visões diferentes mas finalizando os mesmos checkpoints.

checkpoints finalizados (verdes) não são e nem serão confirmados, já que pertencem a outros *forks* e, portanto, a visões conflitantes da blockchain. Neste sentido, o Casper garante através de um conjunto de regras que dois checkpoints conflitantes não são finalizados simultaneamente pelos validadores, o que viabiliza o consenso, mesmo a partir de visões diferentes da blockchain. Como os blocos são confirmados apenas quando estão entre dois checkpoints finalizados, o consenso do mecanismo é atrasado e, como a finalização do checkpoint é realiza por meio dos votos do comitê, o consenso é determinístico.

Um validador pode ser qualquer participante que realize um depósito de uma quantidade de recursos (moedas) e concorde com a perda destas moedas caso as regras do mecanismo de consenso não sejam seguidas por ele. Este depósito é a quantidade de *stake* do participante (validador) e a contribuição do seu voto é proporcional a este *stake*. Um voto é uma mensagem criada e assinada pelo validador, que é enviada em *broadcast* para todos os outros validadores da rede. A tabela 3.1 mostra os parâmetros que definem um voto no mecanismo Casper. O *hash* de um *checkpoint* é utilizado para identificá-lo de maneira exclusiva, enquanto os parâmetros $h(s)$ e $h(t)$ correspondem as alturas dos *checkpoints*, ou seja, são os índices que estes blocos ocupam na blockchain. Estes valores são utilizados para determinar se um voto emitido pelo validador segue as regras definidas pelo mecanismo. Por fim, o validador assina a mensagem com sua chave privada, garantindo a autenticidade da mesma para os outros validadores da rede.

Diz-se que quando mais de 2/3 dos validadores votam em um link, cuja origem é s e o destino é t, o link $s \rightarrow t$ passa a ser chamado de *supermajority link*. Quando t vem

Tabela 3.1 – Atributos que definem um voto

Notação	Descrição
s	hash do *checkpoint* de origem
t	hash do *checkpoint* de destino
$h(s)$	altura do *checkpoint* s medido a partir do bloco genesis
$h(t)$	altura do *checkpoint* t medido a partir do bloco genesis
S	Assinatura da mensagem realizada com a chave privada do validador

imediatamente após s na blockchain, t é um filho direto de s. Abaixo são definidos os conceitos de *checkpoint* justificado e finalizado:

- **checkpoint justificado**: Um *checkpoint* t é justificado se ele é o bloco gênesis ou se o link $s \rightarrow t$ é *supermajority* e s é justificado.
- **checkpoint finalizado**: um *checkpoint* s é finalizado se é justificado e existe um *supermajority link* $s \rightarrow t$, onde t é filho direto de s.

Nem todo *checkpoint* que tenha sido justificado se tornará finalizado, já que dois *checkpoints* justificados (mas em ramos diferentes) serão conflitantes. Entretanto, não existe uma forma de finalizar dois *checkpoints* conflitantes, se ao menos 2/3 dos validadores seguirem as regras apresentadas abaixo:

- um validador não pode publicar dois votos $[s_1, t_1, h(s_1), h(t_1)]$ e $[s_2, t_2, h(s_2), h(t_2)]$, onde
  - **Regra 1:** $h(t_1) = h(t_2)$ (um validador não deve publicar dois votos na direção de dois *checkpoints* diferentes e que tenham a mesma altura);
  - **Regra 2:** $h(s_1) < h(s_2) < h(t_2) < h(t_1)$ (um validador não deve publicar um voto em um link $s_2 \rightarrow t_2$ cuja altura está contida em um voto publicado anteriormente).

Caso um determinado validador não siga as regras definidas acima, ele é punido com a destruição dos valores depositados como garantia. Por outro lado, todos os validadores que contribuem para a finalização de um *checkpoint* são recompensados de maneira proporcional ao *stake* depositado. Essa é uma vantagem do mecanismo em relação a outros mecanismos baseado em PoS já que, por meio da divisão da recompensa entre todos os participantes e pelo fato da entrada no comitê ser aberta a qualquer nó, a participação no consenso torna-se interessante mesmo para nós com poucos recursos financeiros. Assim, é esperado que o casper atinja melhores níveis de descentralização quando comparado com outros mecanismos de consenso.

3.4 Outros tipos de provas (PoX)

3.4.1 Proof of Elapsed Time (PoeT)

O PoeT é um consenso probabilístico que cria o conceito de espera aleatória, para definir qual nó pode criar o próximo bloco. Neste sentido, cada nó da rede recebe um número aleatório no início de cada rodada, onde esse número corresponde ao tempo que ele deve esperar até que seu bloco possa ser transmitido na rede (CHEN et al., 2017). O bloco que esperar o menor tempo, será o primeiro a chegar nos nós e, portanto, será aceito. A Eq. 3.2 mostra como o tempo de espera é produzido em cada nó, onde o parâmetro *minimum_wait* é fixo e definido para todo o sistema. Já o cálculo do parâmetro *local_average_wait* depende da estimativa do número de nós ativos no sistema. O objetivo desse parâmetro é ajustar o tempo de espera ao número de nós, ou seja, quanto maior o número de nós ativos maior será este valor. Por fim, o parâmetro $r \in [0, 1]$ é um número real aleatório e derivado de uma função hash que está associado a identidade do nó e a parâmetros estáveis da rodada. Como é utilizado um tipo de função hash cuja saídas podem ser consideradas pseudoaleatórias (SHA256, por exemplo), r é uniformemente distribuído no intervalo $[0, 1]$.

$$wait\_time = -local\_average\_wait \times \log(r) + minimum\_wait \tag{3.2}$$

Cada nó utiliza a máquina de estados finita mostrada na Fig. 3.8 para controlar o processo de alteração do parâmetro *wait_time*. Assim, no estado 0, o nó calcula um novo *wait_time* e segue para o estado 1. Em cada um dos estados $i \in \{1, 2, 3, .., 25\}$, após a geração do bloco, o nó pode retornar para o estado 0 com probabilidade p_i ou seguir para o próximo estado com probabilidade $1 - p_i$. Ao retornar para o estado 0 a partir do estado i, um novo *wait_time* é definido, e o próximo bloco produzido passa a considerá-lo como o novo tempo que deve ser aguardado até que o bloco possa ser divulgado na rede. Por outro lado, quando o processo de maneira aleatória define que o nó deve seguir para o próximo estado $(i + 1)$, o *wait_time* do estado anterior permanece válido e ainda define o tempo de espera do próximo bloco. Um mesmo *wait_time* pode ser utilizado no máximo 25 vezes, o que é equivalente a dizer que $\sum_{i=1}^{25} p_i = 1$.

A integridade do processo de produção do *wait_time* é obtida através do módulo de hardware chamado Software Guard Extensions (SGX) desenvolvido pela Intel (CORPORATION, 2013). Esta tecnologia é capaz de proteger um software contra modificações externas, buscando garantir que uma vez embarcado no módulo, as saídas produzidas pelo software não podem ser modificadas ou influenciadas por um agente externo. Além disso, o SGX auxilia na emissão de uma prova, utilizada para comprovar aos outros nós que de fato o nó aguardou o tempo necessário antes de enviar o bloco.

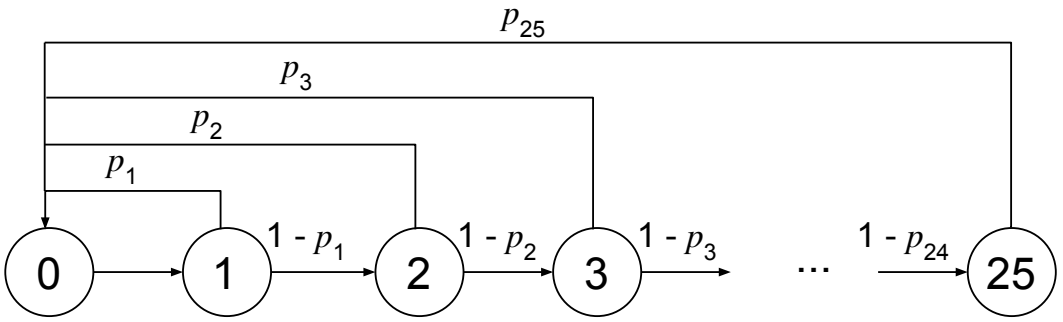

Figura 3.8 – Em cada estado o nó tem probabilidade p_i de produzir outro *wait_time*

Como os tempos de espera seguem uma distribuição de probabilidade, esse fato pode ser utilizado pelos nós para validarem se os tempos de espera são corretos, o que contribui para aumentar a segurança do sistema que passa a não depender unicamente da prova emitida pelo SGX.

A principal desvantagem deste método está associada ao fato de toda a segurança do mecanismo ser garantida por um hardware proprietário. Esta abordagem é contrária a ideia de descentralização das ações tomadas pelo mecanismo de consenso distribuído, já que as ações são garantidas a partir de uma relação de confiança existente entre os nós e o próprio hardware. Além disso, Chen et al. (CHEN et al., 2017) mostrou ser possível manipular a produção de blocos no PoeT, quando os adversários controlam uma fração na ordem de $O(\frac{\log\log(n)}{log(n)})$, onde n é o número de nós na rede. Esse resultado mostra que o ataque não é delimitado por uma fração constante, como ocorre, por exemplo, no PoW (51% do poder de mineração) ou nos mecanismo baseado em PoS que implementam comitês de validação (1/3 do *stake*).

3.4.2 Proof-of-Activity (PoA)

O mecanismo de consenso PoA é uma extensão do PoW do Bitcoin, onde um novo bloco é proposto através do PoW e, em um segundo momento, um grupo de validadores é aleatoriamente definido para inserir as transações neste bloco e validá-lo. A formação deste grupo é baseada no PoS, onde as chances de um determinado participante ser selecionado para fazer parte do grupo é dependente do seu *stake*. Segundo Bentov et al. (BENTOV et al., 2014) a motivação principal do PoA está na possibilidade de aumentar a segurança contra o ataque de 51%, a partir de um esquema em que o bloco é confirmado apenas se os N validadores assinarem o mesmo.

Neste sentido, em um primeiro momento todos os nós do sistema competem entre si para produzir um novo bloco, utilizando o PoW. Nenhuma transação é inserida neste bloco e apenas seu cabeçalho é enviado na rede. Após essa fase, um grupo com N validadores é formado, e seu objetivo é validar o bloco recém produzido, além de inserir as transações. Diz-se que o mecanismo tem um novo bloco válido quando os $N-1$ primeiros

validadores assinam o bloco recém criado e o último validador insere as transações no bloco e assina o cabeçalho com a sua chave privada. Todo esse processo é ilustrado na Fig. 3.9, para um grupo formado por 3 validadores.

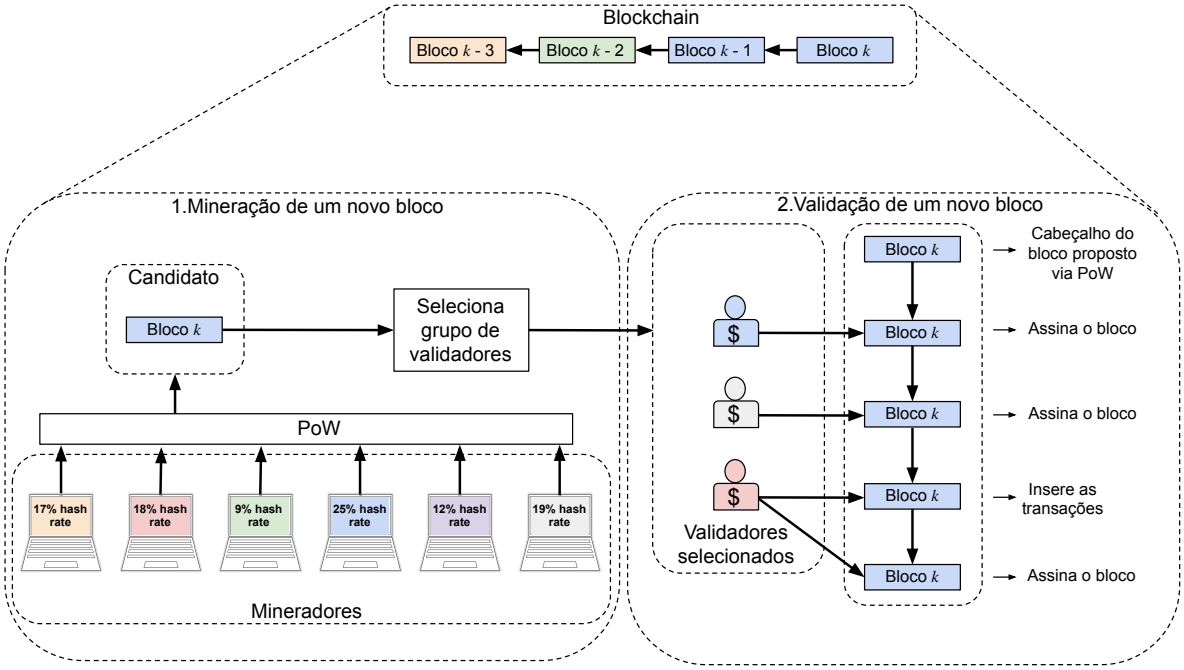

Figura 3.9 – Prova de atividade (PoA)

Quando algum dos N validadores está *offline*, o bloco é descartado e o processo é reiniciado, dando origem a criação de um novo bloco, e, consequentemente, ao sorteio de um outro grupo de validadores. O PoA reduz consideravelmente a incidência de *forks* durante o consenso, já que é necessário que cada bloco receba N assinaturas para que seja anexado à blockchain. No PoW tradicional, o impacto causado por um conluio de nós desonestos capaz de controlar mais de 51% do poder de mineração da rede, pode ocasionar reversões na cadeia principal. Já no PoA, é criada uma nova proteção contra esse tipo de ataque, pois mesmo que os nós desonestos consigam reverter a cadeia principal a partir da produção de uma cadeia mais longa, ainda é necessário que cada bloco criado pelo conluio obtenha N assinaturas, que devem ser produzidas por um grupo de validadores gerado aleatoriamente e que, portanto, pode conter nós honestos que não fazem parte do conluio.

3.4.3 Proof-of-Retrievability (PoR)

Miller et al. (MILLER et al., 2014) desenvolveram uma proposta que busca resolver o problema do alto gasto de energia elétrica do PoW, a partir de algumas mudanças nas características que definem o desafio a ser alcançado para produzir o próximo bloco da cadeia. Estas mudanças deram origem a um novo mecanismo de consenso, que faz parte de uma nova blockchain chamada Permacoin. Neste sentido, eles utilizaram um esquema

baseado em Proof-of-Retrievability (PoR), onde um participante seleciona uma pequena fração de um conjunto de dados conhecido por toda a rede para armazenar em sua própria infraestrutura local. A fração de dados escolhida para armazenamento, é associada a sua chave pública, utilizando para isso a saída de um função *hash* criptográfica de 256 bits (SHA256, por exemplo).

As provas de recuperabilidade (PoR) tradicionais permitem que a entidade que esteja armazenando o conteúdo forneça uma prova para que qualquer validador interessado possa confirmar o armazenamento. Entretanto, essas provas exigem que o validador envie um desafio para a entidade, o qual pode ser resolvido apenas se a mesma estiver de fato armazenando o conjunto de dados em questão. Essa interação entre pares não é razoável para a criação de uma nova blockchain, pois todos os nós são considerados validadores de um bloco produzido, o que constribui para o aumento do custo de verificação em um modelo interativo. Neste sentido, Miller et al. (MILLER et al., 2014) desenvolveram uma mudança nas abordagens tradicionais de PoR, para que o desafio gerado pelos validadores pudesse ser utilizado como forma de comprovação do armazenamento dos dados, mesmo sem estabelecer comunicação com o armazenador. Esta mudança foi fundamental para que o novo mecanismo PoR fosse aplicado a uma blockchain.

Desta forma, para propor um novo bloco, um nó escolhe aleatoriamente um subconjunto de dados e associa esta escolha à sua chave pública. Como a limitação está no espaço de armazenamento, é esperado que um único nó não seja capaz de armazenar todo o conjunto de dados, contribuindo para que o volume de dados total seja repartido de maneira descentralizada entre os participantes. Assim, aquele nó que demonstrar maior capacidade de armazenamento terá uma maior probabilidade de ganhar o direito de propor um novo bloco.

Não é claro se este novo mecanismo é capaz de resolver o problema do desperdício de energia elétrica, já que a mineração passa a ser alicerçada em um novo recurso computacional: espaço de armazenamento. Além disso, essa proposta pode privilegiar poucas empresas que dispõem de grandes datacenters de armazenamento, o que pode não favorer a descentralização das ações que levam ao consenso.

3.5 Comparação entre os mecanismos

Uma avaliação inicial dos mecanismos apresentados neste capítulo, mostra que algumas novas abordagens ainda utilizam o PoW como mecanismo de prova. É o caso do Bitcoin-NG, que apesar de melhorar o desempenho a partir de uma proposta que aumenta a taxa de produção de blocos, ainda depende de uma mineração tradicional para que os *key blocks* sejam produzidos. A prova de atividade (PoA), busca resolver problemas de segurança do PoW tradicional, empregando o uso de múltiplas assinaturas geradas por

um grupo de validadores que é formado utilizando o Proof-of-Stake (PoS), mas também depende do PoW para que os blocos sejam produzidos.

O PoR e o EtHash oferecem de fato uma nova prova, mas que ainda está amparada em recursos computacionais, o que pode manter os problemas ligados à desperdício de energia elétrica ou baixa capacidade de descentralização. Por outro lado, o PoeT mostra-se capaz de resolver a competição computacional imposta pelo PoW, já que a prova passa a ser baseada em um tempo de espera aleatório, que pode ser comprovado pelos outros participantes. Esse tempo é produzido com baixo custo computacional, a partir de um módulo de hardware seguro produzido pela Intel. Entretanto, este mecanismo tem como grande desvantagem a utilização de um hadware proprietário, que tem o papel de garantir a integridade dos tempos de espera que são emitidos pelo sistema. Isso traz à tona a necessidade de uma relação entre os nós da rede e uma terceira parte confiável, algo contrário às premissas básicas de uma blockchain.

Por fim, os mecanismos baseados em PoS são escolhidos para uma análise mais minuciosa, já que o controle de identidade é realizado por meio de uma prova totalmente nova, que utiliza a posse do *stake* ao invés dos recursos computacionais, sem que exista uma relação direta entre a segurança do mecanismo e uma terceira parte confiável. Para uma comparação mais direta destes mecanismos, a tabela 3.2 relaciona alguns parâmetros importantes para uma blockchain pública à forma com que eles são implementados nos mecanismos baseados em PoS. Os dados da tabela são estendidos para os mecanismos PoW e FPC, para que esta avaliação também tenha representantes probabilísticos.

De acordo com a tabela, os mecanismos baseados em PoS com comitê de validação apresentam características desejáveis para o consenso de blockchains públicas. Entre estas características, destacam-se as elevadas taxas de transações confirmadas por segundo, os menores tempos necessários (latência de confirmação) para que os blocos aceitos sejam confirmados na blockchain e as políticas de incentivos, que buscam favorecer a descentralização do consenso, através da distribuição das recompensas angariadas durante o processo.

Nesta linha, o mecanismo de consenso Algorand permite que qualquer nó da rede possa verificar de maneira descentralizada se ele é considerado sorteado para produzir um bloco na rodada. Esta verificação é realizada apenas por meio de parâmetros locais do próprio nó. Além disso, o Algorand permite que um novo comitê de validação seja formado a cada rodada, preservando a identidade de quem é sorteado até que seu voto seja divulgado na rede. Estas duas características aliadas a um eficiente protocolo de distribuição de mensagens, garantem ao Algorand a maior taxa de transações confirmadas por segundo e a menor latência de confirmação entre todos os mecanismos analisados, além de evitar que os nós do comitê sejam subornados.

Com relação ao incentivo à participação nas atividades do consenso, o me-

canismo Casper destaca-se por meio de uma política que permite a distribuição das recompensas entre todos os membros do comitê. No Algorand isso também ocorre, porém apenas entre os nós sorteados, o que acaba privilegiando aqueles que dispõem de maiores recursos financeiros. Portanto, o Casper tem um processo de distribuição de recompensas que favorece a descentralização da rede. A segurança de ambos os mecanismos é garantia se menos de 1/3 do stake da rede pertencer a nós desonestos, o que está relacionado com a condição limite para que o problema dos Generais Bizantinos tenha solução determinística (LAMPORT; SHOSTAK; PEASE, 1982).

Por outro lado, estes mecanismos procuram resolver o consenso por meio da convergência dos votos distribuídos por um comitê de confirmação em direção de um único bloco ou *checkpoint*. Estes comitês trazem novas complexidades para o consenso no que tange a sua implementação em uma rede com características assíncronas, como é caso de uma blockchain pública (FISCHER; LYNCH; PATERSON, 1985). Desta forma, caso não haja convergência dos votos em direção de um único valor, a blockchain pode não evoluir, impactando diretamente no tempo que os blocos levam até serem confirmados pelos mecanismos. Além disso, o consenso destes mecanismos é determinístico, ou seja, é necessário atingir mais de 2/3 dos votos em direção de um único valor para que o acordo entre os participantes ocorra.

Algumas aplicações podem requerer um tempo de confirmação menor, a partir, por exemplo, da possibilidade de se obter um consenso mais configurável, fato que não é possível em sistemas que buscam atender um único limiar, como é o caso destes mecanismos. Esse menor tempo de confirmação pode ser atingido por meio de novos esquemas baseados em confirmações probabilísticas, como é o caso do FPC. Entretanto, o FPC também depende de um comitê de validação para que os números pseudoaelatórios sejam produzidos de maneira descentralizada, o que pode vir a reduzir o seu desempenho final.

Tabela 3.2 – Comparação entre os mecanismos de consenso

Mecanismo	**Proof-of-Work (PoW)**	**Fast Probabilistic Consensus (FPC)**	**Algorand**	**Casper**	**Ouroboros**
Tolerância a adversários	até 1/2 do poder de mineração	menos de 1/3 da *mana* total	menos de 1/3 do *stake* total	menos de 1/3 do *stake* total	ate 1/2 dos participantes
Transações confirmadas por segundo	0,55	sem informação	95	90 (considerando que os *checkpoints* são sempre finalizados a cada 100 blocos)	16,7
Tempo até a confirmação do bloco	1 hora	sem informação	20 segundos	em média 1043 segundos (considerando períodos com 50 blocos).	2 minutos.
Estratégia de confirmação	Atrasada e probabilística	Atrasada e probabilística	Imediata e determinística	Atrasada e determinística	Atrasada e determinística
Incentivo para participação no consenso	(−) Recompensas são recebidas apenas pelo nó que minerar o bloco	sem informação	(±) Recompensas são divididas entre os nós sorteados para criar o bloco ou participar do comitê de validação.	(+) Recompensas são divididas entre todos os validadores.	(±) Recompensas são divididas entre os validadores sorteados.
Vantagens	simplicidade de operação; utilizado com sucesso em várias blockchains.	eficiente sistema de troca de mensagens; confirmação individual das transações.	escolha secreta de qual participante cria o próximo bloco; comitê formado de maneira secreta; livre de *forks*; baixo tempo de confirmação.	identifica e executa punições em nós desonestos; recompensas são distribuídas entre os participantes do comitê.	maior robustez quanto à tolerância a adversários.
Desvantagens	alto consumo de energia elétrica; ações centralizadas em poucos nós; tempo de confirmação elevado.	produção centralizada de números aleatórios.	grandes restrições nos atrasos de comunicação na rede; possibilidade de bloqueio na evolução da blockchain.	o depósito de garantia deixa uma quantidade de recursos indisponibilizada; comitê não muda a cada rodada.	comitê é fixo dentro de um longo período; tempo de confirmação elevado.

3.6 Conclusões

Neste capítulo foram apresentados os mecanismos de consenso que representam o estado da arte em blockchains públicas. A partir destes mecanismos, diferentes formas de prova foram mostradas, onde a prova de posse (Proof-of-Stake) tem se mostrado mais eficiente principalmente pela sua total desassociação com o PoW, o que reduz consideravelmente o gasto energético de sua operação. Além disso, mostrou-se que é possível aumentar a descentralização do consenso a partir de abordagens baseadas em PoS, utilizando políticas de incentivo distribuídas, como a implementada pelo mecanismo Casper, que é capaz de dividir as recompensas recebidas entre todos os validadores ativos.

Com relação ao desempenho, os mecanismos que implementam comitês de validação como forma de resolver o problema dos generais Bizantinos sucetíveis a falhas (BFT), alcançam as melhores taxas de transações confirmadas por segundo e as menores latências de confirmação. Porém, esses esquemas de validação são de complexa implementação e, em geral, exigem um grande sincronismo da rede para que uma rápida convergência do consenso ocorra, o que pode dificultar a escalabilidade destes mecanismos para uma rede com muitos nós. Na sequência deste trabalho, o novo mecanismo de consenso proposto será mostrado em detalhes, bem como suas principais diferenças em relação aos mecanismos baseados em comitês de validação.

4 Um novo mecanismo de consenso: Committeeless Proof-of-Stake (CPoS)

É sabido que os mecanismos de consenso para blockchains públicas buscam atender de maneira eficiente três objetivos (XIAO et al., 2020):

- Segurança: capacidade em manter as operações do consenso seguras mesmo em uma infraestrutura Bizantina;
- Descentralização: adoção de políticas que incentivam a participação de um grande número de nós, reduzindo com isso a influência individual de cada participante nas ações que levam ao consenso;
- Escalabilidade: capacidade de alcançar altas taxas de confirmações (consensos) por segundo.

Estes objetivos são as bases de um importante trilema na área de consenso distribuído, cujo principal resultado diz que estes objetivos não podem ser atendidos simultaneamente. Xiao et al. (XIAO et al., 2020), apresentou as propriedades fundamentais que contribuem para que cada um destes objetivos sejam individualmente atendidos no contexto das blockchains públicas e a relação dessas propriedades com os outros objetivos. Essas propriedades foram adaptadas para refletir o resultado da avaliação feita no capítulo anterior deste trabalho e os resultados são mostrados na Fig. 4.1

Na figura foram destacadas apenas duas características desejáveis de cada um dos três objetivos, sendo possível notar que a alta segurança do consenso é a característica que mantém uma relação conflitante com o maior número de outras propriedades desejáveis. A segurança do consenso é normalmente associada aos mecanismos BFT clássicos, que priorizam um acordo determinístico, ao mesmo tempo em que são tolerantes às falhas Bizantinas. Em ambientes descentralizados isso é alcançado à medida que se aumenta o sincronismo na rede, o que contribui para a redução do particionamento da blockchain em diferentes visões; entretanto, torna-se mais difícil descentralizar as ações do consenso em uma rede globalmente distribuída, quando as restrições de sincronismo entre os nós aumentam (XIAO et al., 2020). Por outro lado, quando o consenso é obtido a partir da participação de uma grande quantidade de nós, é esperado que sua segurança cresça, já que os principais modelos de ataques são baseados na formação de um conluio de nós desonestos que têm que controlar uma fração dos recursos da rede (DEIRMENTZOGLOU; PAPAKYRIAKOPOULOS; PATSAKIS, 2019). A descentralização capaz de distribuir os

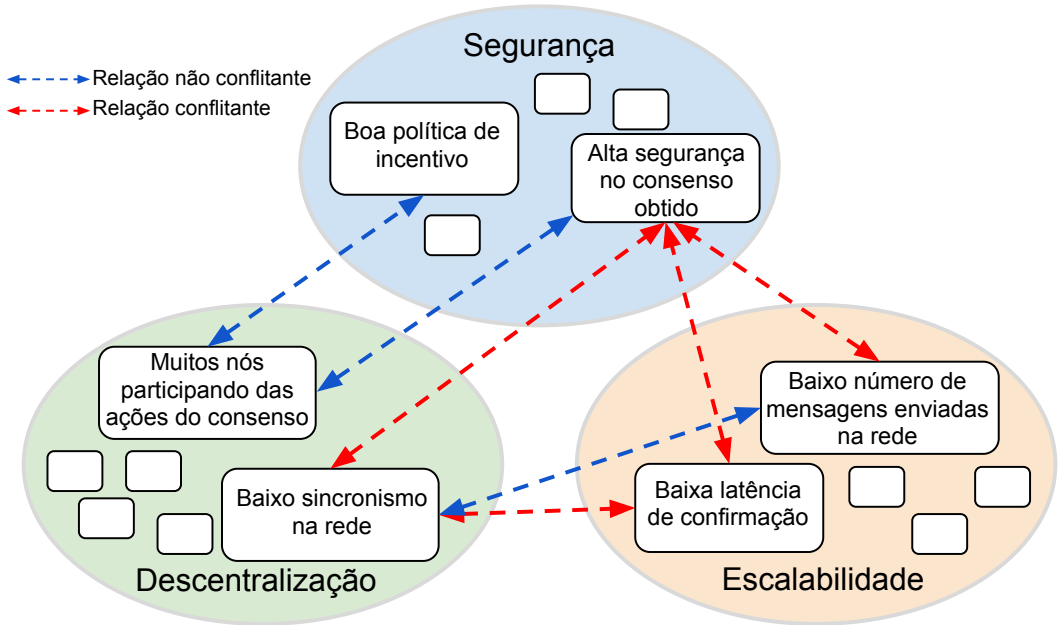

Figura 4.1 – As relações negativas existentes entre as propriedades desejáveis são entraves para o atendimento simultâneo dos três objetivos. Adaptado do original disponível em (XIAO et al., 2020)

recursos em um grande número de nós, colabora para que a abrangência deste tipo de ataque seja atenuada.

A alta segurança do consenso também tem relações conflitantes com as propriedades mais próximas da escalabilidade do sistema. A latência de confirmação diminui quando é exigido menos tempo para que o consenso sobre o próximo bloco seja alcançado, o que pode ser obtido a partir de medidas que buscam enfraquecer as propriedades do acordo. Além disso, no capítulo anterior mostrou-se que o consenso é atingido deterministicamente a partir da implementação de algoritmos BFT, que estão amparados em comitês de confirmação distribuído. Essa propriedade é contrária à ideia de se atingir a escalabilidade através da redução do número de mensagens enviadas na rede, já que os mecanismos baseados em comitês devem propagar os votos para toda a rede em cada rodada de confirmação.

Por esses motivos, torna-se salutar para um novo mecanismo de consenso, a definição de propriedades que permitam que o nível de segurança do acordo seja configurável, para que a blockchain possa se beneficiar de maneira mais ativa de outras propriedades. Vale ressaltar que o surgimento de outros modelos de negócios que não estão diretamente ligados a operações financeiras como, por exemplo, as aplicações voltadas a *IoT* (CHRISTIDIS; DEVETSIKIOTIS, 2016), também apelam para a criação de um consenso mais configurável, já que é possível que esses novos negócios tolerem uma diminuição controlada no nível de segurança, em favor, por exemplo, de uma diminuição da latência de confirmação dos blocos.

Neste sentido, este capítulo apresenta o Committeeless Proof-of-Stake (CPoS) como um novo mecanismo de consenso capaz de oferecer maior configuração ao processo de confirmação de blocos, quando comparado aos mecanismos baseados em PoS que implementam comitês de confirmação. A abordagem utilizada para alcançar esse objetivo consiste em obter o consenso por meio da implementação de estratégias probabilísticas, além de aproveitar algumas das características dos mecanismos descritos no capítulo anterior.

O CPoS propõe um novo mecanismo de confirmação de blocos que não requer o uso de comitês de confirmação, removendo o tempo e a complexidade adicional desses comitês, além de trazer parâmetros de controle que influenciam diretamente na latência de confirmação e na segurança do acordo estabelecido. Além disso, esse mecanismo é uma interessante alternativa ao PoW, já que oferece baixo consumo de energia elétrica, melhor desempenho e estratégias que promovem a descentralização. De maneira mais direta, as principais contribuições do CPoS são listadas abaixo:

- não requer comitê de validação para confirmação dos blocos;
- oferece confirmação probabilística com parâmetros ajustáveis;
- desempenho pode ser melhorado sem modificar o nível de segurança do consenso;
- não é amparado por provas de trabalho alicerçadas em grandes esforços computacionais, o que reduz os requisitos de energia.

4.1 Visão geral do CPoS

O mecanismo Committeeless Proof-of-Stake (CPoS) é um algoritmo de consenso baseado em sorteios que ocorrem em rodadas de tempo finito com duração T, onde a cada início de rodada um novo ciclo para produção de blocos se inicia. Os nós participantes do consenso devem tentar produzir o próximo bloco sempre dentro da rodada atual, a partir das suas referências de tempo locais.

Neste sentido, caso um nó não seja sorteado, ele deve aguardar o início da próxima rodada, a qual garante o direito de participação em outro sorteio para obter uma nova chance de produzir o próximo bloco. A necessidade de aguardar o início de uma próxima rodada tem a função de evitar que nós com alto poder computacional tenham vantagens em relação àqueles com poucos recursos, já que neste esquema todos os nós estão limitados pelo tempo de duração de cada rodada. A verificação se a rodada de criação de um bloco é igual à rodada esperada pode ser realizada por qualquer nó, por meio da Eq. 4.1. O quociente desta equação fornece o número de rodadas inteiras decorridas entre o tempo de criação do primeiro bloco produzido (gênesis) e o bloco proposto (corrente) que

deve ser verificado.

$$r_{exp} = \left\lfloor \frac{t_{cb} - T_0}{T} \right\rfloor \tag{4.1}$$

Os parâmetros usados na Eq. 4.1 são definidos a seguir:

- r_{exp} : rodada esperada calculada;
- t_{cb} : tempo (local) de chegada do bloco corrente (bloco sob verificação);
- T_0 : tempo de produção do bloco gênesis conhecido por todos os nós.

O fato de todo nó que recebe um bloco proposto utilizar seu próprio tempo local para definir a rodada esperada, evita que ele tenha que confiar na rodada de criação do bloco definida por outros nós. Assim, todo bloco deve ser produzido dentro da rodada esperada pelos nós da rede. No Anexo B é mostrado que é possível definir um intervalo de tolerância para o CPoS, onde um bloco pode ser aceito mesmo que seja produzido em uma rodada diferente daquela esperada pelo nó destinatário. Este intervalo é necessário para considerar as diferenças de relógios entre os nós e possíveis variações nos atrasos de transmissão da rede.

A Fig. 4.2 mostra os passos seguidos pelos nós do CPoS (círculos brancos numerados) desde a participação no sorteio a partir do início de uma nova rodada até a aceitação do bloco produzido por um nó sorteado. No passo 1, quando uma nova rodada é iniciada, os três nós participantes fornecem ao mecanismo de sorteio a rodada atual, calculada localmente a partir do tempo corrente e dos parâmetros definidos na Eq. 4.1 (T_0 e T), algum saldo de moedas (*stake*) associado a um endereço controlado por eles e o *hash* do primeiro bloco confirmado no período atual. No CPoS um período é um intervalo de tempo onde uma quantidade fixa de blocos é produzida e seus detalhes serão discutidos na seção 4.3. O saldo de um endereço é o *stake* utilizado pelo nó durante o sorteio em cada rodada. A chance de um nó ser sorteado para produzir o próximo bloco na rodada atual é diretamente proporcional à quantidade de *stake*, como será apresentado em detalhes na próxima seção. Já o *hash* do primeiro bloco do período atual é uma maneira de associar o processo de sorteio a um parâmetro desconhecido pelos nós antes do início do período.

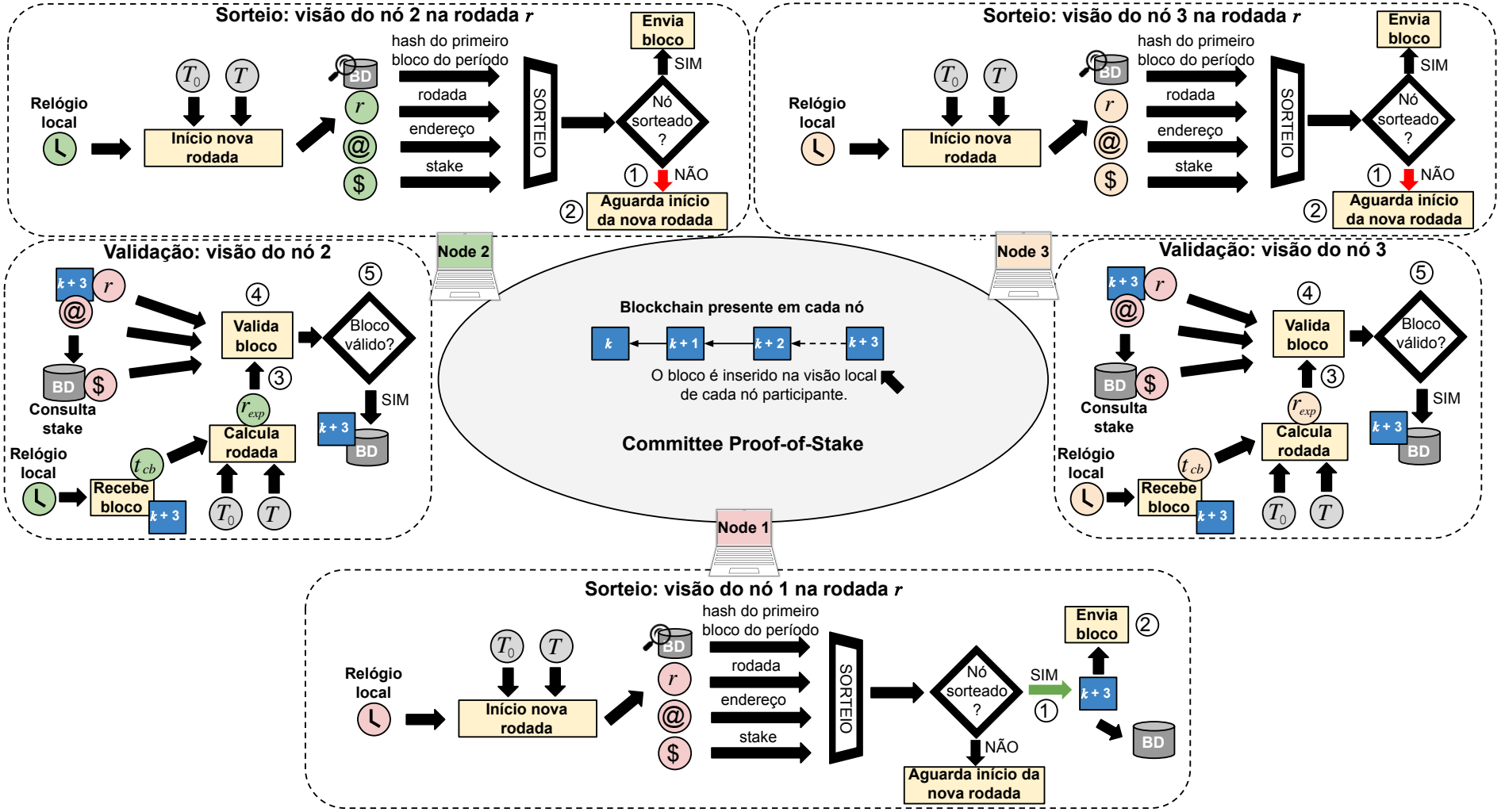

Figura 4.2 – Visão simplificada da produção e validação de um novo bloco no mecanismo CPoS

No passo 2, os nós 2 e 3 não são sorteados na rodada r e, portanto, eles devem aguardar o início da próxima rodada para que uma nova participação no sorteio possa ocorrer. Já o nó 1, foi sorteado na rodada r, ganhando com isso o direito de produzir e divulgar um novo bloco nesta rodada. Ao receber o bloco recém produzido, os nós 2 e 3 iniciam o processo de validação por meio do cálculo da rodada esperada (Eq. 4.1), utilizando o tempo de chegada do bloco em validação (t_{cb}) calculado com suas próprias referências de tempo e as constantes T_0 e T, como mostrado no passo 3. A rodada esperada, a rodada presente no cabeçalho do bloco, o endereço e o *stake* do nó criador são utilizados como entradas do mecanismo de validação de blocos, como mostrado no passo 4. Durante a validação, caso a rodada do bloco seja válida, os nós verificam se o bloco foi de fato produzido por um nó sorteado e, se válido, ele é inserido na blockchain local dos validadores como um bloco recém criado, porém ainda não confirmado.

O CPoS é um mecanismo de consenso probabilístico e de confirmação atrasada, ou seja, todo bloco recebido e aceito pelo nó participa de um processo de confirmação que ocorre em rodadas futuras. A confirmação probabilística do CPoS define condições para que os nós avaliem se seus blocos podem ser confirmados, a partir da análise de suas visões locais da blockchain. Como será mostrado em detalhes na seção 4.4, os nós decidem, aplicando os critérios definidos, quais blocos serão mantidos por eles na blockchain. Portanto, a visão local da blockchain em cada nó é formada pelos blocos aceitos em cada uma das rodadas de criação. Em outras palavras, a visão local é a cadeia de blocos resultante das decisões de aceitação tomadas a partir do conjunto de blocos recebidos a cada rodada.

Posteriormente, os blocos aceitos devem ser confirmados, como será mostrado em detalhes na seção 4.5. Para isso, os nós consideram os blocos recebidos que são sucessores diretos do último bloco de suas visões locais. Dessa forma, cada nó pode mensurar o número de nós sorteados que estão produzindo blocos na mesma cadeia. Essa medida permite que os nós sejam capazes de avaliar se suas visões estão sendo consideradas pela maioria dos nós da rede ou se estão associadas à visões secundárias, isto é, a cadeias que estão sendo consideradas por poucos nós e que, portanto, são incapazes de apresentar uma métrica que possa confirmar seus blocos.

4.2 Sorteio criptográfico

4.2.1 O algoritmo de sorteio

O protocolo de sorteio utilizado no CPoS é baseado no mecanismo de consenso do Algorand (GILAD et al., 2017). Desta forma, são utilizados os conceitos de sorteios aleatórios baseados em *hashes* criptográficos, onde, em cada rodada, novos nós são selecionados (sorteados) para produzir o próximo bloco da blockchain.

Neste sentido, em uma rodada r qualquer, o nó i calcula primeiramente o *hash* do nó ($U_i^{(r)}$). Como este valor é a saída de uma função *hash* (H), resistente a ataques de colisão e de primeira e segunda pré-imagens, ele pode ser considerado pseudoaleatório e seu valor representa o nó i durante a rodada r. O cálculo desse *hash* utiliza como entradas a rodada atual (r), o identificador exclusivo do nó (Id_i) e o *hash* H_{block} (Eq. 4.8) do primeiro bloco confirmado do período mais recente (aqui denotado por H_{fc}).

Este cálculo é apresentado na Eq. (4.2) por meio da saída de uma função *hash* de n bits (H), como a conhecida função SHA-256 ($n = 256$). Os três parâmetros utilizados como entrada são colocados no cabeçalho do bloco, caso o nó i seja sorteado para criá-lo na rodada r.

$$U_i^{(r)} = H(Id_i||r||H_{fc}) \tag{4.2}$$

No CPoS, um período é formado por um intervalo de rodadas, onde um número fixo de blocos são produzidos e confirmados. O conceito de período e suas implicações no cálculo do *stake* são apresentados em detalhes na seção 4.3. Como explicado a seguir, o valor H_{fc} não é conhecido pelo nó i antes de ele depositar seu *stake* do período atual, evitando assim que ele conheça, antes do sorteio, os momentos em que ele será de fato sorteado. Isso ocorre, porque para participar dos sorteios com o objetivo de produzir blocos em um período, o nó i deve realizar, durante o período anterior, alguma transação que movimente recursos financeiros (*stake*) para algum endereço de seu controle. Assim, como o valor H_{fc} será conhecido apenas após o início do próximo período corrente, um nó não é capaz de realizar depósitos apenas em períodos específicos, contribuindo para que a quantidade total de *stake* da rede não mude abruptamente entre períodos.

Para participar do sorteio é necessário que exista também uma associação entre o nó e o seu *stake* w_i. Para esta finalidade é definido um intervalo de números inteiros $1 \leq k \leq w_i$. Cada valor de k representa uma oportunidade do nó i ser sorteado dentro da rodada r, como será mostrado mais adiante. A soma W dos *stakes* de todos os nós da rede ($W = \sum w_i$) fornece o número total de sorteios que podem ser feitos durante a rodada r. Além disso, cada sorteio realizado está associado a uma probabilidade p de ele ser sorteado com sucesso. Deste modo, é possível calcular o parâmetro τ que representa o número médio de sorteios bem sucedidos por rodada em toda a rede (Eq. 4.3).

$$\tau = p \times W \tag{4.3}$$

É desejado que esse número de sorteios médio por rodada τ seja fixo e igual para toda a rede. O *stake* total W é um valor variável que depende dos recursos financeiros investidos pelos participantes, mas é fixo dentro de uma rodada, como explicado mais adiante.

A probabilidade de haver exatamente k sorteios bem sucedidos na rodada r segue uma distribuição binomial que pode ser calculada pela Eq. 4.4, onde p é a

probabilidade de sucesso definida acima.

$$B(k, W, p) = \binom{W}{k} p^k (1-p)^{W-k} \tag{4.4}$$

Seja $\sigma_i^{(r)}$ o resultado da assinatura de $U_i^{(r)}$ com a chave privada do nó i, denotada por Pr_i,

$$\sigma_i^{(r)} = \{U_i^{(r)}\}_{Pr_i} \tag{4.5}$$

e seja $B(k, w_i, p)$ a probabilidade de ocorrerem exatamente k sorteios bem sucedidos em w_i tentativas.

$$B(k, w_i, p) = \binom{w_i}{k} p^k (1-p)^{w_i-k} \tag{4.6}$$

A partir dos valores de $\sigma_i^{(r)}$, do *stake* w_i e da probabilidade p, cada nó i realiza localmente o sorteio criptográfico da rodada r por meio do Algoritmo 1, adaptado a partir da versão original apresentada por Gilad et al. (GILAD et al., 2017).

Algoritmo 1: Sorteio $(\sigma_i^{(r)}, w_i, p)$

Entrada: assinatura do *hash* $U_i^{(r)}$, *stake* do nó i e probabilidade p
Saída: número j de sorteios bem sucedidos
$q \leftarrow H(\sigma_i^{(r)})/(2^n - 1)$
$j \leftarrow 0$
while $q > \sum_{k=0}^{j} B(k, w_i, p)$ **do**
    $j++$
end
return j

Ao requerer a assinatura do *hash* do nó, o sorteio passa a ser realizado com a utilização de um parâmetro conhecido apenas pelo nó i (dono da chave privada Pr_i), pois apenas ele pode realizar tal assinatura (mas todos podem verificá-la com a chave pública Pu_i). Além disso, tal assinatura é realizada no exato momento em que o nó i participa localmente do sorteio (rodada r). Esse valor da assinatura é, portanto, desconhecido pelo restante da rede até que o nó i divulgue o bloco, caso seja sorteado. Deste modo, os outros nós da rede não podem calcular de antemão quais nós serão sorteados na rodada r, evitando assim que nós desonestos realizem tentativas de subornos ou dificultem as ações dos próximos nós que serão sorteados.

Ao utilizar a versão assinada do parâmetro $U_i^{(r)}$, o CPoS faz uma mudança no mecanismo de sorteio original, o qual utiliza funções verificáveis (VRFs) para ocultar a identidade dos nós sorteados (MICALI; RABIN; VADHAN, 1999). A aplicação da função *hash* H ao valor $\sigma_i^{(r)}$ busca garantir que o tamanho de n bits seja mantido na saída, independentemente do tamanho da assinatura.

O objetivo do Algoritmo 1 é determinar o número de sorteios bem sucedidos para o nó i. No início é calculado um número pseudoaleatório q entre 0 e 1. Esse número depende exclusivamente de $H(\sigma_i^{(r)})$ e, quanto maior seu valor, maior é a probabilidade do nó i ter algum sorteio bem sucedido. No laço, o intervalo $[0, 1]$ é dividido em subintervalos, onde a fronteira de cada subintervalo é determinada pela probabilidade de até j sorteios bem sucedidos ocorrerem em w_i tentativas. Neste sentido, na primeira execução do laço quando $j = 0$, o objetivo é verificar se o valor aleatório q é maior que a fronteira do primeiro subintervalo, ou seja, é verificado se q é maior que a probabilidade de nenhum sorteio bem sucedido em w_i tentativas ($q \stackrel{?}{>} (1-p)^{w_i}$). Caso q atenda a condição, o algoritmo faz $j = 1$, pois agora sabe que o nó i possui um valor aleatório q maior que o mínimo necessário para ser sorteado pelo menos uma vez ou, de forma equivalente, vencer a fronteira do primeiro subintervalo. Assim como o valor aleatório q, o *stake* do nó (w_i) também tem contribuição significativa no sorteio. Quanto maior o seu valor, mais subintervalos serão produzidos em [0,1] e, consequentemente, as fronteiras destes intervalos serão mais próximas e menores, onde o número de subintervalos criados é $w_i + 1$.

Na segunda iteração do laço com $j = 1$, a condição verifica se q é também capaz de superar a fronteira do segundo subintervalo. Isto é correspondente a verificar se q é maior que a probabilidade de 0 ou 1 sucessos em w_i sorteios dada por $\sum_{k=0}^{1} B(k, w_i, p) = (1-p)^{w_i} + w_i p(1-p)^{w_i-1}$. Caso a condição seja verdadeira, j passa a valer 2, indicando que o valor aleatório q é capaz de superar o segundo subintervalo. O laço termina quando é encontrado um subintervalo que q não é capaz de superar. Na condição limite temos $q = 1$ e, neste caso, o laço permanece em execução até $j = w_i$, pois $\sum_{k=0}^{w_i} B(k, w_i, p) = 1$. Diz-se que o nó i foi sorteado j vezes para produzir o próximo bloco na rodada r quando o Algoritmo 1 retorna j.

A Fig. 4.3 apresenta a execução do algoritmo 1 para dois nós participantes por meio de um exemplo mais didático, considerando $p = 0,5$. O nó 1 tem 4 *stakes* ($w_1 = 4$), ou seja, 4 moedas associadas a um endereço controlado por ele. Já o nó 2, nas mesmas condições, controla 8 *stakes* ($w_i = 8$). É possível notar que nessas condições, são produzidos

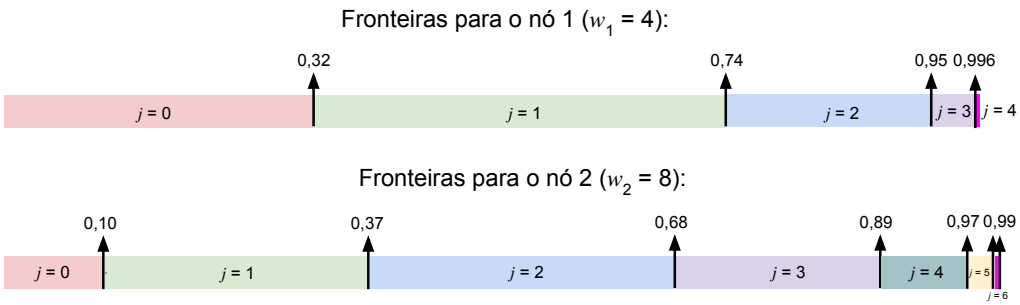

Figura 4.3 – Com o dobro do *stake*, o nó 2 tem maior probabilidade de ser sorteado, já que existem mais subintervalos e os valores das fronteiras são menores.

mais subintervalos a partir da execução do algoritmo 1 para o nó 2 e os números que

definem as fronteiras destes subintervalos são menores. Em outras palavras, é esperado que o nó 2 seja sorteado com maior probabilidade quando comparado ao nó 1, pois com o dobro do *stake* disponível basta que ele obtenha $q > 0,10$ para ser sorteado ao menos uma vez. Por outro lado, o nó 1 precisa de pelo menos $q > 0,32$ para ter ao menos um sorteio bem sucedido.

De maneira geral, para ser sorteado j vezes por meio do Algoritmo 1, o valor pseudoaleatório q deve ser maior ou igual a fronteira do intervalo j. O gráfico da Fig. 4.4 mostra o menor valor que o parâmetro pseudoaleatório q deve assumir (q_{min}) no algoritmo 1, para que seja retornado no mínimo j sorteios bem sucedidos, considerando dois valores diferentes para p e w_i. Sabe-se que, para τ fixo, a probabilidade p varia em função do

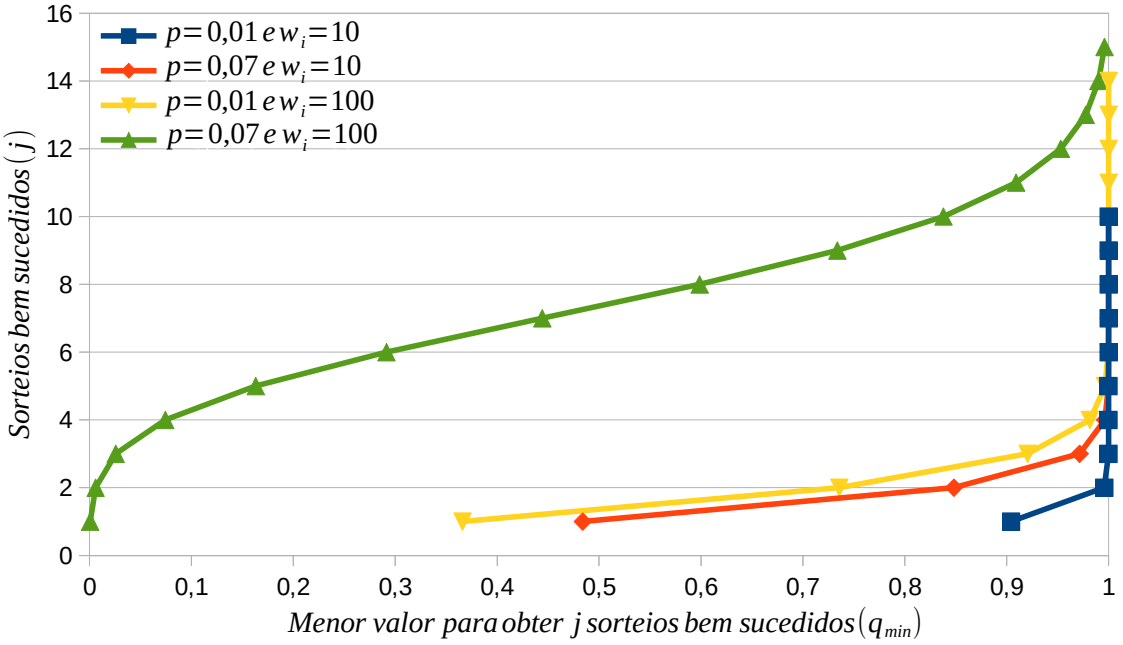

Figura 4.4 – O valor pseudoaleatório q_{min} é o menor q necessário para se obter j sorteios bem sucedidos.

total de *stake* (W). Assim, se p diminui, é porque existem mais *stakes* disponíveis, o que obriga os nós a aumentarem seus *stakes* individuais para manter suas capacidades de serem sorteados. Isso é mostrado no gráfico tomando, por exemplo, $w_i = 100$ e observando que é necessário um q_{min} menor quando a probabilidade de sorteio é $p = 0,07$. Além disso, o *stake* do nó (w_i) influencia o valor q_{min} capaz de produzir j sorteios bem sucedidos. Tomando $p = 0,01$, um nó com $w_i = 100$ é sorteado pelo menos uma vez se o seu valor pseudoaleatório q atender $q \geq q_{min} \simeq 0,39$. Por outro lado, um nó com $w_i = 10$ é sorteado ao menos uma vez se $q \geq q_{min} \simeq 0,9$ também para $p = 0,01$.

4.2.2 *Hash* de prova

Para cada um dos j sorteios bem sucedidos podemos calcular um segundo *hash* chamado de *hash* de prova (Eq. 4.7)).

$$H_t^{(r)} = H(U_i^{(r)}||t) \tag{4.7}$$

O parâmetro t é um número inteiro definido no intervalo $1 \leq t \leq j$. Assim, o *hash* de prova vincula o *hash* do nó ($U_i^{(r)}$) calculado na rodada r a cada um dos j sorteios bem sucedidos. Se $j > 1$ então teremos pelo menos dois sorteios bem sucedidos na rodada e o nó i pode criar diferentes *hashes* de prova. Qualquer um destes *hashes* são válidos, já que foram produzidos a partir de algum elemento t no intervalo definido.

A possibilidade do nó ser sorteado mais de uma vez aumenta sua probabilidade de produzir o próximo bloco que será mantido na visão local dos nós, como será mostrado mais adiante; entretanto, o nó deve escolher apenas um *hash* de prova entre os possíveis e, com isso, produzir apenas um bloco na rodada. Neste momento, torna-se necessário definir um critério para que o nó criador divulgue apenas um *hash* de prova, mantendo uma única relação possível entre o bloco produzido e sua identidade na rodada r. Este critério deve tornar possível que os outros nós da rede validem o *hash* de prova escolhido pelo nó criador.

Isso é importante para que todos os nós da rede possam concordar sobre qual *hash* de prova deve ser aceito para o caso onde o criador tenha sido sorteado mais de uma vez ($j > 1$). Para resolver este problema, Gilad et al. (GILAD et al., 2017) definem o conceito de *hash* de prioridade. Dentre os *hashes* $H_t^{(r)}$ produzidos pelo nó i durante a rodada r, o *hash* de prioridade ($H_{min}^{(r)}$) é aquele de menor valor, ou seja, $H_{min}^{(r)} = min(H_t^{(r)})$ para $1 \leq t \leq j$. Assim, um nó pode divulgar na rede apenas o seu *hash* de prioridade, descartando todos os outros que porventura poderiam ser criados utilizando outro elemento t.

Qualquer nó na rede pode validar um *hash* de prova e verificar se ele é de fato o *hash* de prioridade de um nó sorteado. Para isso, os nós devem primeiramente executar o Algoritmo 1 com os parâmetros de entradas iguais aos utilizados pelo nó i (nó que criou o bloco). O *stake* do nó i é obtido por meio da soma dos valores de todas as transações destinadas ao endereço Id_i que foram confirmadas dentro do período anterior. A composição do *stake* de um nó qualquer é explicada em detalhes na próxima seção. Os outros parâmetros necessários para a validação do *hash* de prova são obtidos a partir do cabeçalho do bloco divulgado (r, Id_i e H_{fc}). Após a execução do Algoritmo 1 com os mesmos parâmetros do nó criador, são obtidos j sorteios bem sucedidos pelos nós da rede e, para cada valor t no intervalo $1 \leq t \leq j$, são calculados *hashes* $H_t^{'(r)}$. Assim, se os nós calcularem algum *hash* capaz de produzir $H_t^{'(r)} = H'_{min} = H_t^{(r)}$ para $1 \leq t \leq j$, o *hash* de prova divulgado pelo nó i é considerado válido. Portanto, no CPoS um nó sorteado pode

divulgar apenas um bloco por rodada, mesmo que seja sorteado mais de uma vez. Este bloco deve estar associado ao *hash* de prioridade, de modo que blocos divulgados com outros *hashes* de prova não são aceitos pelos outros participantes da rede.

Por fim, o comportamento do número de blocos produzidos por rodada em função do aumento do número de nós da rede (N) e do número esperado de sorteios bem sucedidos (τ) é mostrado no gráfico da Fig. 4.5, construído a partir de simulações do mecanismo de sorteio para diferentes números de nós, considerando que o *stake* total está igualmente dividido entre eles. De acordo com o gráfico, se τ é fixado e o tamanho

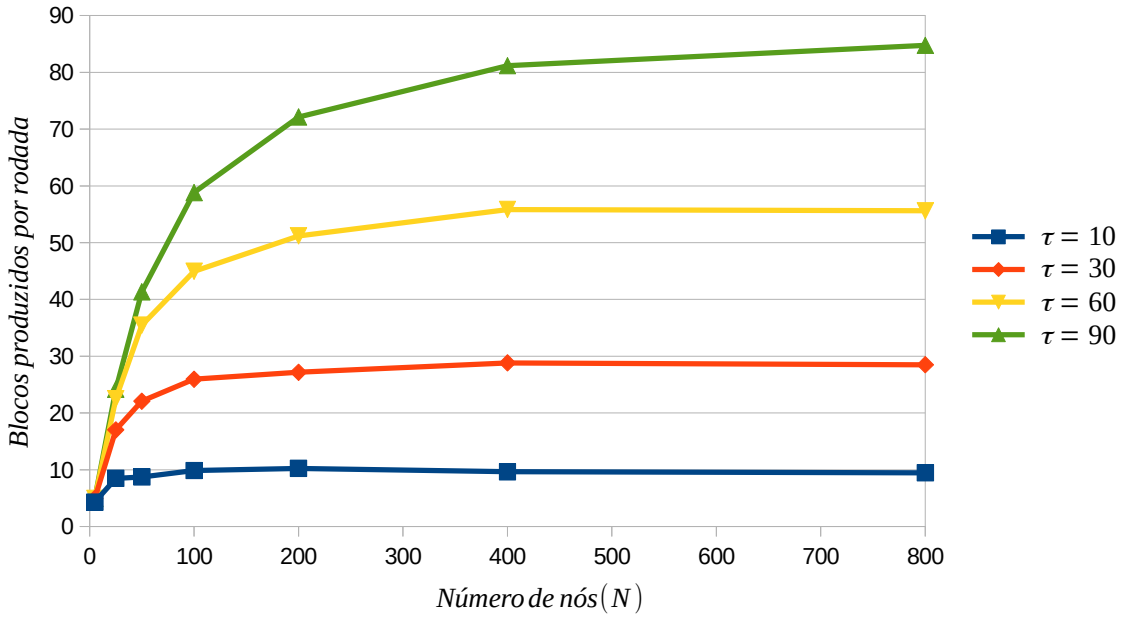

Figura 4.5 – Blocos produzidos por rodada são limitados pelo valor de τ.

da rede aumenta, algo que é esperado na prática, o número total de blocos produzidos por rodada também será limitado pelo valor de τ. Isso significa que a rede pode crescer indefinidamente, mas a produção de novos blocos por rodada não irá acompanhar este crescimento, ficando limitado pelo parâmetro τ e tornando o mecanismo escalável.

Por outro lado, quando τ é aumentado o número de blocos produzidos por rodada também aumenta. Mas este aumento é também limitado pelo número N de nós na rede, já que existe o limitante básico de que cada nó, mesmo que produza vários *hashes* de prova, só poderá produzir um bloco, usando o menor de seus *hashes*. Assim, o número de blocos produzidos, mesmo que τ seja alto, não supera N e, por isso, vemos que não é útil elevar τ para além do número de nós que formam a rede que executa o CPoS.

4.2.3 Divulgação do bloco produzido

Um bloco no CPoS possui um cabeçalho, onde estão disponilizados os parâmetros utilizados na sua construção e o payload com as transações escolhidas pelo criador do

bloco, a partir de um conjunto (*pool*) de transações disponíveis para validação e inserção em blocos (Fig.4.6).

<table>
<tr><td>Hash do bloco
H_{block}</td><td>Hash do bloco anterior
H_{prev}</td><td colspan="2">Hash do primeiro bloco confirmado
H_{fc}</td></tr>
<tr><td>Hash de prova
H_t</td><td>Raiz da árvore de Merkle
M_{tree}</td><td colspan="2">Assinatura do hash do bloco
σ_{block_i}</td></tr>
<tr><td>Assinatura do hash do nó
$\sigma_i^{(r)}$</td><td>Endereço
Id_i</td><td>Rodada
r</td><td>Índice do bloco
k</td></tr>
<tr><td colspan="4">Pu_i - chave pública associada ao endereço Id_i e à chave privada Pr_i</td></tr>
<tr><td colspan="4">Transações</td></tr>
</table>

Figura 4.6 – Campos de um bloco no CPoS

O *hash* do bloco (H_{block}) é calculado por todo nó sorteado em uma rodada por meio da Eq. 4.8.

$$H_{block} = H(H_t^{(r)}||H_{prev}||M_{tree}) \tag{4.8}$$

Este *hash* vincula três parâmetros: o *hash* de prova do nó sorteado, o *hash* do bloco anterior da blockchain ($H_{prev} = H_{block}$ anterior) e a raiz da árvore de *Merkle* M_{tree}, que representa todas as transações inseridas no presente bloco (MERKLE, 1990). O hash do bloco é assinado pelo nó i utilizando sua chave privada Pr_i, produzindo a assinatura σ_{block_i} (Eq. 4.9).

$$\sigma_{block_i} = \{H_{block}\}_{Pr_i} \tag{4.9}$$

Por fim, é adicionada a assinatura do hash do nó i ($\sigma_i^{(r)}$) que é utilizada pelos nós da rede para validar o sorteio. A autenticidade dessas assinaturas é garantida pela chave pública do nó Pu_i que está associada ao seu endereço Id_i. Após sua criação, o bloco é transmitido pelo nó criador para todos os seus *peers*.

4.2.4 Controle de *Sybil Attack*

O mecanismo de sorteio do CPoS é resistente ao ataque baseado em identidades *Sybil* (seção 2.4.2), já que a distribuição binomial utilizada no mecanismo de sorteio do CPoS garante que $B(k_1, w_1, p) + B(k_2, w_2, p) = B(k, w_i, p)$, onde $k = k_1 + k_2$ e $w_i = w_1 + w_2$. Neste sentido, a criação de várias identidades não aumenta a probabilidade de um nó ser sorteado, já que este processo é controlado pelo seu *stake* total, que pode estar concentrado em uma ou distribuído entre várias identidades. Em outras palavras, de nada adianta criar várias identidades se tiver que dividir o *stake* do nó entre elas.

4.3 Definição do *stake* em um período

O conceito de período é descrito no mecanismo Casper (BUTERIN; GRIFFITH, 2017) e o seu objetivo principal é controlar o conjunto de validadores que estão participando do comitê de validação. No CPoS o período é utilizado para realizar o controle do *stake* total W disponível. Neste sentido, o período E é um número inteiro que representa um conjunto de M blocos confirmados. Assim, a cada M blocos, seu valor é incrementado em uma unidade. O período $E = 1$ inclui o bloco *um* (gênesis), que é o primeiro bloco considerado confirmado pelo mecanismo. Todo bloco é associado a um período e, para determiná-lo, o índice do bloco é dividido pelo número fixo de blocos existentes em um período (M). Está sendo assumido que todos os blocos têm um índice que é incrementado a cada novo bloco e que não há lacunas entre estes índices.

Para participar do mecanismo de sorteio, a fim de tentar produzir novos blocos no período E, um nó i deve primeiramente comprovar seu *stake*, realizando, por exemplo, uma transferência de uma certa quantidade de moedas para uma conta específica ou para a conta relacionada à própria identidade Id_i do nó. Tal transferência pode ser feita por meio de uma ou mais transações que devem estar confirmadas em algum bloco do período $E - 1$ e que não podem ser alteradas durante o período corrente E.

A Fig. 4.7, ilustra como o *stake* w_i do usuário i é obtido no início do período E. Na figura, as transações são lidas dos blocos confirmados no período $E - 1$ e os valores em transações do tipo 2, associados ao Id de cada nó são buscadas. Assim, a soma dos valores de todas as transações confirmadas no período $E - 1$ e relacionadas ao nó Id_i forma o *stake* desse nó durante o período E. Uma transação do tipo 2 indica que o seu valor deve ser considerado como *stake*, assim, estas transações são diferenciadas de uma operação

convencional. Apenas transações deste tipo são consideradas para formação do *stake* de cada nó.

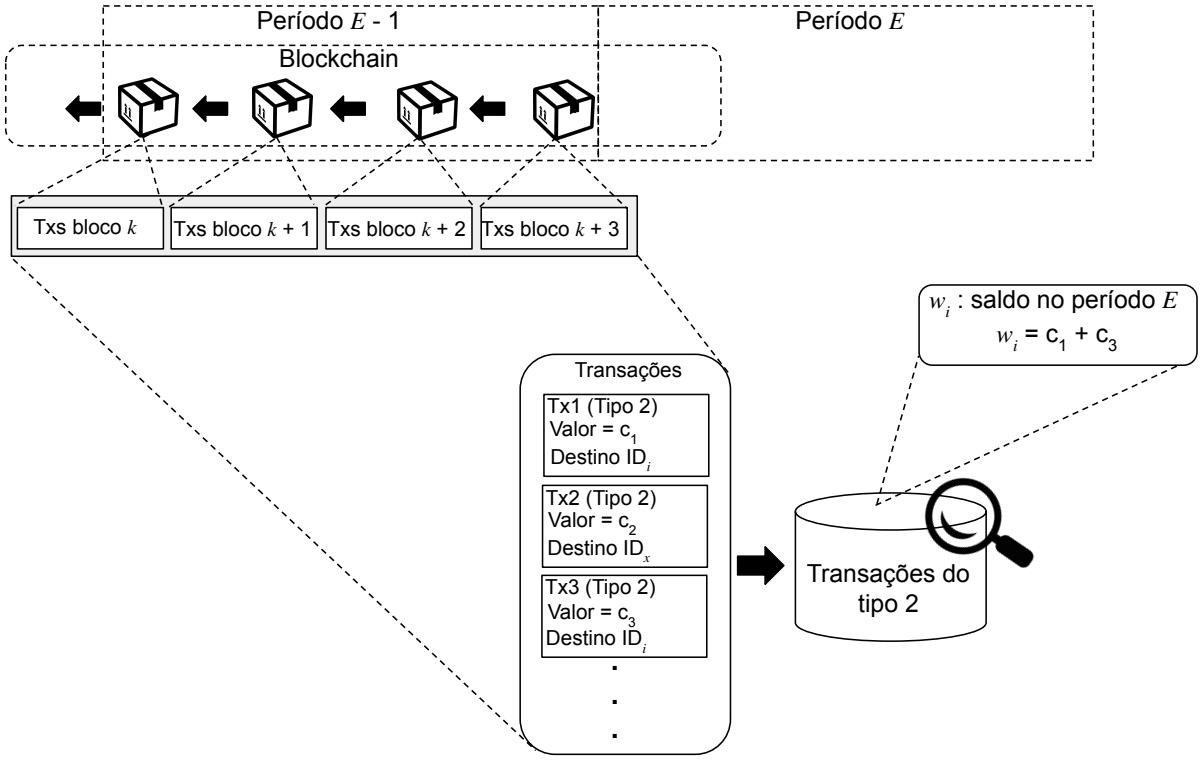

Figura 4.7 – O *stake* de um nó no período E é formado pelas transações do tipo 2 associadas ao endereço de seu controle

No início de cada período, os nós calculam o *stake* total presente na rede (W) por meio da soma de todas as transações do tipo 2 que foram confirmadas no período anterior e que estão presentes na tabela Transações do tipo 2 de seu banco de dados local. Neste momento, é definida a probabilidade de sucesso p (τ/W). Afim de tornar W constante durante o período atual, um nó pode utilizar o seu saldo empregado como *stake* no período $E - 1$ apenas a partir do período $E + 1$. Neste sentido, durante o período E, as movimentações que indicarem como entradas valores de transações do tipo 2 confirmadas no período $E - 1$ são rejeitadas pelos nós participantes. É possível afirmar que a probabilidade de sucesso p é constante em um período, já que τ é um parâmetro definido pelo sistema e W é mantido constante.

O controle do *stake* W em um período e, consequentemente, da probabilidade p, são importantes premissas para o modelo probabilístico de confirmação de blocos, como será mostrado na seção 4.5. Este modelo define probabilidades que são baseadas no fato de que o número esperado de sorteios bem sucedidos seja de fato τ. Neste sentido, é preciso calcular p de maneira precisa, buscando considerar o valor real de todo o *stake* W, que efetivamente é empregado nos sorteios pelos nós. Este é o principal motivo pelo qual o CPoS define o conceito de período e determina que os nós revalidem seus *stakes* antes

do início do próximo período. Estas ações permitem que os nós calculem o *stake* W de maneira precisa e limita a influência de nós que participaram do mecanismo no passado, mas que já não têm *stake* ou interesse em participar dos sorteios nos períodos recentes.

A partir dos conceitos apresentados até este ponto, é possível definir o processo completo seguido por um nó para que ele possa participar dos sorteios em cada rodada do período atual, com o objetivo de produzir novos blocos. Esse processo é ilustrado em um diagrama de sequência que é apresentado na Seção A.1 do Anexo A.

Política de incentivo Como explicado no Capítulo 2, uma política de incentivo eficiente colabora para que haja descentralização do consenso em torno de um grande número de nós, o que permite aumentar a segurança do consenso empregado em blockchains públicas. Neste sentido, a política deve priorizar uma distribuição de recompensas que seja proporcional ao *stake* de cada nó que esteja participando das ações do consenso, ou seja, os nós que tiveram transações do tipo 2 confirmadas no período $E - 1$ devem receber remunerações proporcionais aos valores empenhados através destas transações.

Com o objetivo de aumentar a descentralização, essa distribuição de recompensas deve ser independente do sucesso que estes nós tenham no sorteio de novos blocos, haja vista que essa independência torna a política de incentivo favorável também para nós que não possuem grandes recursos financeiros e que, portanto, não são sorteados com frequência. Assim, uma maior descentralização é esperada, já que a política pode aumentar o interesse dos nós com menos recursos, fato que não ocorre, por exemplo, na política adotada no PoW, que premia apenas um único nó a cada bloco produzido (aquele que vence o desafio da mineração).

Esse tipo de política capaz de promover a descentralização é adotada pelo mecanismo Casper (BUTERIN; GRIFFITH, 2017), onde as recompensas são divididas entre os validadores e aqueles nós que não seguem as regras do mecanismo são punidos com a redução do *stake* depositado como garantia. É desejável que CPoS siga uma política de incentivo semelhante a do Casper, que seja capaz de promover a distribuição das recompensas angariadas durante o processo. Apesar disso, em um primeiro momento, não parece ser necessária a utilização de um sistema de punições.

Por ser um mecanismo que não depende de um comitê de confirmação distribuído, e também pelo fato de promover uma ampla mudança, a cada rodada, no conjunto de nós que são sorteados para produção dos novos blocos, o CPoS pode tolerar a presença minoritária de nós desonestos sem realizar punições, pois estes nós não podem manipular constantemente seu esquema de consenso. Entretanto, a ausência de punições pode ser modificada se for adotado um esquema de controle de *stake* que prevê o depósito do mesmo em uma conta de um contrato inteligente, por exemplo,

tirando-o do controle do nó. Caso o nó não siga as regras determinadas, ele pode ser punido com a não devolução total ou parcial deste *stake*.

4.4 Gerenciamento dos blocos recebidos

Como apresentado na seção 4.2, o número de sorteios bem sucedidos esperado por rodada é τ. Portanto, no geral, mais de um bloco é produzido dentro de uma rodada e, o CPoS deve estabelecer critérios para o gerenciamento destes blocos, no que tange às condições necessárias para aceitá-los nas visões locais dos nós. Cada nó no CPoS mantém localmente apenas uma visão da blockchain, que é definida como o conjunto dos blocos recebidos e escolhidos nas rodadas anteriores mais os blocos recebidos na rodada atual. Na seção 4.5 é definido o conceito de confirmação probabilística, que fornece aos nós a possibilidade de confirmar os blocos recebidos a partir de uma avaliação probabilística que é realizada nas suas visões locais.

Como a blockchain é uma estrutura crescente e semelhante a de uma lista ligada, apenas um bloco de cada índice é mantido na visão local de cada nó, enquanto os outros são removidos da visão por meio da utilização das premissas que serão estabelecidas nesta seção. Assim, o objetivo dessas premissas é definir os critérios para que apenas um dos blocos criados em uma rodada seja mantido na visão local do nó.

A premissa básica de aceitação estabelece que apenas blocos criados com rodada dentro do intervalo de tolerância $[r; r + tol]$ são aceitos. Na notação que define o intervalo, $r + tol$ é a rodada máxima que um bloco recebido pode ter para ser aceito por um nó que, segundo o seu relógio local, está na rodada r. Portanto, diz-se que tol é a tolerância definida pelo mecanismo. Por outro lado, a rodada mínima de um bloco recebido deve ser exatamente r, o que é equivalente a dizer que um nó não aceita blocos com rodadas mais antigas que aquela calculada através do seu relógio. Os detalhes referentes a definição do intervalo de tolerância podem ser vistos no Anexo B.

Como segunda premissa, é definida uma prioridade em relação à rodada de cada bloco recebido dentro do intervalo de tolerância. Assim, entre todos os blocos recebidos com rodada dentro do intervalo, aquele com menor rodada é mantido na visão local do nó, enquanto os outros são removidos. Neste trabalho um bloco B é representado pela notação ${}_vB_r$, onde os subíndices indicam:

- r: rodada de criação do bloco;
- v: *hash* de prova do bloco.

A Fig. 4.8 mostra a evolução da rodada $r + 1$ na visão de um nó, a partir do recebimento de três blocos em diferentes momentos, com $tol = 2$. O bloco ${}_cB_{r+2}$ é

recebido primeiro e, como sua rodada está dentro do intervalo de tolerância da rodada $r+1$ ($[r+1; r+3]$), o bloco é aceito. Logo depois, um novo bloco com rodada $r+3$ é

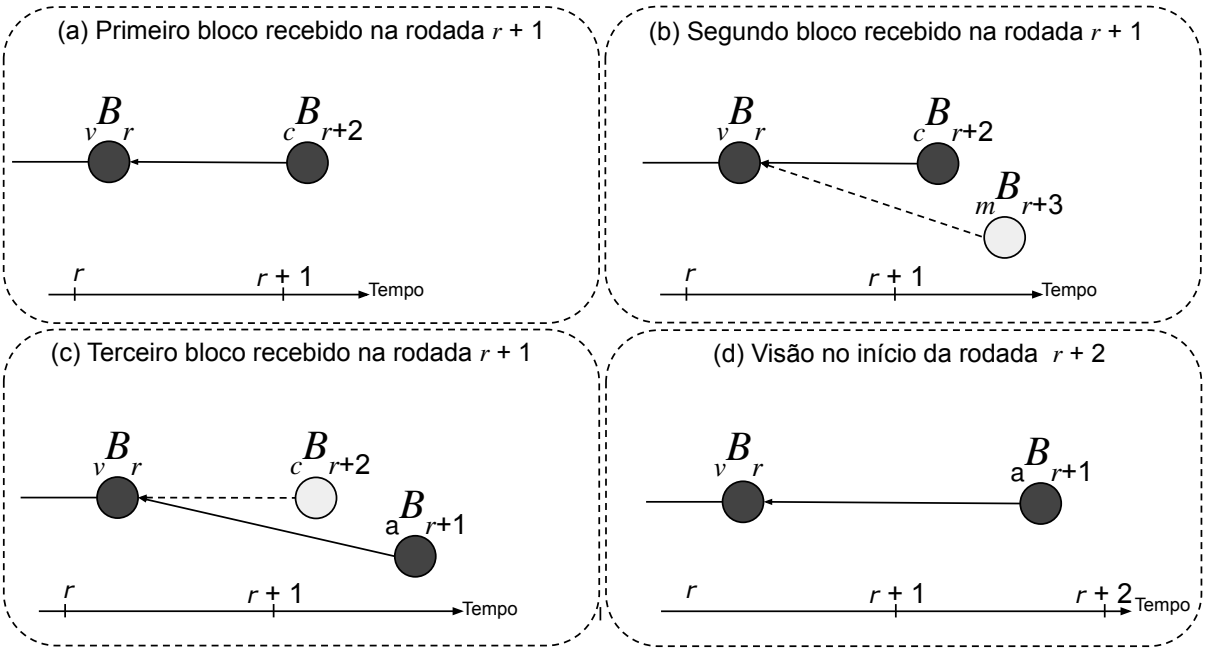

Figura 4.8 – Blocos com rodadas menores e dentro do intervalo de tolerância são aceitos pelo nó e os demais são removidos

recebido (${}_mB_{r+3}$). De acordo com o intervalo de tolerância ($[r+1; r+3]$) sua rodada é válida, mas o nó não aceita o bloco, já que sua rodada é maior do que a do bloco recebido anteriormente na rodada $r+1$, e está, em desacordo com a segunda premissa. Por fim, na visão (c) da figura, em um novo instante de tempo da rodada $r+1$, o bloco ${}_aB_{r+1}$ é recebido e aceito pelo nó, substituindo o bloco ${}_cB_{r+2}$ que até então era o bloco com a menor rodada. A blockchain consolidada pelo nó ao final da rodada $r+1$ é mostrada na visão (d) da figura. Nota-se que apenas o bloco com a menor rodada é mantido.

O critério de aceitação baseado na menor rodada recebida, permite que o nó decida qual bloco deve ser aceito para os casos em que todas as rodadas de criação sejam diferentes. Entretanto, um *fork* é produzido na blockchain sempre que dois blocos são criados com uma mesma rodada. Como existe um esforço no CPoS para que os nós se mantenham sincronizados em relação a rodada atual, é esperado que a maioria dos novos blocos sejam produzidos com a mesma rodada. Além disso, o impacto destes *forks* no desempenho do CPoS foi avaliado por meio da criação do modelo teórico apresentado no Anexo F. Este modelo mostrou que, de fato, o prolongamento de *forks* por muitas rodadas é indesejado, pois colabora com o aumento do tempo necessário para que os blocos sejam confirmados no mecanismo.

Portanto, é preciso definir um critério mais específico, para permitir que um nó decida rapidamente qual bloco produzido deve permanecer em sua visão local e quais podem ser eliminados. É ideal que este novo critério permita que esta decisão ocorra ainda

dentro da rodada atual, evitando a ocorrência de *forks*. Neste sentido, o CPoS passa a utilizar um terceiro critério baseado na extensão do conceito de *hash* de prioridade definido na seção 4.2.

Em uma rodada qualquer todo nó sorteado deve divulgar apenas um bloco, mesmo que seja sorteado mais de vez (devido a um alto *stake*, por exemplo), onde o *hash* de prova deste bloco deve ter o menor valor entre todos os *hashes* de prova produzidos a partir de cada elemento t ($1 \leq t \leq j$). Assim, em uma determinada rodada são transmitidos na rede um conjunto de blocos formados pelos menores *hashes* de prova de cada nó sorteado. A partir deste fato, o novo critério define que os nós mantenham em suas visões locais apenas o bloco associado ao menor *hash* de prova entre todos aqueles recebidos. Desta forma, os nós primeiramente verificam se um bloco recebido possui rodada de criação dentro do intervalo de tolerância e, em seguida, se sua rodada é menor que a rodada do bloco aceito atualmente, caso ele exista. Se estas verificações forem incapazes de definir qual bloco deve ser considerado, os nós aplicam o terceiro critério, verificando se o novo bloco possui *hash* de prova menor que aquele atualmente aceito pelo nó. O novo bloco substitui o anterior se seu *hash* de prova for de fato menor.

Com isso, os nós decidem sobre qual bloco deve permanecer em sua visão local ainda dentro de sua rodada de criação, alcançando o objetivo de não prolongar *forks* na blockchain para mais de uma rodada. Na Fig. 4.9, é apresentada a visão do nó i em três momentos distintos, onde foram aplicados os critérios de escolha aos blocos recebidos, considerando $tol = 2$. Na visão (a) da Fig. 4.9, o nó está na rodada $r + 1$ e dois blocos, sucessores do seu último bloco aceito são recebidos (${}_cB_{r+2}$ e ${}_aB_{r+1}$). Em um primeiro momento o bloco ${}_cB_{r+2}$ é aceito, já que chega antes e possui rodada válida para $tol = 2$, porém logo depois ele é substituído pelo bloco ${}_aB_{r+1}$ que possui rodada menor. Este novo bloco é mantido até o final da rodada $r + 1$. Neste caso, o *hash* de prova não é utilizado como critério de decisão, já que o critério da menor rodada tem prioridade em relação ao menor *hash* de prova e, neste caso, ele é suficiente para definir a escolha.

Logo depois é iniciada a rodada $r + 2$, como mostrado na visão (b) da Fig. 4.9. Nesta rodada, o nó recebe dois blocos com rodadas iguais (${}_gB_{r+2}$ e ${}_qB_{r+2}$) e que estão dentro do intervalo de tolerância estabelecido ($[r + 2; r + 4]$). Deste modo, a decisão passa a ser tomada considerando o *hash* de prova dos blocos recebidos, ou seja, aquele bloco associado ao menor *hash* de prova é o escolhido. Na figura é representado o caso em que $q < g$ e, portanto, o bloco ${}_qB_{r+2}$ permanece na visão local do nó, como mostrado na visão (c) da figura.

Com base nos critérios definidos nesta seção, um bloco ${}_vB_r$ que ocupa a posição (altura) k da blockchain (Fig. 2.1) é aceito por um nó na rodada x se qualquer uma das três condições a seguir for atendida:

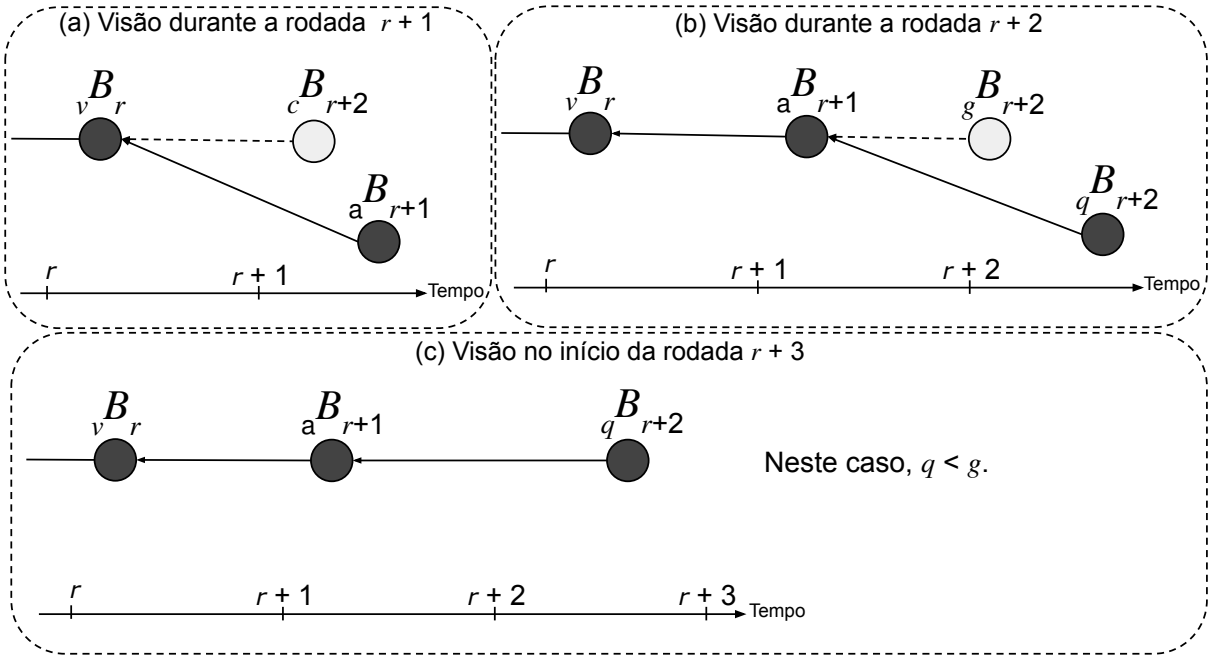

Figura 4.9 – Aplicação dos critérios de decisão para definir qual bloco deve permanecer na visão local da blockchain

- **condição 1:** $r \in [x; x + tol]$ e não há outro bloco na posição k da blockchain;
- **condição 2:** $r \in [x; x + tol]$, há um bloco ${}_uB_r$ na posição k da blockchain e $v < u$;
- **condição 3:** $r \in [x; x + tol]$, há um bloco ${}_uB_{r'}$ na posição k da blockchain e $r < r'$.

No diagrama de estados da Fig. 4.10 é mostrado como o CPoS realiza a funcionalidade *listen* para obter novos blocos aplicando os critérios descritos nesta seção, desde o recebimento do bloco através do *socket* conectado ao *peer* até sua aceitação ao rejeição. Sempre que um novo bloco é recebido, por meio de um socket de rede conectado a algum *peer* do nó no estado *Waiting new block*, o nó instancia um novo objeto da classe bloco passando como parâmetros as informações recebidas no socket. No estado *Checking if exists last block* é verificado se o nó conhece o bloco imediatamente anterior ao novo bloco recebido (chamada da função *existLastBlock*). Isto é necessário para verificar se o nó tem a mesma visão que o criador do bloco em relação a blockchain, ou seja, se ele de fato recebeu um bloco sucessor de seu último bloco conhecido. No estado *Rejecting block*, o bloco não é aceito caso o nó não conheça seu antecessor, porém ele é armazenado durante algumas rodadas em um *buffer* na memória para o caso de uma ressincronização de cadeia. Este processo é detalhado na seção 5.2.

No estado *Validating block round* é verificado se a rodada do bloco é aceita como válida pelo nó. Para isso, a função *checkBlockRound()* retorna *True* se, de acordo com a Eq. (4.1), a rodada do bloco for considerada válida. Caso contrário a função retorna *False*. Caso a rodada do bloco não seja válida, ou seja, $block.round \notin [round; round + tol]$,

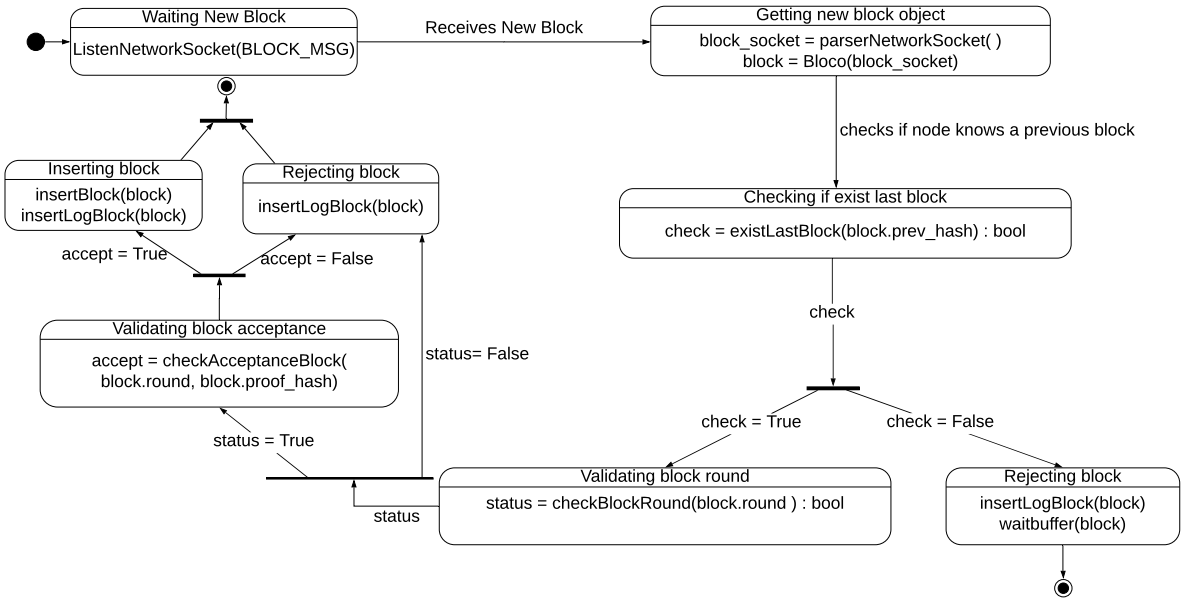

Figura 4.10 – Diagrama de estado do processo de recebimento de um bloco no CPoS

o diagrama segue para o estado de *Rejecting block* e o bloco não é aceito. Para uma rodada válida, é verificado se as condições de aceitação 2 e 3 são satisfeitas por meio da execução da função *checkAcceptanceBlock* no estado *Validating block acceptance*. Por fim, o bloco é inserido no estado *Inserting block*, caso a função retorne que esta premissa foi satisfeita. A Seção A.2 do Anexo A apresenta a interação entre as instâncias de diferentes classes, que participam do processo de recebimento de um bloco no CPoS, por meio de um diagrama de sequência.

Os critérios definidos nesta seção colaboram para que o CPoS alcance uma convergência, na qual a maioria dos nós possam concordar sobre a visão da blockchain. Um conjunto de regras que garante a convergência e a continuidade (evolução) de uma blockchain é conhecido como *Plausible Liveness* e uma discussão mais detalhada sobre seus detalhes é feita mais adiante. Por fim, os critérios desta seção são aplicados apenas aos blocos recebidos pelo nó em cada rodada. Neste sentido, é possível que existam diferentes visões da blockchain, já que alguns blocos podem não ser recebidos por todos os nós. Portanto, os blocos aceitos não são confirmados de maneira instântanea pelo mecanismo. A confirmação ocorre algumas rodadas depois da aceitação do bloco, caso a visão do nó seja de fato compartilhada entre os outros nós da rede, como será discutido na seção 4.5. Neste sentido, o CPoS é um mecanismo de consenso baseado em confirmações atrasadas, ou seja, são necessários novos ciclos (rodadas) para que a confirmação de um bloco ocorra.

Plausible Liveness Trata-se de um conjunto de regras necessárias para mecanismos de consenso distribuídos, que quando seguidas pelos nós honestos, permitem que o sistema tenha convergência e continuidade (MOINDROT; BOURNHONESQUE, 2017). No contexto das blockchains públicas, a continuidade está associada à capacidade do

mecanismo em definir critérios para que os nós escolham um mesmo bloco já inserido na cadeia e, a partir desta convergência de escolha, possam definir um sucessor para este bloco que futuramente será indexado à mesma cadeia por meio do processo de consenso.

No *PoW*, os critérios levam os mineradores a escolher o último bloco produzido de suas visões locais, pois apenas nós que produzam blocos na cadeia mais longa são recompensados. Assim, mesmo que existam nós com visões diferentes da blockchain, eles procuram empregar seus esforços computacionais sempre para criar um sucessor para seu último bloco conhecido, pois sabem que esta é única maneira possível de receber recompensas. Como no *PoW* é esperado que a maioria dos nós compartilhem a mesma visão da blockchain, é correto afirmar que os nós na maior parte do tempo empregam seus esforços computacionais para estender a mesma cadeia, pois partem de um mesmo bloco. Este exemplo, ilustra como as regras definidas pelo PoW são capazes de garantir a continuidade da blockchain a partir de uma convergência de escolha.

No CPoS, um nó sorteado também deve tentar produzir a cada rodada r um sucessor para o seu último bloco conhecido. Essa é a opção natural, pois qualquer escolha diferente leva o nó a produzir um bloco em uma nova cadeia (*fork*) que não será aceito pelas condições definidas nesta seção. Entretanto, é necessário garantir que de fato os nós sorteados na rodada r possam criar um sucessor para apenas um bloco e que esse seja o último bloco de suas visões locais. A unicidade da escolha é garantida por meio da obrigatoriedade do nó divulgar apenas um bloco, mesmo quando é sorteado mais de uma vez, como explicado na seção 4.2. Além disso, o mecanismo de sorteio do CPoS é resistente a *Sybil Attack*, sendo capaz de evitar que um nó amplie suas chances de ser sorteado por meio da produção de diferentes identidades. Estas condições fazem com que os nós tenham apenas uma oportunidade de produção em cada rodada em que são sorteados. Desta forma, os nós são forçados a investirem o sorteio bem sucedido na criação de um bloco que tenha chances de participar do processo de escolha descrito nesta seção, ou seja, é de interesse dos nós sorteados que eles criem sucessores para o último bloco aceito em suas visões locais.

Para completar a característica de *Plausible Liveness* do CPoS, é preciso definir condições para que os nós compartilhem a mesma visão da blockchain, de modo a garantir que exista convergência em relação ao bloco escolhido pelos nós a cada rodada. Estas condições são alcançadas primeiramente pela definição de uma rodada com tempo de duração adequado, capaz de permitir que os blocos sejam recebidos pelos nós dentro do intervalo de tolerância definido. Segundo, é preciso evitar que um conluio de nós desonestos controle totalmente a transmissão de blocos dentro da rede *peer-to-peer*, o que é conhecido na literatura como *eclipse attack* . Segundo Heilman et al. (HEILMAN et al., 2015), o eclipse ataque ocorre quando o adversário

consegue criar um isolamento entre a vítima e o restante das comunicações que estão ocorrendo na blockchain. O atacante então pode filtrar quais blocos são efetivamente enviados para a vítima, produzindo uma visão local diferente daquela que realmente está ocorrendo no restante da rede. Bissias et al. (BISSIAS et al., 2016) afirma que o ataque pode ser amplificado a partir de uma *botnet* controlada pelo adversário. Neste caso, as chances da vítima se conectar apenas com *peers* do próprio atacante aumentam.

No CPoS este ataque é controlado por meio de um esquema aleatório para que os nós escolham seus *peers*, evitando que apenas participantes desonestos façam parte do conjunto de *peers* de um nó qualquer. Como pode ser visto em detalhes no Anexo C, a partir da adoção deste esquema, é possível afirmar que a única forma de um nó desonesto se conectar a um alvo específico é por meio de força bruta.

4.5 Confirmação probabilística dos blocos

As condições de aceitação definidas na seção 4.4 são aplicadas apenas aos blocos recebidos pelos nós em cada rodada. Considerando um cenário ideal de operação, onde todos os blocos produzidos são recebidos por toda a rede dentro dos limites estabelecidos pelo intervalo de tolerância, essas condições garantem ao CPoS uma confirmação determinística para todo bloco que permanece na visão local dos nós ao final de cada rodada, haja vista que nestas condições, é garantida a existência de apenas uma visão. Entretanto, em situações reais, existem falhas de transmissão na rede e presença de nós desonestos que podem levar os nós a divergências sobre qual visão da blockchain deve de fato ser seguida. Em outras palavras, o CPoS deve operar em uma infraestrutura Bizantina (LAMPORT; SHOSTAK; PEASE, 1982), e as condições de aceitação definidas passam a não ser suficientes para garantir que um mesmo bloco seja escolhido por todos os nós da rede.

Portanto, a produção de novos blocos pode ser realizada a partir de visões diferentes da blockchain. Essa seção define o conceito de confirmação probabilística, onde um nó i pode medir se sua visão atual da blockchain é de fato seguida pelos outros nós da rede, sem a utilização de um esquema baseado em comitês de confirmação. A partir desta medição, o nó i confirma um bloco sempre que a probabilidade de existirem visões conflitantes envolvendo este bloco seja suficientemente baixa. Diz-se que dois nós têm a mesma visão no início de uma rodada r quando eles têm, em suas respectivas cópias da blockchain, os mesmos blocos desde o bloco inicial (gênesis).

Quando o nó i recebe algum bloco produzido em uma rodada, ele ganha informações a respeito do nó que o produziu e, a partir do Algoritmo 1, ele pode calcular o número de vezes que este nó foi sorteado durante a rodada em questão. Além disso, em uma rodada r, o nó i escolhe o bloco ${}_vB_r$ para permanecer na sua visão local aplicando

as condições definidas na seção 4.4 e, para cada rodada x ($x > r$), ele escolhe da mesma forma os sucessores do bloco ${}_vB_r$. A partir da escolha do bloco que deve permanecer em sua visão local na rodada r (${}_vB_r$), o nó i passa a somar o número de sorteios bem sucedidos ($s_v^{(x)}$) dos nós que produzem blocos na mesma visão que a dele (Eq. 4.10). Nesta equação, o somatório representa o número total de blocos recebidos na rodada x; já o subíndice y representa os nós que produziram blocos na mesma visão do nó i, ou seja, são os únicos nós cujo número j de sorteios bem sucedidos será somado na rodada x ($j_y^{(x)}$) pelo nó i.

$$s_v^{(x)} = \sum_y j_y^{(x)} \tag{4.10}$$

Assim, apenas blocos sucessores do bloco escolhido pelo nó i na rodada $x-1$ têm seus números de sorteios bem sucedidos somados em $s_v^{(x)}$ pelo nó i, garantindo que este valor reflita uma medida do número total de sorteios bem sucedidos que foram realizados por nós que compartilham exatamente a mesma visão da blockchain que ele.

A extensão deste conceito é o cálculo do número médio de sorteios bem sucedidos ($\bar{s}_v^{(x)}$) recebidos pelo nó i. Este número médio é calculado entre a rodada r de criação do bloco ${}_vB_r$ e o final da rodada x (na iminência da rodada $x+1$). Portanto, para $\Delta_r = x - r$, podemos definir a Eq. 4.11.

$$\bar{s}_v^{(x)} = \frac{\sum_{t=r+1}^{x} s_v^{(t)}}{\Delta_r} \tag{4.11}$$

O valor $\bar{s}_v^{(x)}$ aumenta quando mais nós compartilham a mesma visão do nó i em relação ao bloco ${}_vB_r$, ou seja, quando são produzidos sucessores do bloco ${}_vB_r$ a partir de uma mesma visão para cada rodada no intervalo Δ_r. Por outro lado, o valor diminui quando esta visão não é considerada pelos nós que compõem a rede, indicando que em alguma rodada anterior houve divergência entre o bloco que foi escolhido pelo nó i e aquele que o restante da rede está de fato considerando.

O exemplo da Fig. 4.11 mostra a visão dos últimos blocos recebidos pelo nó i quando ele está no início de duas rodadas diferentes. Na visão (a), o nó está no início da rodada $r+2$ e, durante a rodada $r+1$ ele recebeu três sucessores do bloco ${}_vB_r$ (${}_aB_{r+1}$, ${}_gB_{r+1}$ e ${}_cB_{r+1}$). Além disso, o nó i também recebe o bloco ${}_uB_{r+1}$. Como este bloco não é sucessor do bloco ${}_vB_r$ ele não contribui com a visão atual, já que foi produzido por um nó com outra visão da blockchain. Desta forma, o nó i considera que o antecessor do bloco ${}_uB_{r+1}$ é desconhecido localmente e o mesmo é descartado. Entre os sucessores do bloco ${}_vB_r$, o nó i define, a partir das condições de aceitação, que o bloco com o menor *hash* de prova deve permanecer em sua visão local (indicado pelo círculo preto com seta contínua), enquanto os outros são utilizados apenas para compor a soma $s_v^{(r+1)}$ sendo descartados logo em seguida (indicado pelos círculos cinzas com setas tracejadas). Portanto, a soma $s_v^{(r+1)}$ é o número de vezes que os nós criadores dos blocos sucessores de ${}_vB_r$ foram sorteados

durante a rodada $r+1$. Deste modo, o número médio de sorteios bem sucedidos que confirmam o bloco ${}_vB_r$ no início da rodada $r+2$ é dado por

$$\bar{s}_v^{(r+1)} = \frac{s_v^{(r+1)}}{(r+1)-r} = 9.$$

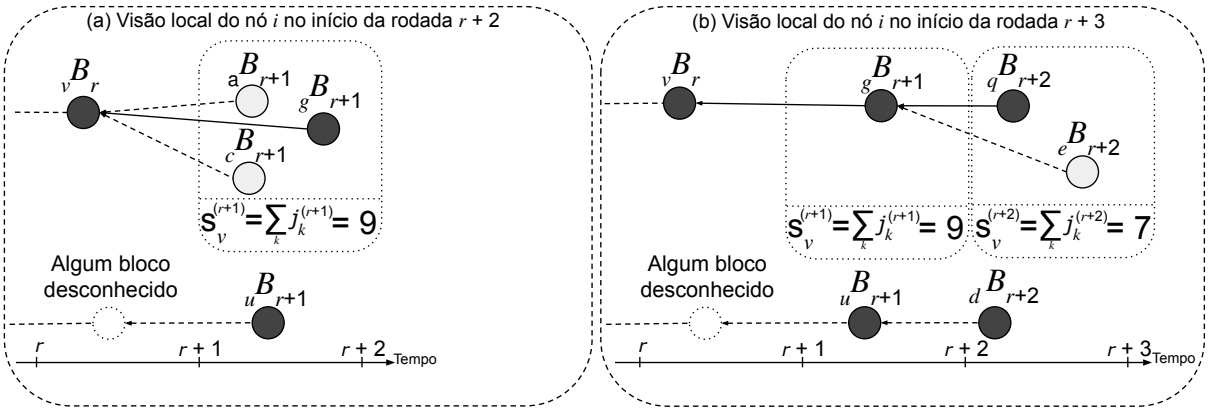

Figura 4.11 – Sorteios bem sucedidos recebidos pelo nó i em um intervalo de rodadas

O nó i recebe também dois sucessores do bloco ${}_gB_{r+1}$ (${}_eB_{r+2}$ e ${}_qB_{r+2}$) durante a rodada $r+2$, ou seja, blocos que foram produzidos por nós com a mesma visão que a dele. Da mesma forma, ele calcula o número de sorteios bem sucedidos recebidos na rodada $r+2$ que contribuem para a confirmação do bloco ${}_vB_r$. No exemplo da Fig. 4.11 este cálculo resulta em $s_v^{(r+2)} = \sum_y j_y^{(r+2)} = 7$. O bloco ${}_dB_{r+2}$ é sucessor do bloco ${}_uB_{r+1}$ e ambos são produzidos por nós que não compartilham a mesma visão do nó i, portanto, de maneira similar o bloco ${}_dB_{r+2}$ é descartado e o número de sorteios bem sucedidos do nó que o produziu não é contabilizado em $s_v^{(r+2)}$. O número médio de sorteios bem sucedidos produzidos por nós que compartilham a mesma visão do nó i e que, portanto, são sucessores do bloco ${}_vB_r$ ao final da rodada $r+2$ é dado por

$$\bar{s}_v^{(r+2)} = \frac{s_v^{(r+1)} + s_v^{(r+2)}}{(r+2)-r} = 8.$$

Ao final da rodada $r+2$ o nó i também calcula o número médio de sorteios bem sucedidos que confirmam sua visão a partir da rodada de criação do bloco ${}_gB_{r+1}$. Este valor é dado por

$$\bar{s}_g^{(r+2)} = \frac{s_g^{(r+2)}}{(r+2)-(r+1)} = 7.$$

Vale ressaltar que os únicos blocos que são considerados na soma $s_v^{(x)}$ ao final da rodada x são os sucessores diretos do último bloco aceito pelo nó i durante a rodada $x-1$. Para ilustrar essa afirmação, a Fig. 4.12 apresenta a continuação da blockchain até o início da rodada $r+4$. Os blocos ${}_mB_{r+3}$ e ${}_oB_{r+3}$ são sucessores diretos do bloco mantido

na visão local pelo nó i na rodada anterior $(r+2)$ e, assim, foram produzidos por nós com a mesma visão. Deste modo, eles contribuem com a visão que inclui o bloco ${}_vB_r$ e por isso são considerados na soma $s_v^{(r+3)} = 6$. Por outro lado, o bloco ${}_wB_{r+3}$ é sucessor de um bloco conhecido pelo nó i, mas que não foi mantido em sua visão local (${}_eB_{r+2}$). Esse fato mostra que o nó que o produziu contribui com outra visão da blockchain, já que não considerou o mesmo bloco antecessor na rodada anterior. Isso ocorre quando o bloco ${}_qB_{r+2}$ é perdido ou recebido fora do intervalo de tolerância pelo criador do bloco ${}_wB_{r+3}$.

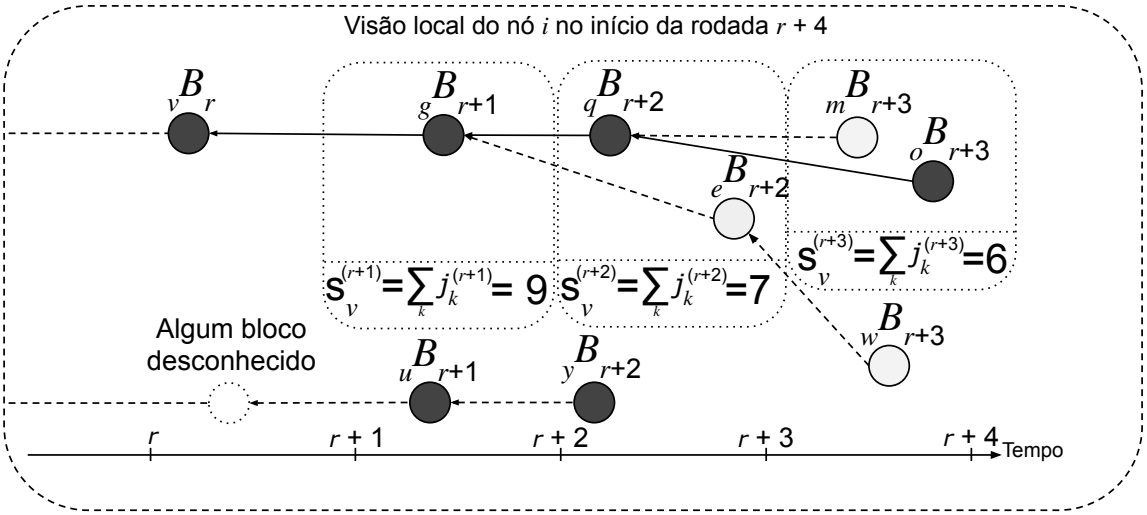

Figura 4.12 – Sorteios bem sucedidos recebidos pelo nó i em um intervalo de rodadas (continuação)

Como o bloco ${}_wB_{r+3}$ não contribui com a mesma visão, o número de sorteios bem sucedidos do nó que o produziu não é adicionado a $s_v^{(r+3)}$. Deste modo, o número médio de sorteios bem sucedidos no intervalo entre a criação do bloco ${}_vB_r$ e a rodada atual é dado por

$$\bar{s}_v^{(r+3)} = \frac{s_v^{(r+1)} + s_v^{(r+2)} + s_v^{(r+3)}}{(r+3) - r} = 7,33.$$

Ao final da rodada $r+3$ o nó i também calcula o número médio de sorteios bem sucedidos que confirmam os blocos ${}_gB_{r+1}$ e ${}_qB_{r+2}$, escolhidos para permanecerem em sua visão local nas rodadas $r+1$ e $r+2$ respectivamente. Estes valores são dados por

$$\bar{s}_g^{(r+3)} = \frac{s_g^{(r+2)} + s_g^{(r+3)}}{(r+3) - (r+1)} = 6,5 \ ; \ \bar{s}_q^{(r+3)} = \frac{s_q^{(r+3)}}{(r+3) - (r+2)} = 6.$$

O CPoS utiliza o número médio de sorteios bem sucedidos definido na Eq. 4.11 e o total esperado de sorteios por rodada (τ) para criar o conceito de confirmação probabilística. É assumido que o próximo bloco a ser confirmado na visão do nó i é o ${}_vB_r$, ou seja, é considerado que todos os blocos anteriores a ${}_vB_r$ já foram confirmados na

blockchain. Além disso, é esperado que $\bar{s}_v^{(x)} \cong \tau$ quando a visão do nó i que contém o bloco ${}_vB_r$ é única e seguida por todos os nós da rede, já que τ é o número esperado de sorteios bem sucedidos em uma rodada.

Neste cenário, se existir outra cadeia formada a partir de um bloco ${}_uB_r$, espera-se que $\bar{s}_u^{(x)} \cong 0$. Entretanto, τ é apenas o valor esperado de um experimento aleatório, ou seja, mesmo nos casos em que o nó i calcule $\bar{s}_v^{(x)} \cong \tau$ a partir de sua visão local, ainda pode existir uma cadeia desconhecida produzida a partir de um bloco ${}_uB_r$ que também seja capaz de produzir $\bar{s}_u^{(x)} \cong \tau$ ao final da rodada x. Os blocos ${}_vB_r$ e ${}_uB_r$ são conflitantes, já que são criados na mesma rodada r e pertencem a *forks* diferentes da blockchain. Neste sentido, o objetivo do mecanismo é garantir probabilisticamente que um deles não seja confirmado, evitando que duas visões conflitantes sejam consideradas válidas.

Para alcançar este objetivo, primeiramente, a probabilidade $P[\bar{s}_u^{(x)} \geq \tau | \bar{s}_v^{(x)} \geq \tau]$ é calculada, ou seja, a probabilidade de existir um bloco desconhecido ${}_uB_r$ com média $\bar{s}_u^{(x)} \geq \tau$ mesmo quando o nó i já conhece um bloco ${}_vB_r$ com $\bar{s}_v^{(x)} \geq \tau$. Logo depois, este resultado é utilizado para definir o conceito de confirmação probabilística de modo genérico. Assim, quando $\Delta_r = 1$, é necessário que ocorram pelo menos 2τ sorteios bem sucedidos na rodada $r+1$, já que esta é a menor quantidade de sucessos capaz de satisfazer a condição $\bar{s}_v^{(r+1)} \geq \tau$ e ainda produzir $\bar{s}_u^{(r+1)} \geq \tau$. Logo, o valor $P[\bar{s}_u^{(r+1)} \geq \tau | \bar{s}_v^{(r+1)} \geq \tau]$ é obtido por meio do cálculo da probabilidade de ocorrerem $k \geq 2\tau$ sorteios bem sucedidos durante a rodada $r+1$. Este valor é dado pelo complementar da probabilidade do número de sorteios bem sucedidos ser no máximo igual a $2\tau - 1$, como mostrado na Eq. 4.12

$$1 - \sum_{k=0}^{2\tau-1} B(k, W, p) \tag{4.12}$$

Quando $\Delta_r = 2$, uma probabilidade equivalente é calculada ($P[\bar{s}_u^{(r+2)} \geq \tau | \bar{s}_v^{(r+2)} \geq \tau]$), porém agora o número de sorteios bem sucedidos é dividido no intervalo de duas rodadas completas ($r+1$ e $r+2$). Qualquer quantidade de sorteios bem sucedidos maior que 4τ é capaz de atender as condições pelas mesmas razões discutidas no caso anterior. Consequentemente, o valor desejado é obtido por meio do cálculo da probabilidade de ocorrerem pelo menos $k = 4\tau$ sucessos divididos em duas rodadas. Como o número de sorteios bem sucedidos pode ser distribuído de diferentes maneiras entre duas rodadas, é necessário primeiramente encontrar o número de soluções para a inequação 4.13 e depois calcular a probabilidade de cada uma delas acontecer. Por fim, o valor desejado é encontrado a partir do complementar da soma destas probabilidades.

$$s_v^{(r+1)} + s_u^{(r+1)} + s_v^{(r+2)} + s_u^{(r+2)} \leq 4\tau - 1 \tag{4.13}$$

A inequação 4.13 é formada pela soma do número de sorteios bem sucedidos

a partir de duas visões diferentes, onde uma inclui o bloco ${}_vB_r$ e a outra o bloco ${}_uB_r$. Qualquer combinação de coeficientes inteiros e não negativos que resulte no máximo em $4\tau - 1$ é uma solução para esta inequação. O número de soluções cresce quando é considerada uma inequação equivalente calculada em um intervalo de rodadas maior e, por este motivo, as probabilidades foram calculadas através do software matemático *Sage math*. As probabilidades obtidas para diferentes valores de τ e intervalos de rodadas (Δ_r) estão apresentadas no gráfico da Fig. 4.13. De acordo com o gráfico, para valores de τ

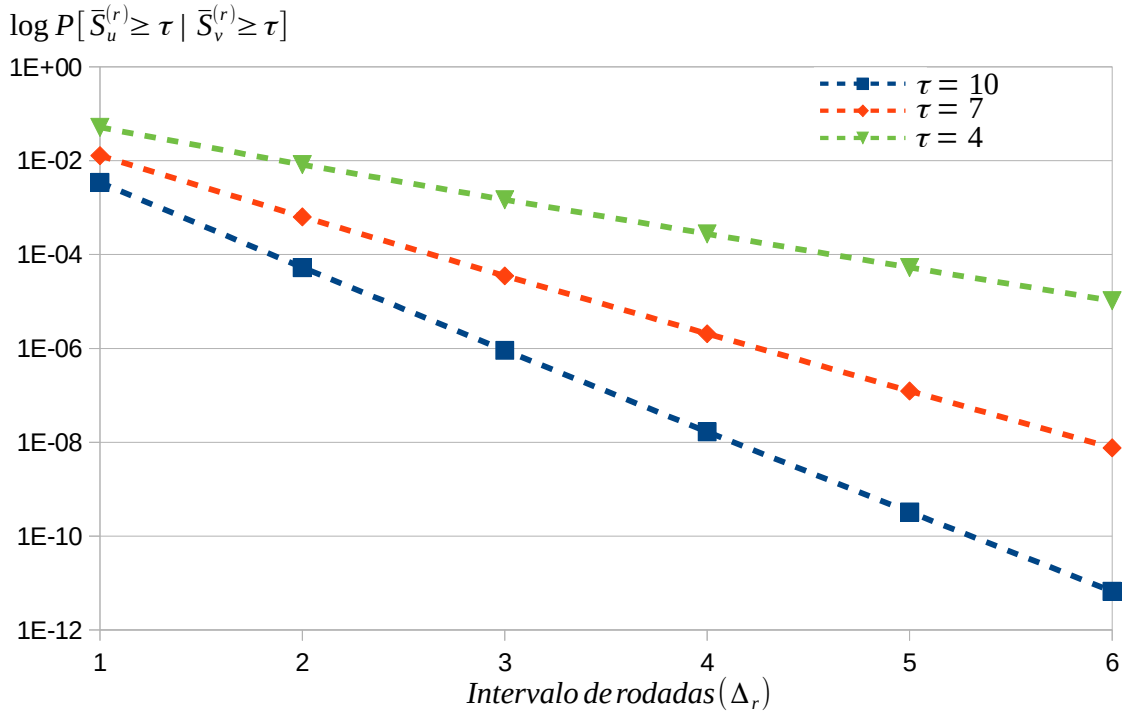

Figura 4.13 – A probabilidade $P[\bar{s}_u^{(x)} \geq \tau | \bar{s}_v^{(x)} \geq \tau]$ diminui com o aumento do intervalo de rodadas Δ_r para todos os valores de τ

mais elevados, a queda da probabilidade $P[\bar{s}_u^{(x)} \geq \tau | \bar{s}_v^{(x)} \geq \tau]$ é mais acentuada, o que está de acordo com a distribuição binomial. Para valores elevados de τ é menos provável obter $\bar{s}_u^{(x)} \geq \tau$ dado $\bar{s}_v^{(x)} \geq \tau$.

Em linhas gerais, a confirmação do bloco ${}_vB_r$ ocorre quando o nó i alcança uma média de sorteios bem sucedidos ($\bar{s}_v^{(x)} \geq \tau$) a partir de sua visão local, que seja capaz de produzir $P[\bar{s}_u^{(x)} \geq \tau | \bar{s}_v^{(x)} \geq \tau] \leq \epsilon$ para um parâmetro de segurança ϵ escolhido. A partir do gráfico pode-se mostrar por meio de um exemplo as ideias principais para a confirmação probabilística, que será formalmente definida mais a frente. Assim, escolhendo $\epsilon = 10^{-6}$ e considerando $\tau = 10$ é possível ver, a partir do gráfico, que o bloco ${}_vB_r$ será confirmado três rodadas após a sua criação ($\Delta_r = 3$) se o nó i calcular em sua visão uma média de sorteios bem sucedidos que atenda no mínimo $\bar{s}_v^{(r+3)} = \tau = 10$, já que $P[\bar{s}_u^{(r+3)} \geq \tau | \bar{s}_v^{(r+3)} \geq \tau] \leq \epsilon = 10^{-6}$.

Na prática, a média $\bar{s}_v^{(x)}$ observada pelo nó i não será sempre τ. Se essa média é

calculada em um intervalo pequeno de rodadas (Δ_r) a partir de um experimento aleatório (sorteio criptográfico), a média pode se afastar do valor esperado. Além disso, é esperado que alguns nós estejam com uma visão diferente daquela seguida pelo restante da rede, contribuindo para a formação de cadeias secundárias, o que reduz o número médio de sorteios bem sucedidos observados na cadeia principal. Por estes motivos, o nó i confirma o bloco ${}_vB_r$ na rodada x se a inequação 4.14 for satisfeita.

$$P[\bar{s}_u^{(x)} \geq s_{min}^{(x)} | \bar{s}_v^{(x)} \geq s_{min}^{(x)}] \leq \epsilon \tag{4.14}$$

Quando o nó i calcula $\bar{s}_v^{(x)} \geq s_{min}^{(x)}$ em alguma rodada x, a inequação 4.14 passa a ser atendida, pois o parâmetro $s_{min}^{(x)}$ é definido como o menor valor inteiro capaz de satisfazê-la na rodada x para um parâmetro de segurança ϵ definido. Em outras palavras, é necessário para a confirmação do bloco ${}_vB_r$ na rodada x, que $\bar{s}_v^{(x)} \geq s_{min}^{(x)}$, pois essa condição faz com que a probabilidade de existir uma outra visão, definida a partir de um bloco ${}_uB_r$ desconhecido pelo nó i, seja menor que ϵ. É necessário também que o bloco antecessor de ${}_vB_r$ na blockchain seja considerado confirmado. Esse fato garante que um conluio de nós desonestos não seja capaz de confirmar uma sequência de blocos de uma cadeia secundária a partir da confirmação de um único sucessor desta cadeia, pois qualquer nó da rede pode atestar se todos os blocos antecessores de qualquer cadeia também foram confirmados. Portanto, as condições para a confirmação probabilística do bloco ${}_vB_r$ pelo nó i ao final da rodada x são definidas abaixo:

- **condição 1:** seu antecessor deve estar confirmado;
- **condição 2:** $\bar{s}_v^{(x)} \geq s_{min}^{(x)}$.

A partir das condições de confirmação, é necessário definir os valores de $s_{min}^{(x)}$ para cada rodada x, considerando o valor de τ e ϵ utilizados. O procedimento adotado para encontrar este valor mínimo consiste em calcular a probabilidade $P[\bar{s}_u^{(x)} \geq s | \bar{s}_v^{(x)} \geq s]$ para vários valores de s inteiros no intervalo Δ_r de interesse. O menor s capaz de satisfazer a inequação 4.14 é considerado o $s_{min}^{(x)}$ da rodada x. Para ilustrar a definição do parâmetro $s_{min}^{(x)}$ a Fig. 4.14 apresenta o gráfico de $P[\bar{s}_u^{(x)} \geq s | \bar{s}_v^{(x)} \geq s]$ em função de Δ_r para $\tau = 10$. De acordo com o gráfico, a probabilidade decresce a medida que a média $\bar{s}_v^{(x)} \geq s$ conhecida pelo nó i aumenta.

Esse resultado é esperado, já que aumentando s torna-se menos provável a existência de uma cadeia capaz de produzir $\bar{s}_u^{(x)} \geq s$. Outro ponto observado é que para qualquer $s \geq 6$, a probabilidade analisada decresce quando o intervalo de rodadas aumenta. Esse resultado, diz que para Δ_r pequeno, é possível que uma cadeia desconhecida possa atingir $\bar{s}_u^{(x)} \geq s$, mas à medida que o nó i mantém a média $\bar{s}_v^{(x)} \geq s$ (para $s > \tau/2$) por mais rodadas, esta possibilidade diminui. Este fato está em concordância com a distribuição de

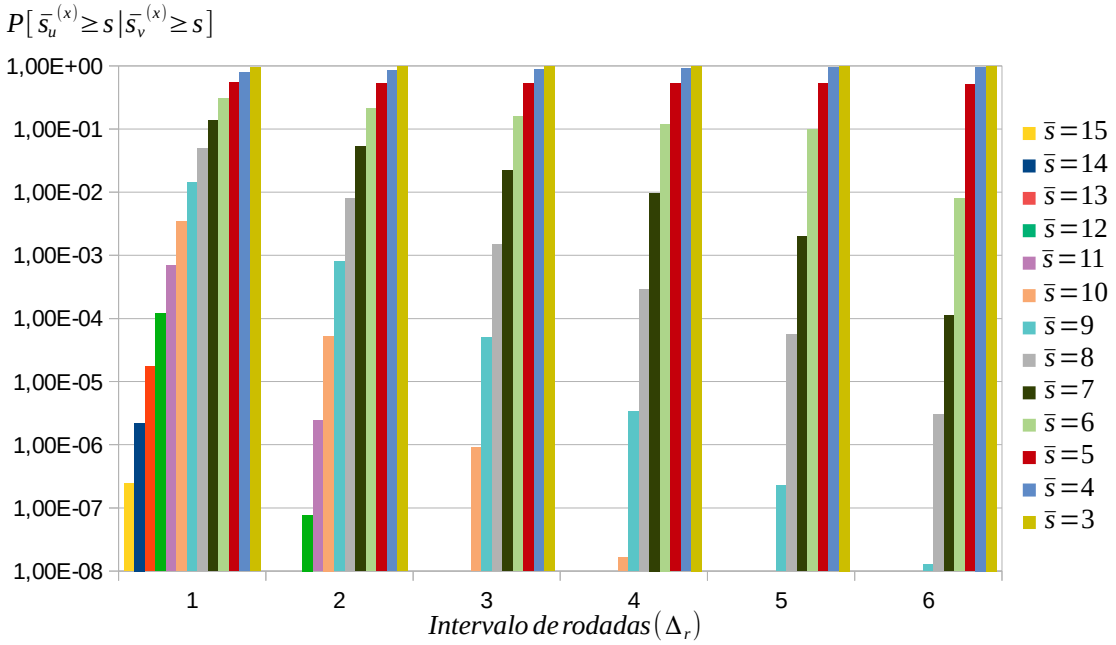

Figura 4.14 – Probabilidade $P[\bar{s}_u^{(x)} \geq s | \bar{s}_v^{(x)} \geq s]$ para s calculada em função de Δ_r e $\tau = 10$

probabilidade do evento aleatório em questão. Quando o intervalo de rodadas aumenta, o número de execuções do experimento também aumenta e com isso espera-se que o número médio de sorteios bem sucedidos se aproxime do valor esperado τ. Portanto, mesmo para valores de s menores que τ, mas suficientemente próximos a ele ($s > \tau/2$), espera-se que a probabilidade atinja valores abaixo do ϵ estabelecido, para um dado intervalo de rodadas mínimo.

Por outro lado, quando o nó i tem uma visão que produz $\bar{s}_v^{(x)} < 5$ a probabilidade de haver outra cadeia capaz de produzir $\bar{s}_u^{(x)} \geq s$ aumenta com o número de rodadas, já que pelos mesmos motivos espera-se que com o aumento do número de rodadas o número médio de sorteios se aproxime de τ. Portanto, os sorteios bem sucedidos que não estão contribuindo para a visão do nó i quando $\bar{s}_v^{(x)} < 5$, são consumidos por alguma visão desconhecida, indicando que a cadeia seguida por ele não está sendo considerada pelos outros nós da rede. Por fim, a partir do gráfico da Fig. 4.14 é possível extrair os valores de $s_{min}^{(x)}$ que produzem a probabilidade capaz de atender a inequação 4.14 para cada intervalo de rodadas ($\Delta_r = x - r$). Estes valores são apresentados na tabela 4.1, considerando $\epsilon = 10^{-6}$. Para $\Delta_r = 7$, 8 e 9 é possível atender a inequação 4.14 quando $s_{min}^{(x)} = 8$, 8 e 7, respectivamente. Sendo assim, para confirmar um bloco, o nó i calcula $\bar{s}_v^{(x)}$ e verifica se o valor encontrado é maior que $s_{min}^{(x)}$, a partir de uma consulta à tabela, tornando baixa a complexidade computacional em tempo de execução.

O gráfico da Fig. 4.15 apresenta o valor de $s_{min}^{(x)}$ para cada intervalo de rodadas Δ_r, considerando também outros valores de τ e ϵ. De acordo com o gráfico, quando ϵ diminui, um valor maior para $s_{min}^{(x)}$ é exigido. Isso ocorre, porque a partir de um ϵ menor, o

Tabela 4.1 – $s_{min}^{(x)}$ capaz de satisfazer a inequação 4.14 para $\tau = 10$ em função de Δ_r com $\epsilon = 10^-6$

Δ_r	$s_{min}^{(x)}$	$P[\bar{s}_u^{(x)} \geq s_{min}^{(x)} \mid \bar{s}_v^{(x)} \geq s_{min}^{(x)}]$
1	15	$2,46 \times 10^{-7}$
2	12	$7,67 \times 10^{-8}$
3	10	$9,11 \times 10^{-7}$
4	10	$1,66 \times 10^{-8}$
5	9	$2,26 \times 10^{-7}$
6	9	$1,28 \times 10^{-8}$

valor de $s_{min}^{(x)}$ capaz de satisfazer a inequação 4.14 torna-se maior. Assim, é correto afirmar que a segurança do mecanismo aumenta para valores de ϵ menores, já que a probabilidade de uma outra visão desconhecida confirmar um bloco conflitante diminui; entretanto, o tempo para que o nó alcance a confirmação do bloco é maior, pois em geral são necessárias mais rodadas para que a condição $\bar{s}_v^{(x)} \geq s_{min}^{(x)}$ seja obtida e, com isso, a confirmação seja alcançada.

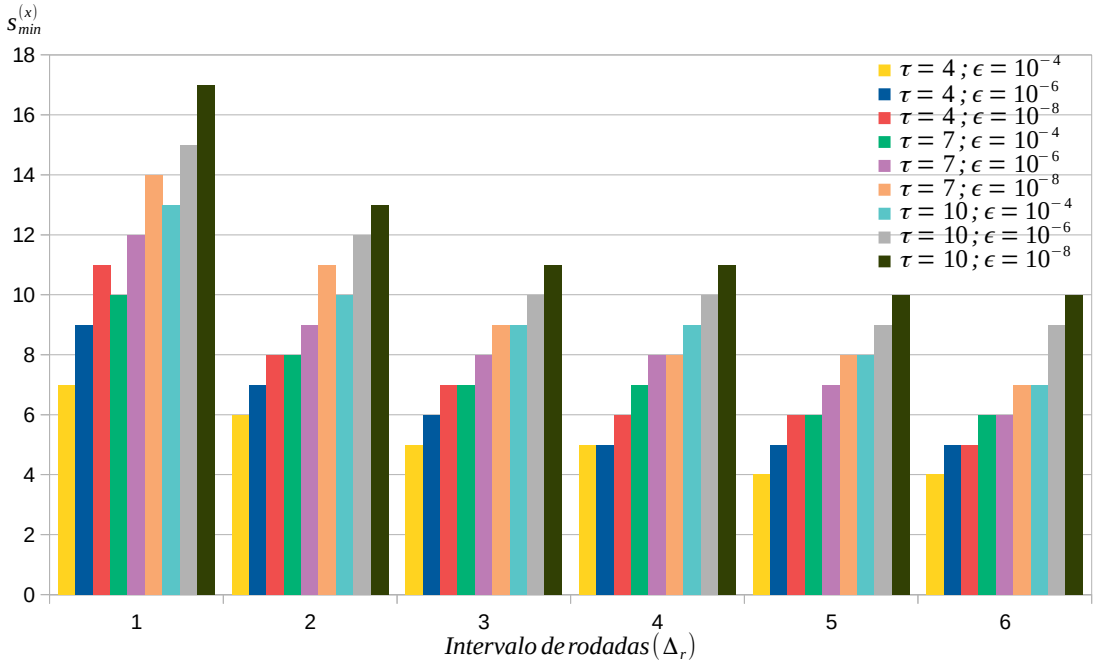

Figura 4.15 – $s_{min}^{(x)}$ necessário para confirmação de um bloco em função dos parâmetros τ, Δ_r e ϵ

Outro ponto importante está associado ao parâmetro τ escolhido. Quando $\tau = 10$, sabe-se que $s_{min}^{(x)} = 9$ para $\Delta_r = 6$ e $\epsilon = 10^{-6}$. Por outro lado, $s_{min}^{(x)} = 5$ para $\tau = 4$, mantendo Δ_r e ϵ iguais. Neste exemplo, percebe-se que é exigido $s_{min}^{(x)} > \tau$ quando $\tau = 4$, o que não ocorre para $\tau = 10$. Comparando os dois casos, como o valor médio esperado é sempre τ, podemos afirmar que é menos provável obter uma média $\bar{s}_v^{(x)}$ capaz de atender

$\bar{s}_v^{(x)} \geq s_{min}^{(x)} = 5$ e com isso confirmar o bloco ${}_vB_r$ quando $\tau = 4$, já que $s_{min}^{(x)}$ neste caso ainda é maior que τ. Por outro lado, para $\tau = 10$ é necessário obter uma média $\bar{s}_v^{(x)}$ que atenda apenas $\bar{s}_v^{(x)} \geq s_{min}^{(x)} = 9$, ou seja, já abaixo do valor esperado ($\tau = 10$). Em linhas gerais, é esperado que o CPoS tenha um menor tempo de confirmação de blocos quando o parâmetro τ é aumentado, já que neste caso o valor de $s_{min}^{(x)}$ necessário para a confirmação dos blocos cai mais rapidamente em função do aumento do intervalo de rodadas, ficando inclusive abaixo de τ em algumas situações.

Nesta linha, é definido o conceito de margem de confirmação de uma rodada x ($M^{(x)}$), que pode ser calculada a partir da Eq. 4.15.

$$M^{(x)} = \tau - s_{min}^{(x)} \tag{4.15}$$

De acordo com a Eq. 4.15, a margem de confirmação de uma rodada x depende do parâmetro τ e do valor mínimo $s_{min}^{(x)}$, o qual a média calculada $\bar{s}_v^{(x)}$ deve atingir ao final da rodada x para que a confirmação do bloco ${}_vB_r$ ocorra. Desta forma, quanto mais rapidamente a margem de confirmação crescer, menor é o número de rodadas esperadas para que as confirmações dos blocos ocorram, já que torna-se mais fácil alcançar o valor mínimo necessário ($s_{min}^{(x)}$). As margens de confirmação para os diferentes valores de τ analisados são mostradas no gráfico da Fig. 4.16 em função de Δ_r, τ e ϵ.

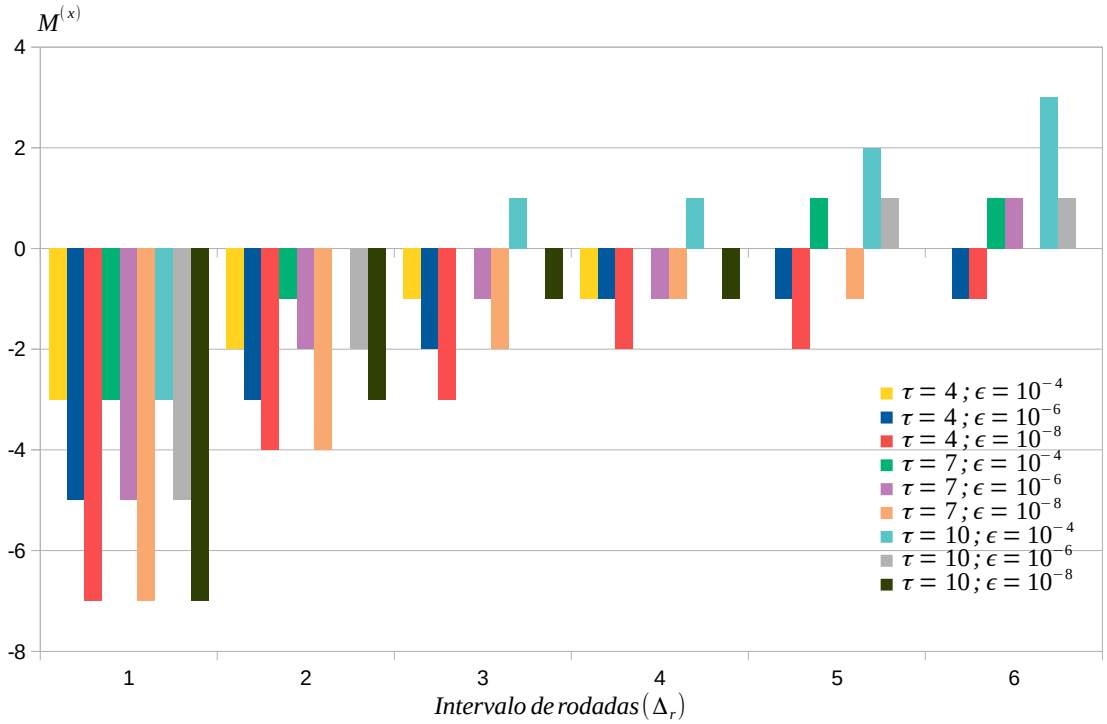

Figura 4.16 – Margem de confirmação melhora mais rapidamente a partir do aumento de τ

No capítulo 6, é mostrado como a margem de confirmação associada a cada valor de τ é capaz de influenciar o tempo de confirmação dos blocos e a robustez do

consenso na presença de conluios formados por nós desonestos. Já no Anexo E, é feito um esboço preliminar sobre uma prova de que as condições de confirmação estabelecidas nessa seção são suficientes para obter um consenso seguro para blockchains públicas, no que tange à sua capacidade de evitar que confirmações conflitantes ocorram.

4.6 Conclusões

Este capítulo apresentou o CPoS como um novo mecanismo de consenso que é capaz de atingir o acordo de maneira probabilística e sem a utilização de comitês de validação. Foi definido que a cada rodada novos blocos são produzidos a partir de nós sorteados e que o número médio de sorteios bem sucedidos por rodada é dado pelo parâmetro τ.

A partir do *hash* de prova, foi definido um critério para aceitação de blocos, que é utilizado para decidir qual bloco deve permanecer na visão local de um nó, quando este recebe mais de um bloco válido com uma mesma rodada de criação. Como o critério diz para manter o bloco com o menor *hash* de prova entre aqueles com uma mesma rodada de criação, é possível evitar o prolongamento de *forks* no mecanismo, já que as decisões são tomadas ainda dentro da rodada corrente.

Um novo mecanismo de confirmação de blocos foi apresentado, onde os nós podem acompanhar a evolução de suas visões da blockchain por meio dos blocos recebidos a cada rodada. A confirmação ocorre de maneira probabilística e a segurança do processo pode ser ajustada por meio da definição do parâmetro de segurança ϵ. Neste modelo, é mostrado que o número de rodadas necessárias até que um bloco seja confirmado diminui a partir da elevação do valor do parâmetro τ.

No próximo capítulo, é apresentada a latência de confirmação esperada para o CPoS, uma caracterização dos principais ataques mapeados e um método de sincronização que deve ser utilizado pelos nós que constatam que estão seguindo visão(ões) diferente(s) daquela que é considerada válida pela maioria dos participantes.

5 Análise teórica do mecanismo de consenso proposto

De acordo com os conceitos referentes à confirmação probabilística apresentados no capítulo anterior, é possível afirmar que o CPoS, além de dispensar a exigência de um comitê de confirmação, é um mecanismo que possui parâmetros de configuração que permitem influenciar o tempo e a segurança da confirmação dos blocos. Esse é um outro aspecto que diferencia o CPoS de outros mecanismos baseados em Proof-of-Stake. Nesse contexto, é importante conhecer qual o tempo gasto pelo mecanismo para que novos blocos sejam confirmados a partir da modificação de seus principais parâmetros. Além disso, é preciso fazer uma análise teórica sobre a segurança do mecanismo na presença de adversários que buscam realizar confirmações conflitantes por meio da criação de um conluio que emprega seus *stakes* em uma visão diferente daquela que os nós honestos trabalham.

Este capítulo define a latência de confirmação esperada para o mecanismo, a partir do cálculo da probabilidade de um bloco ser confirmado exatamente em uma rodada x. Além disso, como o mecanismo opera em uma infraestrutura sujeita a falhas Bizantinas, é possível que alguns nós sigam cadeias secundárias que não são consideradas pela grande maioria dos nós. Sendo assim, este capítulo também apresenta um modelo para que os nós avaliem as suas visões e sejam capazes de buscar uma nova sincronização com a visão correta. Por fim, é apresentada uma análise sobre a influência de um conluio de nós desonestos no consenso obtido pelo mecanismo quando eles controlam uma fração f do *stake* total depositado no período atual.

5.1 Latência de confirmação

Segundo Nguyen et al. (NGUYEN et al., 2019) a confirmação de uma transação depende de dois fatores principais, chamados *throughput* de transações e o tempo para confirmação de um bloco. O *throughput* é o número de transações por segundo ($trans/s$) que a rede pode processar, ou seja, ele determina o quão rápido uma transação é adicionada em um bloco da blockchain. Por outro lado, o tempo de confirmação de um bloco depende do tempo de convergência do consenso para que o bloco em questão seja definitivamente aceito na visão dos nós honestos. Por exemplo, no Bitcoin um bloco aguarda seis confirmações e, portanto, como em média cada bloco é produzido a cada dez minutos, o tempo médio para que uma confirmação ocorra é de uma hora.

Sendo assim, é definida no CPoS a latência de confirmação R_c, como o número

médio de rodadas necessárias para que um bloco produzido em uma rodada r seja confirmado. Essa seção apresenta o valor teórico esperado para a latência R_c em função da variação dos principais parâmetros do mecanismo.

No CPoS um bloco ${}_vB_r$ é confirmado pelo nó i em uma rodada x sempre que seu antecessor já estiver confirmado e o nó calcular uma média $\bar{s}_v^{(x)}$ capaz de atender $\bar{s}_v^{(x)} \geq s_{min}^{(x)}$, onde $s_{min}^{(x)}$ é o menor valor inteiro capaz de atender a Eq. 4.14. Como mostrado na seção 4.5, o valor $s_{min}^{(x)}$ depende do intervalo de rodadas Δ_r ($\Delta_r = x - r$), do número esperado de sorteios bem sucedidos τ e do parâmetro de segurança ϵ. A partir destes parâmetros, é possível determinar a latência de confirmação esperada R_c, encontrando o número esperado de rodadas entre duas confirmações sucessivas. A forma como isso é feito segue as premissas acima e o fato de que um bloco ${}_vB_r$ é confirmado ao final da primeira rodada x em que o nó obtém $\bar{s}_v^{(x)} \geq s_{min}^{(x)}$ para este bloco. Assim, é preciso determinar a probabilidade q_x de se alcançar o valor $s_{min}^{(x)}$ necessário para que a confirmação ocorra ao final da rodada x e multiplicá-la pela probabilidade de o bloco não ter sido confirmado em rodadas anteriores a x, desde a rodada seguinte àquela em que o bloco foi criado (r). Diz-se que o resultado desta multiplicação (Eq. 5.1) fornece a probabilidade Q_x de o bloco ser confirmado ao final na rodada x.

$$Q_x = q_x \prod_{t=r+1}^{x-1} (1 - q_t) \tag{5.1}$$

O termo $1 - q_t$ é o complementar da probabilidade q_t e, portanto, seu valor representa a possibilidade do bloco não ser confirmado na rodada t. Assim, o produto de todos estes termos no intervalo $r + 1 \leq t \leq x - 1$, fornece a probabilidade do bloco criado na rodada r não ser confirmado em nenhuma rodada menor que x. Ao multiplicar este valor por q_x tem-se a probabilidade Q_x de confirmação do bloco exatamente na rodada x, como apresentado na Eq. 5.1.

Portanto, primeiramente é necessário calcular a probabilidade q_x do nó obter uma média $\bar{s}_v^{(x)}$ ao final da rodada x que atenda $\bar{s}_v^{(x)} \geq s_{min}^{(x)}$. Para $\Delta_r = 1$, é necessário calcular o valor q_{r+1}, ou seja, no final da rodada $r + 1$ é encontrada a probabilidade de se obter $\bar{s}_v^{(r+1)} \geq s_{min}^{(r+1)}$ a partir dos sorteios bem sucedidos recebidos durante a rodada $r + 1$ que confirmem o bloco ${}_vB_r$. Portanto, o valor q_{r+1} é igual à probabilidade do nó receber pelo menos $s_{min}^{(r+1)}$ sorteios bem sucedidos durante a rodada $r+1$. Este valor é dado pelo complementar da probabilidade do nó receber até $s_{min}^{(r+1)} - 1$ sorteios bem sucedidos durante a rodada $r + 1$, como mostrado na Eq. 5.2.

$$q_{r+1} = 1 - \sum_{k=0}^{s_{min}^{(r+1)}-1} B(k, W, p) \tag{5.2}$$

onde $B(k, W, p)$ é calculado de acordo com a Eq. 4.4 definida na Seção 4.2.

Já no final da rodada $r+2$ é necessário calcular a probabilidade de ocorrer $\bar{s}_v^{(r+2)} \geq s_{min}^{(r+2)}$, ou seja, é preciso determinar a probabilidade do nó receber uma quantidade de sorteios bem sucedidos em duas rodadas inteiras ($\Delta_r = 2$) que seja no mínimo igual a $2s_{min}^{(r+2)}$. Essa é a menor quantidade de sorteios bem sucedidos capaz de produzir uma média igual a $s_{min}^{(r+2)}$ ao final da rodada $r+2$. O cálculo da probabilidade q_{r+2} é feito a partir de todas as combinações $(s_v^{(r+1)}, s_v^{(r+2)})$ que são soluções para a inequação 5.3.

$$s_v^{(r+1)} + s_v^{(r+2)} \leq 2s_{min}^{(r+2)} - 1 \tag{5.3}$$

Seja Y o conjunto formado por todos os pares $(s_v^{(r+1)}, s_v^{(r+2)})$ que são soluções para Eq. 5.3. Sendo assim, é calculada a soma das probabilidades de cada um desses pares ocorrerem, onde o complementar dessa soma é q_{r+2}, como mostrado na Eq. 5.4.

$$q_{r+2} = 1 - \sum_{Y} B(s_v^{(r+1)}, W, p) \times B(s_v^{(r+2)}, W, p) \tag{5.4}$$

A Eq. 5.5 é capaz de fornecer a probabilidade q_x e, portanto, é a generalização da Eq. 5.4 para uma rodada x qualquer, onde Y agora é redefinido como sendo o conjunto de todas as tuplas do tipo $(s_v(r+1), s_v(r+2), ..., s_v(x))$ que são soluções para a inequação 5.6, uma generalização da inequação 5.3.

$$q_x = 1 - \sum_{Y} B(s_v^{(r+1)}, W, p) \times B(s_v^{(r+2)}, W, p) \times ... \times B(s_v^{(x)}, W, p) \tag{5.5}$$

$$s_v^{(r+1)} + s_v^{(r+2)} + ... + s_v^{(x)} \leq \Delta_r \times s_{min}^{(x)} - 1 \tag{5.6}$$

A probabilidade do nó obter a média necessária para confirmar o bloco ao final de uma rodada x (q_x) foi calculada para vários valores de Δ_r. Assim, a partir do cálculo da probabilidade q_x e da Eq. 5.1 é possível encontrar a probabilidade Q_x do bloco ser confirmado exatamente ao final da rodada x. O gráfico da Fig. 5.1 apresenta a probabilidade Q_x em função de Δ_r, τ e ϵ. De acordo com o gráfico, a probabilidade Q_x confirma a hipótese da seção anterior que diz que um bloco é confirmado mais rapidamente para maiores valores de τ. Além disso, a confirmação de um bloco depende do parâmetro de segurança ϵ, pois variando este limiar, os valores de Q_x são alterados em relação ao intervalo Δ_r, mostrando que a latência de confirmação de um bloco está de fato associada com este parâmetro.

A probabilidade acumulada da variável Q_x, que neste caso representa a probabilidade do bloco ser confirmado até o final da rodada x, é apresentada na Fig. 5.2. Esta probabilidade acumulada permite realizar uma melhor análise sobre quando um bloco é confirmado a partir dos parâmetros τ e ϵ definidos para toda a rede. Ela confirma que, quando τ aumenta, os blocos passam a ser confirmados dentro de um intervalo de rodadas menor, pois a probabilidade acumulada converge mais rapidamente para um.

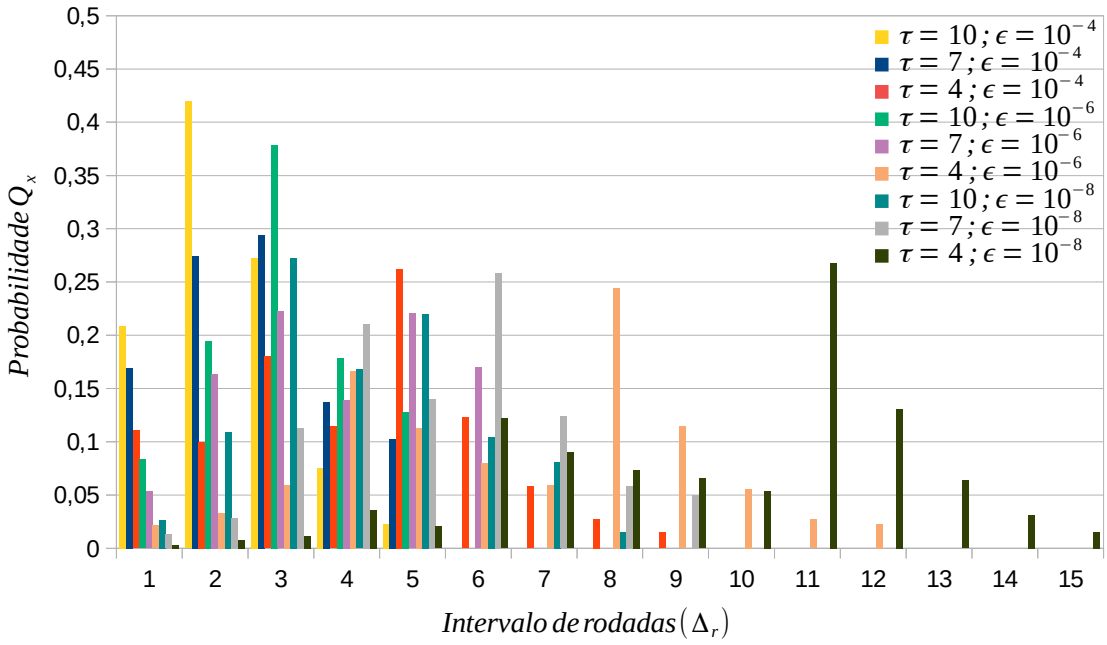

Figura 5.1 – Probabilidade Q_x de um bloco ser confirmado em função do número de rodadas para diferentes valores de ϵ e τ

Como o CPoS é um protocolo de confirmação probabilística que não utiliza comitês de confirmação, é importante que um participante conheça a latência de confirmação esperada R_c. Este conhecimento, permite que um nó avalie localmente se sua visão da blockchain é a mesma dos outros nós da rede, comparando o número de rodadas entre confirmações com o valor teórico esperado. Assim, quando um nó descobre que sua visão local não é seguida pela rede, ele inicia um processo de ressincronização, onde, por meio de consultas a seus *peers*, ele pode obter blocos que porventura foram perdidos em rodadas passadas. Esses blocos podem formar uma nova cadeia que de fato esteja sendo seguida pelo restante da rede e, neste caso, deve substituir a cadeia original do nó até o último bloco comum entre as duas visões. Este processo é chamado de ressincronização de cadeia e seus detalhes são discutidos na próxima seção.

Por fim, é possível calcular a latência de confirmação esperada R_c, utilizando a Eq. 5.7.

$$R_c = \sum_{x=r+1}^{\infty} (x - r) \times Q_x \tag{5.7}$$

A Eq. 5.7 diz que, teoricamente, é necessário realizar uma soma infinita para obter R_c, pois de fato sempre existe uma probabilidade do bloco ${}_vB_r$ ser confirmado em qualquer rodada $x > r$, onde r é a rodada de criação do bloco. Todavia, de acordo com os resultados apresentados no gráfico da Fig. 5.2, é possível notar que a probabilidade acumulada Q_x se aproxima de um em um intervalo finito de rodadas, que depende dos parâmetros τ e ϵ. Sendo assim, para efeitos práticos, deve-se realizar o somatório da Eq. 5.7 considerando um intervalo de rodadas cuja probabilidade acumulada de Q_x esteja suficientemente próxima

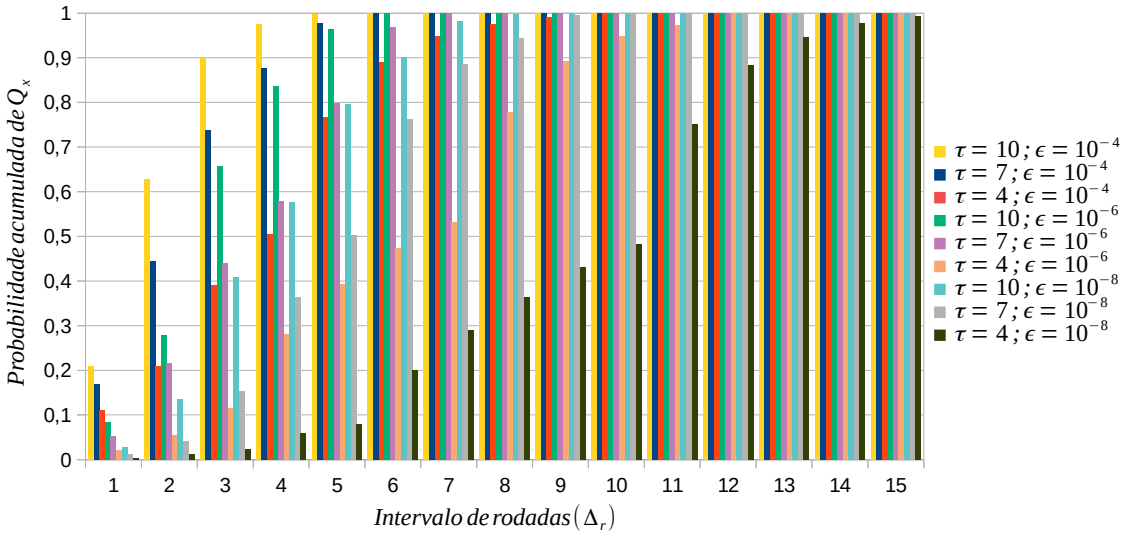

Figura 5.2 – Probabilidade acumulada de Q_x referente à confirmação de um bloco em função do número de rodadas para diferentes valores de ϵ e τ

de um. Neste sentido, o gráfico da Fig. 5.3 apresenta a latência de confirmação R_c em função dos parâmetros τ e ϵ. Como esperado, os valores de R_c diminuem para valores

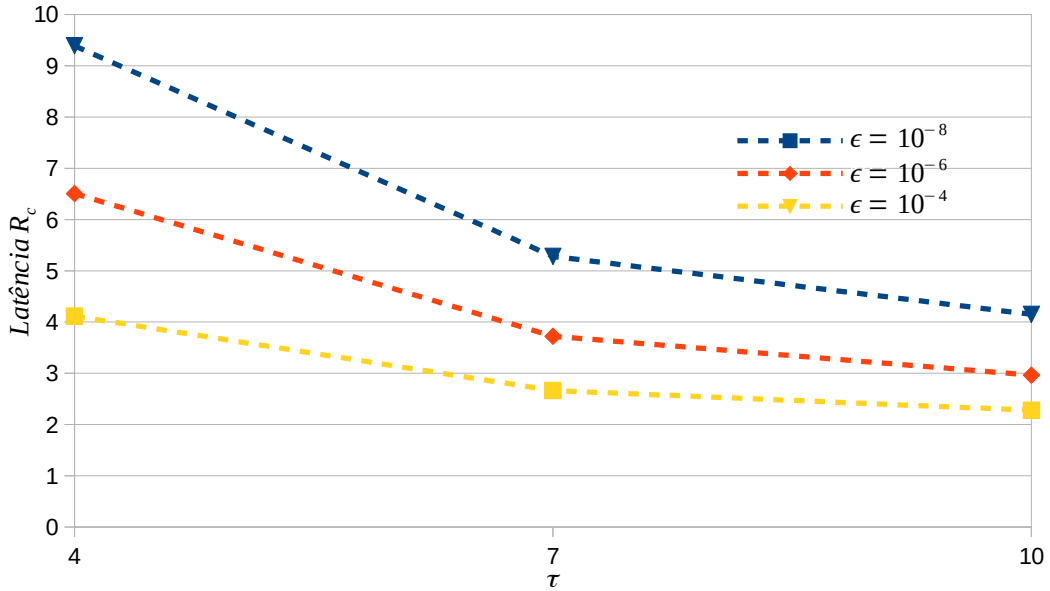

Figura 5.3 – Latência de confirmação R_c diminui para valores de τ mais elevados

de ϵ mais elevados, já que nestes casos, o mecanismo é capaz de alcançar a confirmação mais rapidamente, porém com um consenso menos seguro contra possíveis reversões. A partir do aumento de τ, também é possível reduzir a latência R_c, porém sem alterar o nível de segurança estabelecido, o que é desejável para aplicações que almejam um alto desempenho no que tange o tempo de confirmação de blocos, mas que também precisam manter o consenso em um patamar de segurança bastante elevado.

No capítulo 6, é feita uma comparação entre os resultados teóricos apresentados neste gráfico e aqueles encontrados em implementações práticas do CPoS. Já no capítulo 7, o CPoS é comparado com mecanismos de consenso baseados em comitês de confirmação e também com o *Proof-of-Work* utilizado na criptomoeda Bitcoin.

5.2 Detecção de visão incorreta da cadeia e ressincronização

Um nó i pode seguir uma visão da blockchain não considerada pelo restante da rede, à medida que alguns blocos produzidos não sejam recebidos por ele. Vários fatores contribuem para que isso aconteça e estão normalmente associados ao aumento do atraso de transmissão da rede, presença de nós desonestos ou a falhas nos nós. Neste sentido, o CPoS dispõe de um mecanismo para permitir que um nó faça a ressincronização com a cadeia correta, sempre que identificar a possibilidade de não estar mais sincronizado com a visão correta (ou predominante) da blockchain. Este processo é realizado pelo nó i por meio de consultas aos *peers*, para primeiramente confirmar se de fato está fora de sincronismo e depois, caso realmente esteja, para solicitar os blocos faltantes.

É importante ressaltar que podem ocorrer falsos positivos, ou seja, um nó decidir que uma ressincronização é necessária quando na verdade ele está seguindo a cadeia principal. Isso ocorre porque no processo de sincronismo que será definido nessa seção, o CPoS determina que um nó pode estar sem sincronização sempre que a média de sorteios bem sucedidos calculada a partir de um bloco ${}_vB_r$ é menor que um valor mínimo, definido em função do intervalo de rodadas Δ_r e do número esperado de sorteios bem sucedidos τ. Neste sentido, devido às características probabilísticas do CPoS no que tange a ocorrência de sorteios bem sucedidos, é possível que o protocolo permaneça durante algumas rodadas com valores abaixo do esperado (τ), contribuindo para que em determinados casos a média $\bar{s}_v^{(x)}$ diminua e um nó dispare uma ação de ressincronização mesmo quando já está com a visão correta. Esta situação não é frequente, já que o valor mínimo necessário para não ressincronizar é atingido com alta probabilidade quando a visão do nó é a mesma do restante da rede. Todavia, quando um falso positivo ocorre, o nó é capaz de perceber que já está com a mesma visão de seus *peers* e, desta forma, continua na cadeia atual, na espera de uma melhora gradual da média em rodadas futuras, como será mostrado mais adiante.

No início de cada rodada, o nó avalia a visão de sua blockchain antes de participar do sorteio criptográfico. Nesta avaliação, são confirmados os blocos que já estão com média superior ao mínimo necessário ($s_{min}^{(x)}$), e é verificado se é preciso iniciar o processo de ressincronização. Diz-se que o limiar $\alpha^{(x)}$ determina se uma ressincronização deve ocorrer na rodada x. Esse limiar é definido como um valor inteiro que a média $\bar{s}_v^{(x)}$ deve superar com alta probabilidade, se o nó estiver na cadeia correta, como será detalhado

mais adiante. Sendo assim, $\alpha^{(x)}$ é o menor valor capaz de garantir $P[\bar{s}_v^{(x)} \geq \alpha^{(x)}] \geq 0,95$ para cada intervalo Δ_r, quando o nó está na cadeia correta. Por outro lado, é esperado que o limiar $\alpha^{(x)}$ não seja superado pela média $\bar{s}_v^{(x)}$, quando o nó está em uma cadeia diferente daquele seguida pelo restante da rede, pois espera-se que o número de sorteios bem sucedidos de qualquer cadeia secundária não seja suficiente para sustentar a relação $\bar{s}_v^{(x)} \geq \alpha^{(x)}$ por muitas rodadas. O processo de ressincronização é iniciado pelo nó na primeira rodada em que o limiar $\alpha^{(x)}$ não seja superado.

O gráfico da Fig. 5.4, apresenta $\alpha^{(x)}$ como o menor valor inteiro que atende $P[\bar{s}_v^{(x)} \geq \alpha^{(x)}] \geq 0,95$ em função do intervalo de rodadas Δ_r e do número esperado de sorteios bem sucedidos τ. Nesta condição, probabilidade da média $\bar{s}_v^{(x)}$ superar $\alpha^{(x)}$ é no mínimo $0,95$. O valor $0,95$ foi escolhido de maneira a limitar a probabilidade de falsos positivos abaixo $0,05$, que consideramos ser um valor aceitável na prática.

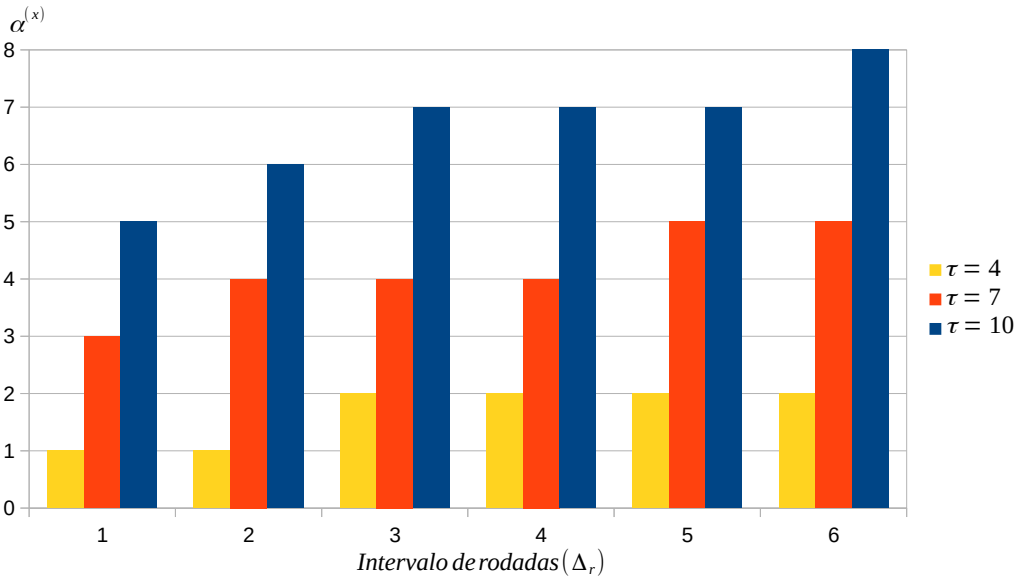

Figura 5.4 – Limiar para ressincronização $\alpha^{(x)}$ aumenta em função do número de rodadas Δ_r

De acordo com o gráfico e também pelo fato de ser esperado que a média $\bar{s}_v^{(x)}$ se aproxime mais de τ com o aumento de Δ_r, pode-se dizer que o limiar $\alpha^{(x)}$ não diminui com o aumento do intervalo de rodadas Δ_r. Isso ocorre porque a partir da convergência de $\bar{s}_v^{(x)}$ em torno de τ, é possível aproximar ainda mais $\alpha^{(x)}$ de τ e mesmo assim garantir que $P[\bar{s}_v^{(x)} \geq \alpha^{(x)}] \geq 0,95$. Portanto, é possível afirmar que a probabilidade de um nó detectar que não está na cadeia correta aumenta com o número de rodadas Δ_r, já que torna-se menos provável alcançar o limiar $\alpha^{(x)}$ a cada rodada, como mostrado no gráfico da Fig. 5.5. A probabilidade de detecção apresentada no gráfico, é calculada a partir da existência de uma cadeia principal seguida pela rede. Esta cadeia, tem a visão confirmada a partir da chegada de τ sorteios bem sucedidos, em média, a cada rodada. Logo, para que um nó i não conclua que está seguindo uma visão diferente, a rede deve produzir no

mínimo $\tau + \alpha^{(x)}$ sorteios bem sucedidos a cada nova rodada x, pois, neste caso, τ sorteios são destinados à cadeia principal e $\alpha^{(x)}$ são da cadeia seguida pelo nó i. Se este número mínimo de sorteios se repete a cada rodada do intervalo Δ_r, o nó i não detecta que está seguindo outra cadeia, já que a média da cadeia secundária é mantida em pelo menos $\alpha^{(x)}$.

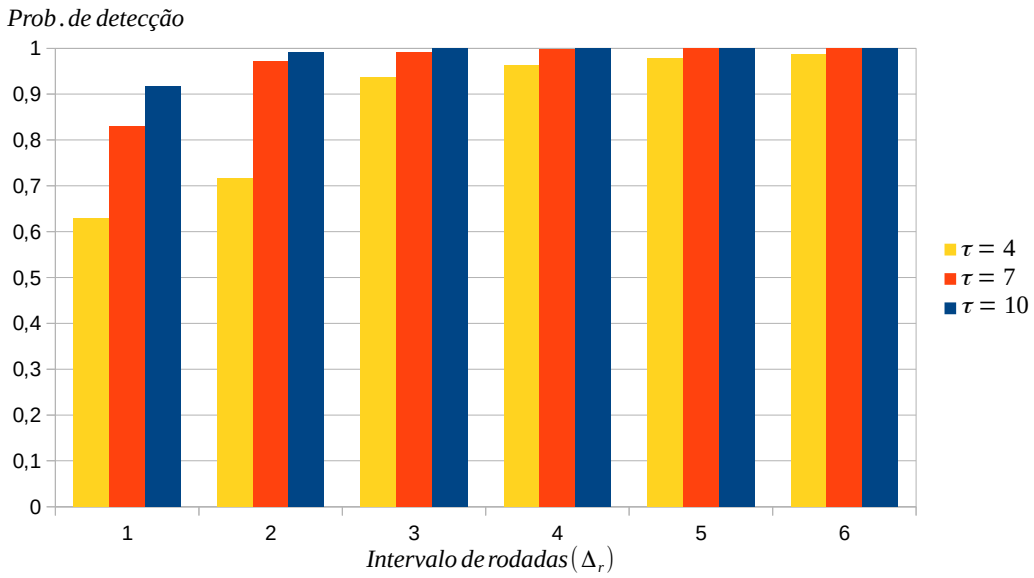

Figura 5.5 – Probabilidade de um nó detectar que está com uma visão errada da blockchain

O gráfico mostra que a probabilidade de detecção é elevada já a partir de $\Delta_r = 1$ para todos os valores de τ. Essa é uma característica desejável, já que por meio de uma detecção rápida, os nós permanecem pouco tempo com uma possível visão errada em relação à blockchain, o que contribui para o aumento do desempenho. Além disso, ao aumentar o valor de τ, é esperado que a detecção ocorra mais rapidamente. Por fim, a seção A.3 do Anexo A apresenta um diagrama de sequência que une o processo de confirmação probabilística com os possíveis pedidos de ressincronização.

5.3 Análise de segurança

Segundo Popov et al. (POPOV; BUCHANAN, 2021), é importante considerar que os nós adversários sejam controlados por um único nó ou organização, aumentando as chances de realizar um ataque bem sucedido contra o mecanismo de consenso. Estes grupos de atacantes somam seus esforços na direção de um objetivo comum, explorando alguma característica do mecanismo. Em sistemas de consenso distribuídos, é comum assumir que a operação dos mecanismos é garantida sempre que os recursos associados aos grupos desonestos estiverem abaixo de um limiar máximo, que varia de acordo com as características de cada mecanismo. A Tabela 3.2 mostra que para o PoW este limiar é determinado pela quantidade máxima de poder computacional que pode estar em posse do atacante, ou seja, é necessário que a maioria simples (51%) dos recursos estejam com os

nós honestos, para que os atacantes não tenham a maior parte do poder computacional à sua disposição. Já os protocolos que realizam o consenso em infraestruturas Bizantinas por meio de esquemas de votação, são em geral tolerantes a grupos de atacantes que detêm menos de 1/3 dos recursos (CASTRO; LISKOV, 2002).

No CPoS, os adversários são primeiramente caracterizados pelo tipo de ação que podem realizar no sistema, considerando particularidades do mecanismo. Logo depois, são definidos os limiares suportados para cada tipo de ação tomada pelos atacantes. Desta forma, são mapeados dois tipos de ataques ao CPoS:

- formação de conluios com o objetivo de confirmar um bloco conflitante;
- divulgação seletiva dos blocos produzidos ou recebidos.

A possibilidade dos atacantes confirmarem um bloco conflitante está diretamente associada à quantidade de *stake* controlado por eles dentro do período atual. Assim, enquanto a maioria do stake disponível em um período qualquer estiver sob o controle de nós honestos, é pouco provável que atacantes com a minoria do stake tenham sucesso em um ataque, já que é esperado que o número de sorteios bem sucedidos dos mesmos seja inferior aos dos nós honestos. Os limites desta ação são estabelecidos a partir da análise da probabilidade do ataque ser bem sucedido quando os adversários têm a disposição uma fração fW do *stake* total disponível, e os nós honestos contam com $(1-f)W$, onde $0 \leq f \leq 1$. Neste sentido, os atacantes conseguem atingir o objetivo quando o bloco conflitante ${}_uB_r$ atinge uma média $\bar{s}_u^{(x)}$ que atenda $\bar{s}_u^{(x)} \geq s_{min}^{(x)}$ ao final da rodada x, enquanto os nós honestos ainda não confirmaram o bloco de mesmo índice em qualquer rodada $t < x$, ou seja, o bloco que ocupa a mesma posição (altura) na cadeia que o bloco ${}_uB_r$.

Assim, a medida que Δ_r aumenta, é esperado que a probabilidade do conluio confirmar um bloco conflitante diminua, já que aumentam as chances dos nós honestos já terem confirmado um outro bloco de mesma posição em uma rodada menor. Portanto, quando $\Delta_r = 2$, a probabilidade dos adversários atingirem a média mínima de confirmação deve ser multiplicada por $1 - Q_1$, ou seja, pela probabilidade dos nós honestos não terem confirmado o bloco ao final da rodada anterior. Neste caso, como há presença de conluio, a probabilidade Q_1 deve ser calculada considerando que os nós honestos controlam apenas $(1-f)W$ do stake total, o que resulta em valores menores que os apresentados na seção 5.1. De maneira semelhante, quando $\Delta_r = 3$ a probabilidade do conluio confirmar um bloco conflitante é multiplicada pela probabilidade $(1 - Q_1)(1 - Q_2)$, indicando que eles terão sucesso apenas se os nós honestos não confirmarem os blocos nas duas rodadas anteriores. Esse processo segue a cada nova rodada, mostrando claramente que a probabilidade de sucesso do conluio é diminuída a cada rodada em que ele não atinge o objetivo.

A probabilidade dos adversários confirmarem um bloco conflitante na rodada x é apresentada no gráfico da Fig. 5.6, assumindo que os atacantes devam alcançar a média mínima $s_{min}^{(x)}$ dispondo apenas de uma fração (fW) do *stake* total. O gráfico da

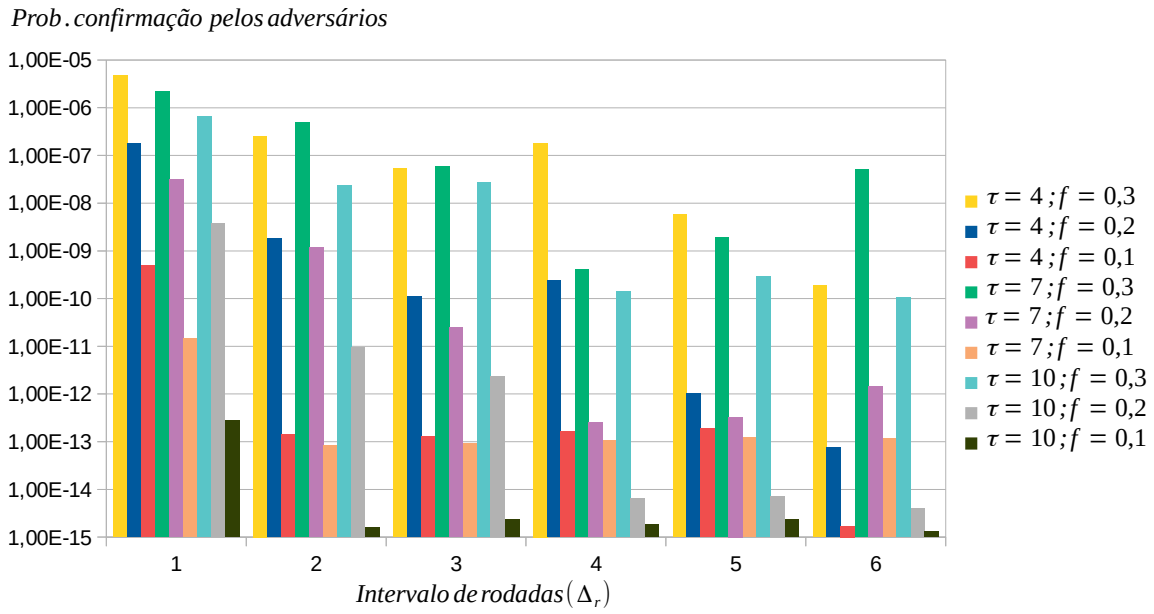

Figura 5.6 – Probabilidade dos adversários confirmarem um bloco conflitante em função de Δ_r, τ e f para $\epsilon = 10^{-6}$

Fig. 5.6 avalia a probabilidade dos adversários confirmarem um bloco conflitante para $\epsilon = 10^{-6}$. Como esperado, o sucesso dos adversários depende da quantidade de *stake* que eles controlam. Assim, para $f \leq 0,3$ o gráfico mostra que a probabilidade dos adversários confirmarem um bloco conflitante é baixa para todos os valores de τ, já que na maioria dos casos ela é menor que o limiar de confirmação estabelecido ($\epsilon = 10^{-6}$).

Nas condições apresentadas ($f \leq 0,3$) é pouco provável que os adversários atinjam o objetivo de confirmar um bloco conflitante, mas é esperado que estas tentativas resultem em impactos na latência de confirmação R_c do CPoS. Para ilustrar esta afirmação, caso os adversários controlem $f = 0,3$ do *stake* total e utilizem este recurso para o ataque, apenas $1 - f = 0,7$ do *stake* total estará contribuindo de maneira honesta com a evolução da blockchain. Como consequência, é esperado um aumento da latência de confirmação R_c, já que em geral mais rodadas serão necessárias para que a média de confirmação seja capaz de atender a condição de confirmação ($\bar{s}_v^{(x)} \geq s_{min}^{(x)}$).

A medida que f cresce para valores maiores que os apresentados ($f > 0,3$), e se aproxima de $0,5$, a latência de confirmação segue aumentando até que a visão atual seja incapaz de confirmar novos blocos, fato que é mais provável quando os atacantes já dispõem da maioria do *stake* ($f > 0,5$). Além disso, a partir do aumento da fração f, a probabilidade de um bloco ser confirmado de maneira conflitante também aumenta. No capítulo 6 são apresentados resultados práticos sobre o desempenho do CPoS quando

existe a presença de adversários na rede, controlando uma fração do *stake* total.

Por fim, os ataques baseado em divulgação seletiva de blocos, são controlados por meio do processo aleatório de formação dos *peers* na rede, como apresentado em detalhes no Anexo C. Este processo evita que os adversários possam escolher de maneira determinística seus próprios *peers*, reduzindo a possibilidade de sucesso no *Eclipse Attack*. Além disso, o mecanismo é capaz de controlar o número de identidades dos adversários por meio da obrigatoriedade de manter ao menos um *stake* associado a cada endereço durante o processo de conexão, evitando a criação sem custo de identidades irmãs (*Sybil Attack*) com a finalidade de atacar o sistema.

5.4 Conclusão

A latência de confirmação R_c de um bloco foi apresentada em função dos parâmetros τ (número esperado de sorteios bem sucedidos por rodada) e ϵ (parâmetro de segurança), mostrando que é possível diminuir tal latência aumentando τ, mas mantendo a segurança do consenso constante e dentro de uma faixa aceitável. Por outro lado, é possível também manter τ constante e mudar o valor de R_c por meio da variação do parâmetro de segurança ϵ.

Como mostrado, o CPoS é robusto contra adversários que buscam realizar a confirmação de blocos conflitantes, desde que tais adversários não possuam a maioria do stake considerado no período de criação do bloco. Porém como esses adversários estão em geral trabalhando em cadeias alternativas, é esperado que eles atrasem a evolução da confirmação de blocos. Por fim, foi apresentada uma análise dos critérios que devem ser considerados pelos nós para decidirem realizar a ressincronização com a cadeia correta, já que em uma infraestrutura Bizantina é possível que alguns nós sigam visões diferentes daquela que o restante da rede está considerando.

No próximo capítulo, estes e outros conceitos teóricos são validados por meio da comparação com uma implementação prática do mecanismo CPoS em um ambiente descentralizado, capaz de reproduzir as condições de funcionamento em um ambiente próximo do esperado em situações reais.

6 Implementação do mecanismo CPoS e seus resultados práticos

Neste capítulo são apresentados resultados práticos de uma implementação em *Python* do CPoS, a partir da sua execução em um ambiente formado por um conjunto de *hosts*, organizados de modo a oferecer descentralização geográfica aos testes. Desta forma, o mecanismo é analisado por meio de diferentes configurações, em um cenário capaz de representar situações reais, em relação ao tempo de propagação dos blocos entre todos os nós da rede.

Em todos os testes realizados, os relógios dos nós foram sincronizados e a tolerância foi mantida em $tol = 2$ ($[r; r + 2]$), o que foi feito para garantir nos cenários avaliados que nenhum bloco fosse aceito se estivesse com rodada mais adiantada que o limite $r + 2$. Como $tol = 2$ produz um intervalo de rodadas bastante conservador, permitindo que blocos adiantados em até duas rodadas sejam aceitos, ele foi suficiente para garantir que nenhum bloco fosse descartado por estar com rodada adiantada até esse limite. O impacto de adotar esse tipo de intervalo de tolerância é melhor discutido no Anexo B.

6.1 Testes básicos

A Fig. 6.1, mostra o ambiente de testes formado por 25 *hosts*, subdivididos em sites administrados por diferentes organizações. Cada *host* é parte de um *cluster* gerenciado pelo Docker swarm. O Docker permite que instâncias do CPoS sejam encapsuladas em containers, oferecendo alta portabilidade e redução do tempo de *deploy*. Já o Swarm permite distribuir de maneira balanceada os containers com as instâncias do CPoS entre os *hosts* do cluster (MARATHE; GANDHI; SHAH, 2019). De acordo com a figura, cada container alocado em um *host* recebe uma instância do CPoS e um endereço IP privado.

A comunicação entre os containers de diferentes sites é garantida por meio de uma rede *overlay*, criada através do Docker. Assim, quando um container inicia uma comunicação externa, utilizando seu endereço IP de origem privado, suas transmissões são encapsuladas em um novo pacote. Desta forma, a comunicação passa a utilizar o IP público associado ao *host* e, apenas no destino, a informação é desencapsulada e entregue ao destinatário. Com esta técnica, é possível que mais de um container seja alocado em um mesmo *host*, permitindo que o CPoS seja também validado com um número maior de nós.

A partir desta infraestrutura distribuída foram realizadas execuções do meca-

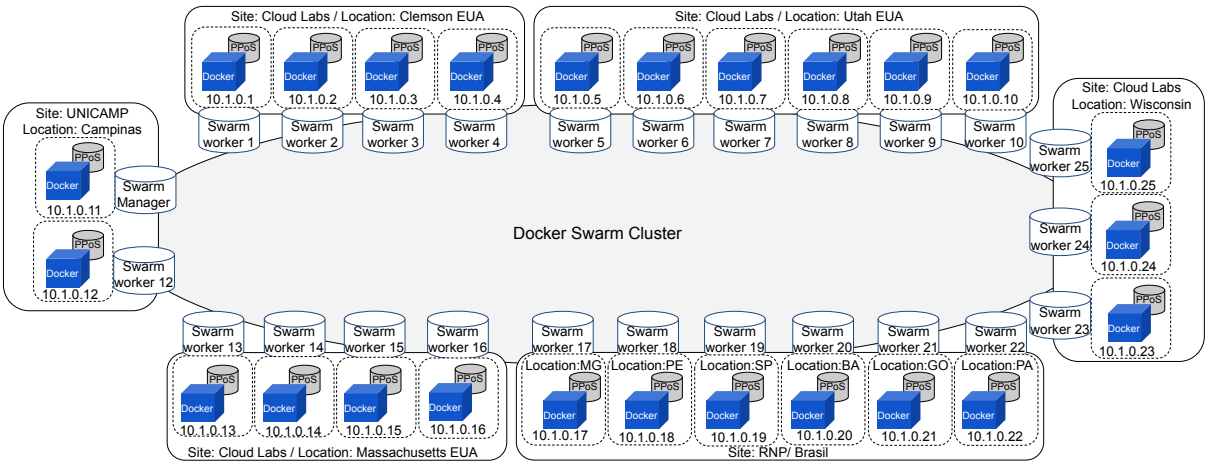

Figura 6.1 – Distribuição dos *hosts* no ambiente de testes

nismo com o objetivo de validar primeiramente a latência de confirmação R_c apresentada na seção 5.1. Neste sentido, o mecanismo foi executado utilizando diferentes valores para τ e ϵ, em uma rede com 400 nós igualmente distribuídos entre os *hosts* da topologia apresentada na Fig. 6.1. O tamanho do bloco foi mantido em um MByte para todas as execuções e cada nó foi conectado a três *peers*, seguindo o modelo de seleção de *neighbors* apresentado no Anexo C. Os valores apresentados no gráfico da Fig. 6.2 representam a latência de confirmação média ($\bar{R}_c$) em uma execução do CPoS com 100 rodadas. Neste mesmo gráfico, é apresentada a latência de confirmação esperada (R_c) para cada valor de τ e ϵ avaliados.

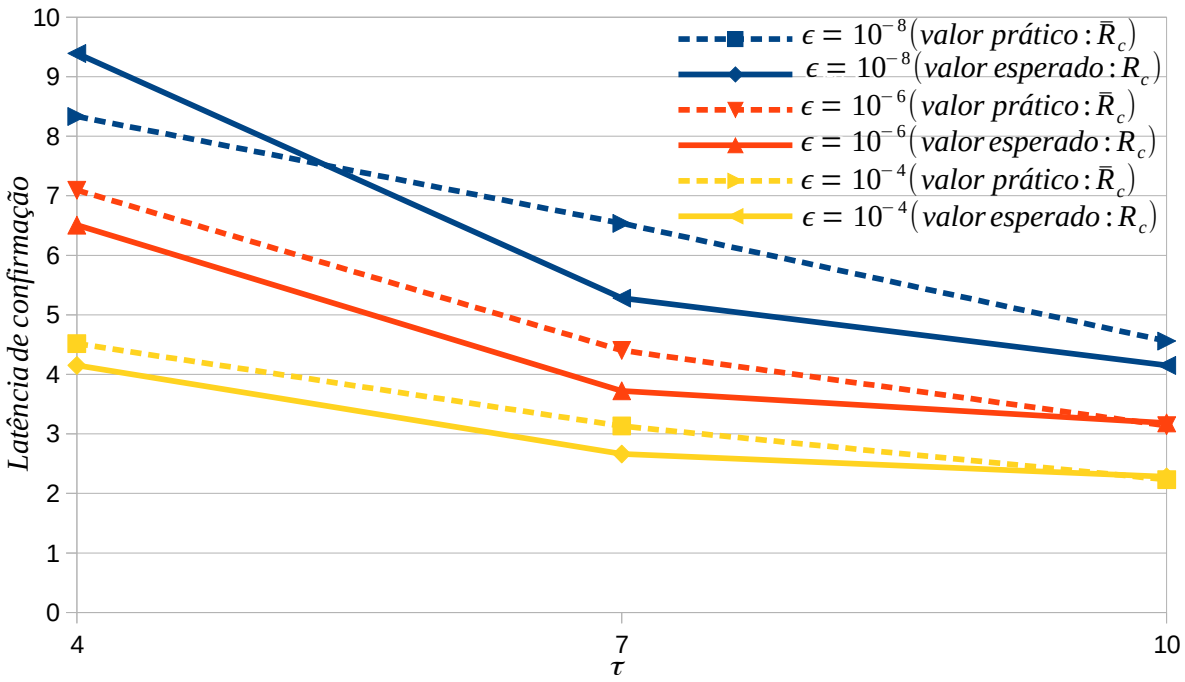

Figura 6.2 – Comparativo entre a latência de confirmação esperada R_c e a latência média $\bar{R}_c$ obtida através da execução do CPoS.

O erro relativo máximo na comparação entre a latência esperada R_c e a latência média $\bar{R}_c$ foi de 19% observado para $\tau = 7$ e $\epsilon = 10^{-8}$, enquanto o erro médio global da comparação foi de $9,8\%$. Pode-se afirmar que houve uma boa convergência entre a latência média de confirmação e os valores teóricos esperados, mostrando que o CPoS é capaz de confirmar blocos com uma latência próxima da esperada.

Já o gráfico da Fig. 6.3 mostra a diferença em números absolutos entre a latência esperada R_c e a latência média $\bar{R}_c$ obtida anteriormente. No gráfico é observada

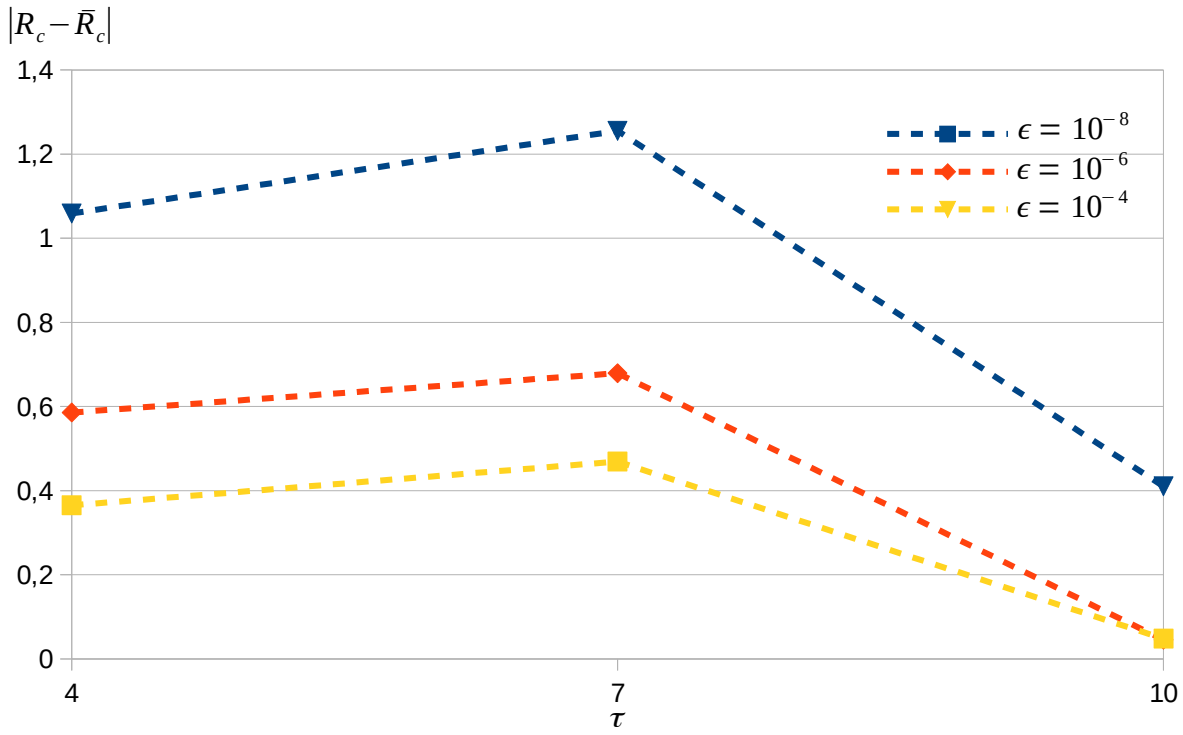

Figura 6.3 – Diferenças entre os valores práticos e teóricos são menos acentuados para ϵ maior em execução com 100 rodadas.

uma diferença menor entre as latências para valores de ϵ maiores. Isso ocorre porque, em geral, os blocos são confirmados mais facilmente a partir de um parâmetro de segurança menos rigoroso, o que contribui para que a média mínima de confirmação seja atingida mais rapidamente. Isso faz com que boa parte dos blocos não sofram a influência dos possíveis intervalos em que o número de sorteios fica abaixo do valor esperado. Já a redução da diferença em função do aumento do parâmetro τ, pode ser atribuída ao aumento da margem de confirmação $M^{(x)}$, que facilita a confirmação dos blocos mesmo quando a rede tem uma quantidade de sorteios abaixo do esperado.

Portanto, os resultados apresentados no gráfico da Fig. 6.2 e 6.3 comprovam que o CPoS pode alterar a latência de confirmação por meio da variação de seus dois principais parâmetro: τ e ϵ. Assim, por exemplo, para uma aplicação que exige baixa latência de confirmação e que não pode flexibilizar a segurança do consenso como, por exemplo, os sistemas de pagamento eletrônico, é recomendável aumentar o parâmetro τ para que as

transações dos pagamentos sejam confirmadas mais rapidamente a partir de uma latência menor, ao mesmo tempo em que a segurança possa ser mantida em um patamar aceitável, por meio da redução do parâmetro ϵ. Para alcançar os resultados apresentados nos gráficos, a duração da rodada foi ajustada para garantir que os blocos fossem recebidos dentro de suas rodadas de criação por todos os nós da rede, o que colaborou para a manutenção de uma única visão da blockchain nos nós durante todo o período de testes. Para atingir esse objetivo, foi utilizado $T = 45\ s$ para a rede de testes com 400 nós.

Por outro lado, se a rede tiver um tamanho e um volume de tráfego que inviabilizem o recebimento de todos os blocos dentro da rodada atual, alguns blocos passam a ser recebidos de maneira atrasada, colaborando para a coexistência de diferentes visões da blockchain, o que dificulta a convergência da confirmação probabilística em torno de uma única cadeia produzida. Para ilustrar este cenário, o gráfico da Fig. 6.4 mostra a latência de confirmação média ($\bar{R}_c$) para diferentes execuções do CPoS com 100 rodadas de duração cada uma delas, em uma rede com 400 nós igualmente distribuídos entre os *hosts* da topologia mostrada na Fig. 6.1. Em cada execução a duração da rodada e o valor de τ foram alterados, mantendo $\epsilon = 10^{-6}$. De acordo com o gráfico, a latência de confirmação

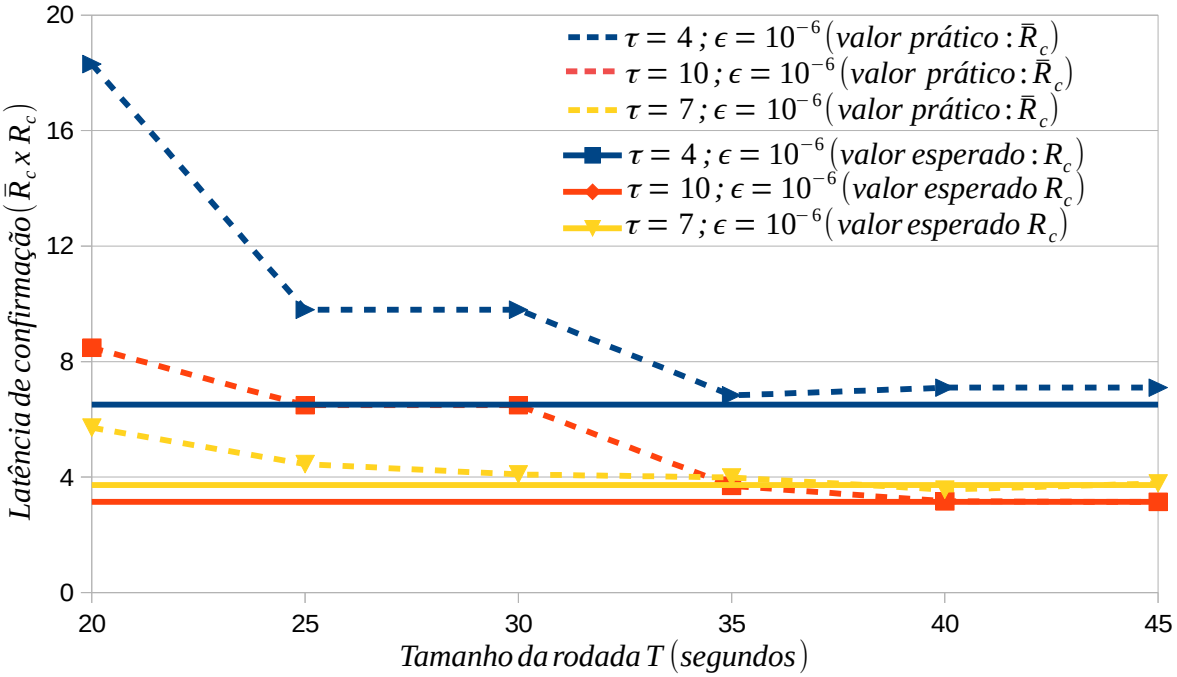

Figura 6.4 – Latência de confirmação média converge para o valor esperado quando uma rodada com tempo de duração adequado é utilizada.

média ($\bar{R}_c$) calculada converge para o valor teórico esperado à medida que a duração da rodada se aproxima do mínimo necessário. Para valores de τ menores (Ex: $\tau = 4$) a convergência ocorre mais rapidamente, já que são produzidos e transmitidos menos blocos a cada rodada, o que reduz o tempo para que todos os blocos sejam propagados na rede.

Assim, para todo tamanho de rede e de τ, existe um valor ideal para T, que

permite ao protocolo atingir seu desempenho máximo; entretanto, o gráfico mostra que aumentar T para além deste valor não traz melhorias para a latência do mecanismo, já que a partir de uma rodada com a duração mínima necessária, a latência passa a depender apenas dos parâmetros ϵ e τ. Neste sentido, é importante utilizar um valor de rodada próximo do ideal, pois rodadas maiores que este valor aumentam o tempo para que um bloco seja confirmado além de reduzir o número de transações confirmadas por segundo, como será mostrado mais adiante. Para efeito de comparação, os experimentos descritos acima também foram realizados em uma rede com 100 nós, onde foi possível manter a duração da rodada em 20 segundos.

As rodadas menores que o mínimo necessário para que todos os blocos sejam recebidos por todos os nós da rede dentro da rodada atual também aumentam o número de pedidos de ressincronizações, já que contribuem para que blocos sejam recebidos de maneira atrasada pelos nós da rede, o que pode provocar visões diferentes da blockchain. Esse cenário é indesejado, pois além de aumentar a latência de confirmação também contribui com o aumento do tráfego de rede devido às trocas de mensagens destinadas às ressincronizações. Todavia, é importante simular esta situação para conhecer qual é a capacidade do mecanismo de ressincronização proposto, ou seja, é necessário verificar se os nós são capazes de detectar rapidamente que estão com uma visão errada, e assim reverter poucos blocos durante o retorno para a cadeia correta. Deste modo, durante as execuções foi medido o número médio de pedidos de ressincronização por nó, como será mostrado a seguir.

Como mostra o diagrama de sequências da seção A.3 do Anexo A, um nó verifica se uma ressincronização é necessária sempre no início de cada rodada. Neste sentido, o número médio de ressincronizações por nó pode ser interpretado também como o número médio de rodadas em que houve ressincronizações nos nós. Como os testes foram executados até que 100 rodadas completas fossem obtidas, essa média representa o número de rodadas com ressincronizações em um total de 100 rodadas executadas. No gráfico da Fig. 6.5 são apresentados os resultados para uma análise com $N = 400$ nós igualmente distribuídos entre *hosts* da topologia mostrada na Fig. 6.1, $\epsilon = 10^{-6}$ e diferentes valores de τ e T. De acordo com o gráfico, para rodadas com tempo de duração menor, ocorre um número maior de ressincronizações para valores mais elevados de τ, já que nestes casos mais blocos são produzidos por rodada, o que aumenta o número de blocos recebidos de maneira atrasada pelos nós. Assim, é esperado, que para rodadas menores, coexistam mais visões, o que aumenta o número de ressincronizações. No gráfico também foi plotado o erro padrão em torno da média que, por ser muito pequeno, não ficou visível. Isso significa que os valores médios plotados neste e em outros experimentos são bem representativos dos dados utilizados.

Para cada teste executado, também foi calculado o número médio de blocos

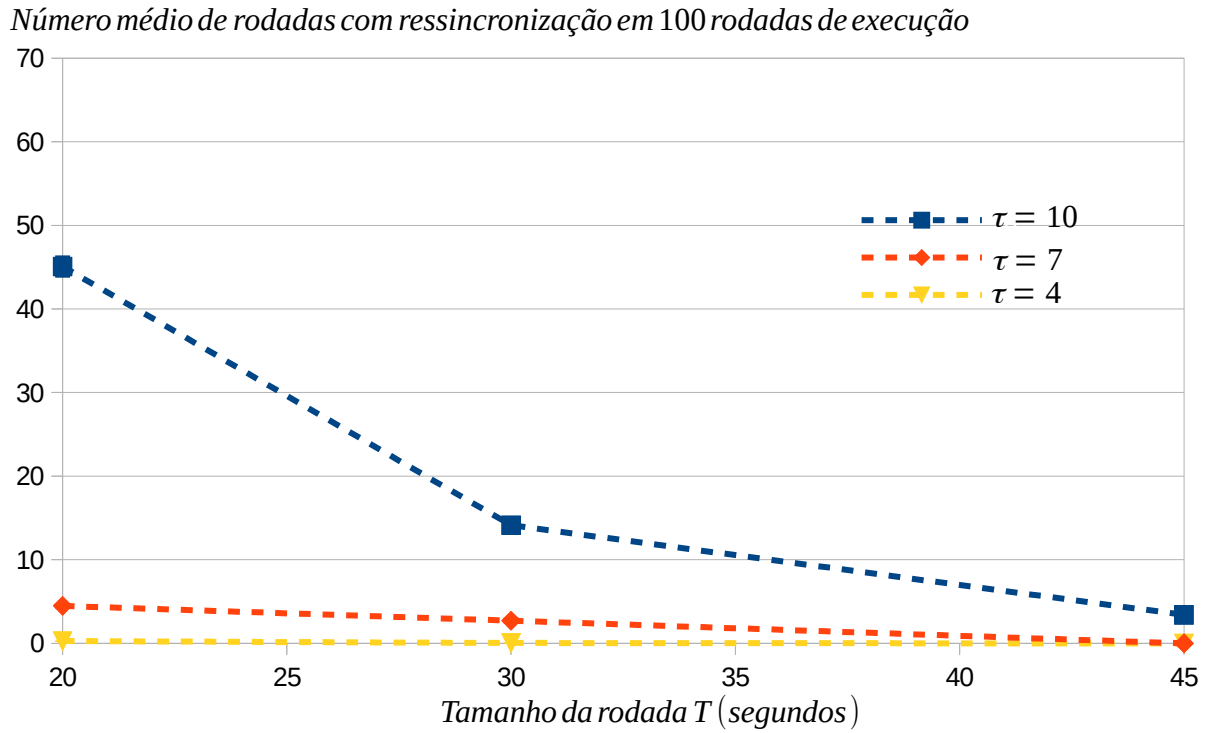

Figura 6.5 – O número médio de rodadas com pedidos de ressincronização aumenta a medida que o tempo de duração da rodada é menor que o necessário

revertidos, ou seja, o número médio de blocos que foram aceitos em um primeiro momento por algum nó, mas, que após algumas rodadas, um pedido de ressincronização mostrou que eles deveriam ser substituídos por blocos de uma outra visão, provocando uma mudança de visão na blockchain local do nó. É apresentada no gráfico da Fig. 6.6 a taxa média de blocos revertidos, ou seja, a razão entre o número médio de blocos revertidos por nó pelo número médio de ressincronizações em função da duração da rodada para diferentes valores de τ em uma rede com 400 nós. De acordo com o gráfico, é possível validar que de fato um nó do CPoS é capaz de detectar uma perda de sincronização com a cadeia principal, onde no pior caso, foram observados em média três blocos revertidos em cada novo pedido de ressincronização. Isso mostra que um nó não permanece muitas rodadas com uma visão errada da blockchain, mesmo em situações em que a rodada não é suficientemente grande para o número de nós da rede, como mostrado no gráfico da Fig. 6.4.

Por fim, para encerrar essa análise mais geral sobre os resultados, voltamos a analisar o gráfico da Fig. 6.2. As execuções práticas confirmaram que a latência de confirmação do mecanismo diminui por meio do aumento do parâmetro τ, como discutido na seção 4.5. Isso é possível devido ao valor mínimo ($s_{min}^{(x)}$), necessário para que a confirmação de um bloco ocorra ao final da rodada x, ser menor que a média esperada (τ) já nas primeiras rodadas após a criação do bloco, como mostra o gráfico 4.15.

Nas próximas seções deste capítulo são apresentados resultados do CPoS para valores de τ maiores que os mostrados até este momento. O tamanho do bloco foi mantido

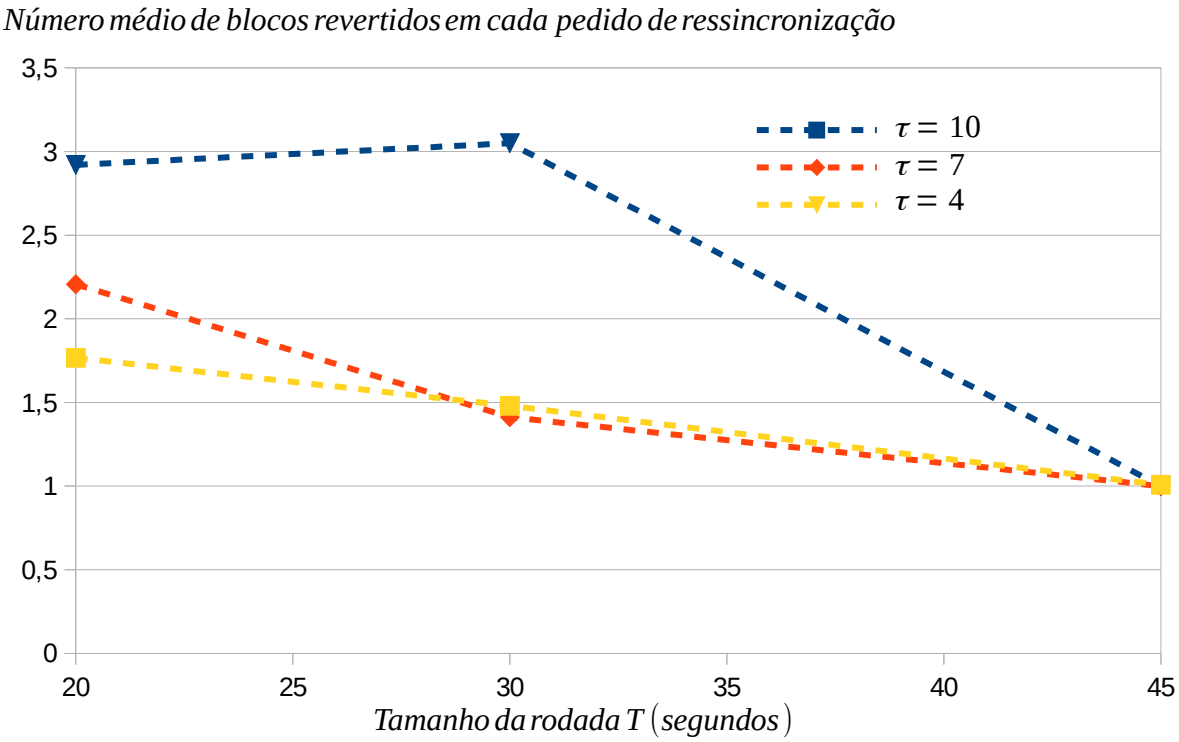

Figura 6.6 – Número médio de blocos revertidos para cada pedido de ressincronização

em $1MB$ e os nós permaneceram conectados a três *peers*.

6.2 Influência do atraso de transmissão na latência de confirmação

Nesta análise busca-se verificar o comportamento da latência de confirmação em função do aumento do tempo de transmissão dos blocos. Para isso, foram inseridos atrasos artificiais nas transmissões realizadas entre cada par de nós da rede, onde um bloco produzido em um nó passa por diferentes atrasos, até ser recebido por todos os nós da rede. Deste modo, cada transmissão realizada por um nó na direção de um de seus *peers* tem o seu próprio atraso artificial, de modo que o instante de tempo que um bloco é recebido em um nó é influenciado pela soma de todos os atrasos inseridos em cada transmissão envolvida. Os atrasos são produzidos a partir das saídas de uma função do Python (*expovariate*) que segue a distribuição exponencial da Eq. 6.1, onde d é o valor esperado dado em segundos.

$$f(x) = \frac{1}{d} e^{-\frac{1}{d}x} \tag{6.1}$$

Assim, quanto maior o valor do atraso médio d, maiores serão os atrasos aleatórios produzidos pela função.

Com estes atrasos aleatórios, foi produzida uma taxa de chegada de blocos atrasados, calculada a partir da razão entre o número de blocos atrasados e o total de blocos recebidos. As possíveis alterações no valor de τ mudam apenas o número absoluto

de blocos recebidos por rodada e, consequemente, o número de blocos atrasados, mas não provocam alteração no valor dessa taxa. Sendo assim, a taxa de blocos atrasados depende apenas do parâmetro d escolhido, como mostrado no gráfico 6.7 obtido a partir de diferentes execuções do CPoS.

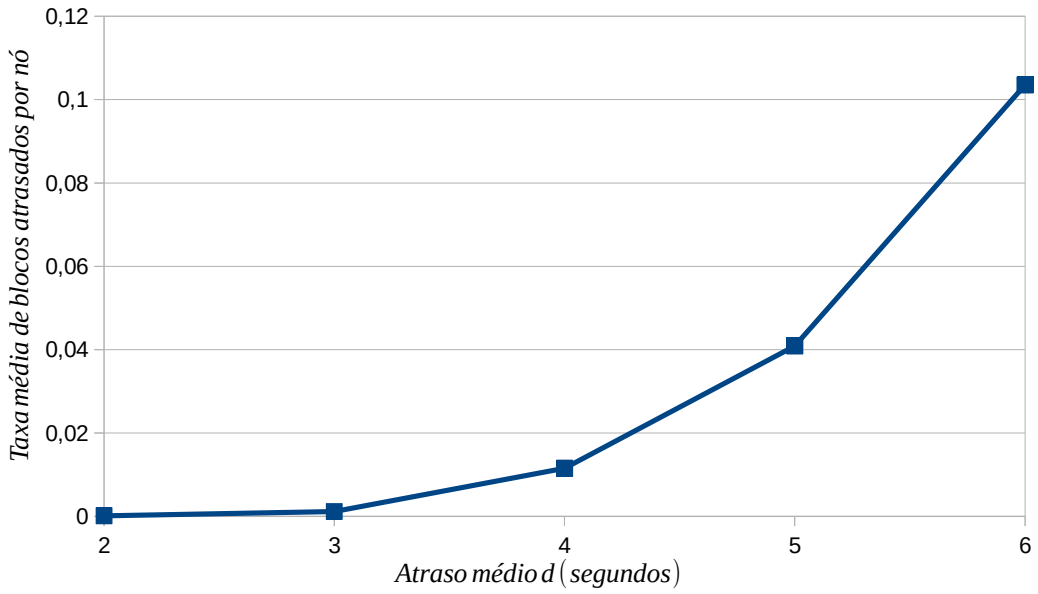

Figura 6.7 – Taxa média de chegadas de blocos atrasados em função do aumento do atraso de transmissão dos blocos

Portanto, é possível utilizar a taxa de chegada de blocos atrasados para avaliar o desempenho do CPoS em um cenário onde o valor de τ é alterado, mas a taxa de chegada de blocos atrasados permanece constante. O gráfico apresentado na Fig. 6.8 mostra a latência de confirmação média $\bar{R}_c$, a partir da execução do mecanismo durante 100 rodadas, utilizando diferentes valores de τ e d. Nestas execuções utilizou-se $N = 25$, ou seja, foi alocado um nó do CPoS em cada *host* da Fig. 6.1, para evitar possíveis distorções que a inclusão de mais nós em um *host* poderia trazer durante a utilização do atraso artificial d. Essa configuração permitiu que a rede estivesse complementamente descentralizada, ou seja, com apenas um nó alocado em cada *host*. Além disso, foi utilizado $\epsilon = 10^{-6}$ e $T = 20s$.

Como esperado, quando o atraso médio de transmissão d cresce, a latência média de confirmação do mecanismo aumenta para todos os valores de τ, já que a confirmação probabilística dos blocos torna-se mais lenta, devido ao aumento da taxa média de blocos atrasados. Por outro lado, é possível notar que esse aumento na latência é menos acentuado para valores de τ mais elevados. Para ilustrar este fato, o gráfico da Fig. 6.9 mostra a variação da latência por meio da diferença entre a latência média $\bar{R}_c$ calculada a partir de execuções com atrasos artificais ($d > 0$) e a latência média obtida em execuções com $d = 0$. Esse gráfico foi produzido a partir das mesmas execuções que deram origem ao gráfico da Fig. 6.8.

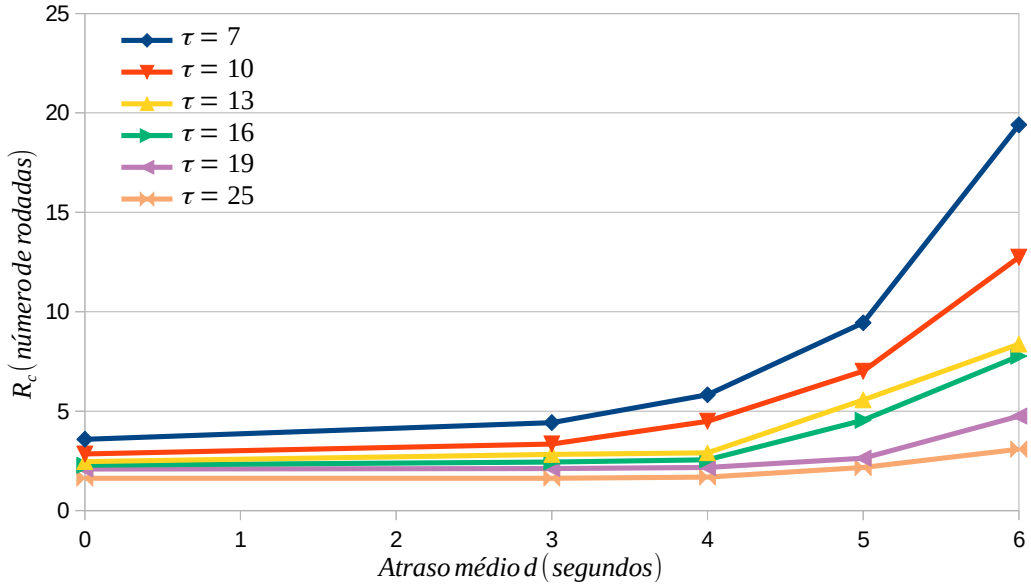

Figura 6.8 – Latência média de confirmação cresce em função do aumento do tempo de transmissão dos blocos na rede

De acordo com o gráfico da Fig. 6.9, pode-se concluir que a diferença da latência média é menos acentuada à medida que o valor de τ é aumentado, considerando um mesmo valor para d. Portanto, a partir do aumento do parâmetro τ, o mecanismo torna-se mais robusto às variações de desempenho provocadas pela chegada de blocos atrasados. Este

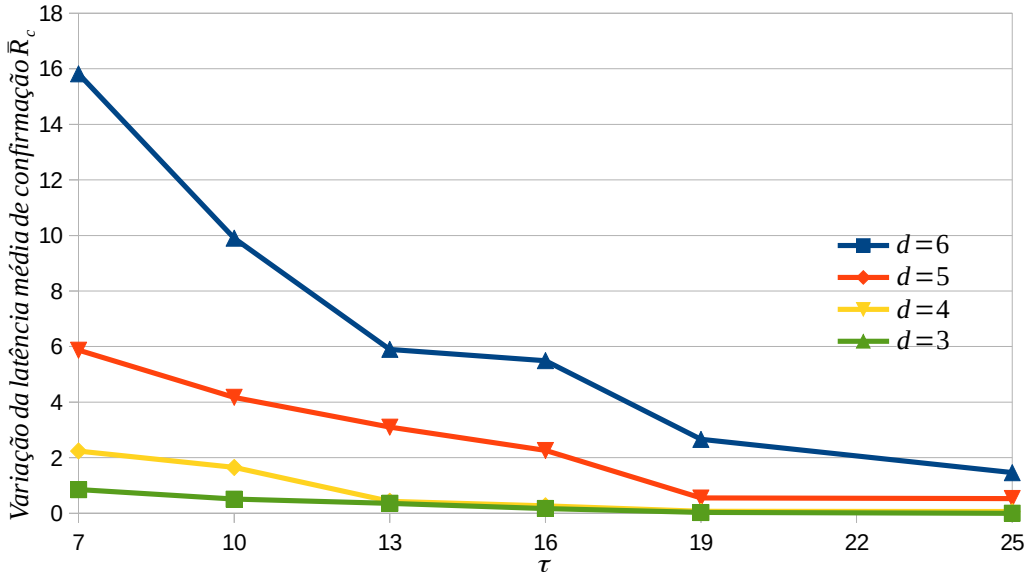

Figura 6.9 – A latência média de confirmação cresce mais lentamente em função do aumento do atraso de transmissão para valores de τ mais elevados.

fato novamente está associado à margem de confirmação $M^{(x)}$. Ao elevar o valor de τ à margem de confirmação também aumenta, contribuindo para que as confirmações ocorram a partir de médias mais baixas que o valor esperado τ, o que torna o mecanismo mais

robusto frente à possível redução da média de confirmação na presença de blocos atrasados.

6.3 Presença de adversários

Como discutido na seção 5.3, os adversários podem se organizar em grupos, direcionando o *stake* controlado por eles em prol da produção de blocos em *forks* conflitantes, ou apenas não divulgando seus blocos na rede quando membros do conluio são sorteados. Neste sentido, apesar da confirmação de blocos conflitantes ser pouco provável caso os adversários não dominem a maior parte do *stake*, é esperado um aumento na latência de confirmação do mecanismo na presença de ataques, já que o número de sorteios bem sucedidos observados pelos nós honestos na cadeia principal torna-se menor que τ e, portanto, faz-se necessário em geral um maior número de rodadas para que os blocos sejam confirmados.

Neste contexto, foram realizadas novas execuções do mecanismo CPoS no ambiente descentralizado da Fig. 6.1, considerando a presença de adversários que controlam diferentes frações do *stake* total da rede. Esses adversários colaboram entre si para que o consenso não seja alcançado pelos nós honestos. Desta forma, quando sorteados, eles não divulgam o bloco produzido, o que traz impactos diretos para o mecanismo de consenso, já que os nós honestos passam a dispor de menos sorteios bem sucedidos que confirmem suas visões a cada rodada. Esse aumento na latência de confirmação média $\bar{R}_c$ foi analisado para diferentes valores de f e τ, considerando $\epsilon = 10^{-6}$ e uma rede completamente descentralizada em relação ao número de *hosts* disponíveis ($N = 25$) com $T = 20s$. Os resultados destas execuções estão apresentados no gráfico da Fig. 6.10, onde é possível notar o crescimento da latência em função do aumento de f para todos os casos analisados. De acordo com o gráfico da Fig. 6.10, é possível afirmar que, os adversários podem contribuir para o aumento da latência de confirmação média já a partir de $f = 0,05$, apesar da probabilidade de confirmação de um bloco conflitante ser de fato pequena (Seção 5.3). É esperado que para valores de f maiores que os apresentados no gráfico, a latência de confirmação média $\bar{R}_c$ continue aumentando até o ponto que os nós honestos não sejam mais capazes de confirmar novos blocos, fato que é mais provável que ocorra quando a soma do *stake* dos nós desonestos representar a maior parte ($f > 0,5$) do *stake* total.

Outro ponto importante é que, em geral, a variação da latência média $\bar{R}_c$ diminui a partir do aumento do parâmetro τ, como mostrado no gráfico 6.11. A variação da latência, neste caso, é obtida por meio da diferença entre o valor da latência $\bar{R}_c$, obtida para um f qualquer ($f > 0$), e o valor desta latência para $f = 0$ (melhor caso). Assim como já ocorreu na análise da seção anterior, a partir de uma margem de confirmação $M^{(x)}$ mais elevada obtida por meio do aumento do parâmetro τ, o CPoS pode apresentar uma maior robustez frente à variações de desempenho, que ocorrem quando nós desonestos

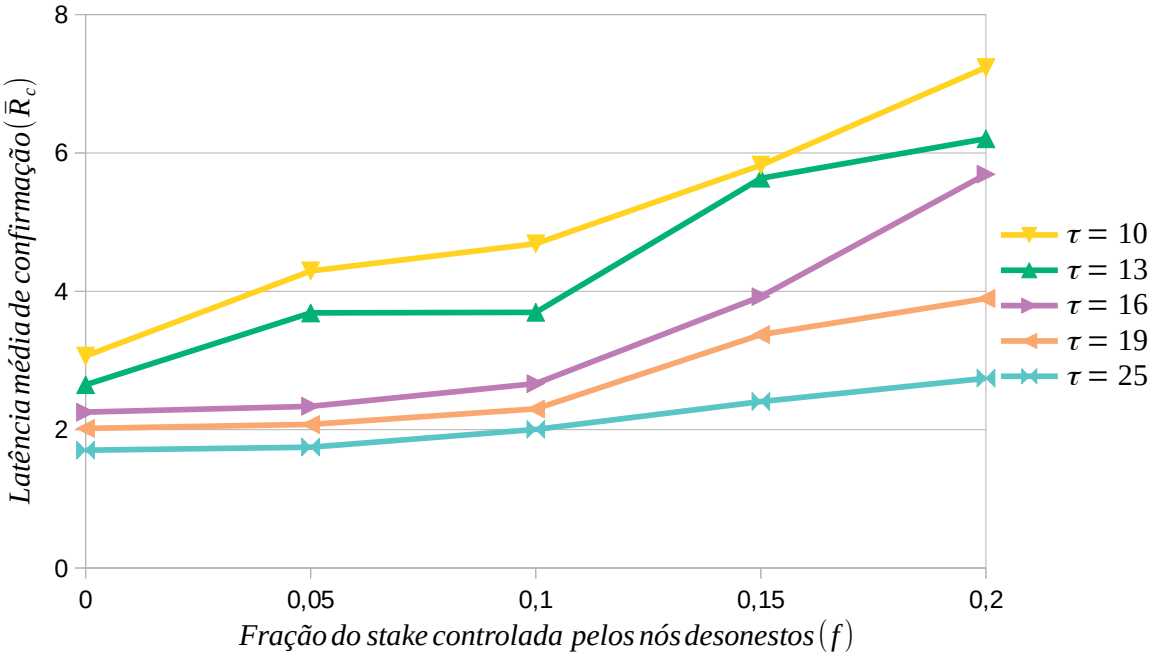

Figura 6.10 – Latência média de confirmação cresce quando o *stake* controlado pelos adversários aumenta

não participam dos sorteios ou colaboram com visões diferentes daquela que é seguida pelos nós honestos.

Por fim, o número médio de pedidos de ressincronizações por nó também aumenta, já que o limiar de ressincronização $\alpha^{(x)}$ passa a não ser superado com tanta facilidade como no caso sem ataques, aumentando o número de falsos positivos. Isso ocorre devido à média de sorteios bem sucedidos ser menor que τ, quando existem adversários que não divulgam os blocos produzidos. O aumento do número de ressincronizações foi avaliado durante as execuções descritas nesta seção e o resultado é apresentado no gráfico da Fig. 6.12. Nele, é mostrado o valor médio por nó da razão entre o número de rodadas em que houve ressincronizações e o número total de rodadas dos testes (100 rodadas). Nota-se que o valor cresce em função do aumento do parâmetro f para todos os valores de τ analisados.

6.4 Transações confirmadas por segundo

A partir do tamanho do bloco utilizado nos testes é possível estimar o desempenho do CPoS em relação ao número médio de transações confirmadas por segundo, utilizando a Eq. 6.2. O parâmetro $Block_{size}$ indica o tamanho do payload dos blocos transmitidos, ou seja, é o espaço ocupado pelas transações dentro do bloco. Já o tamanho

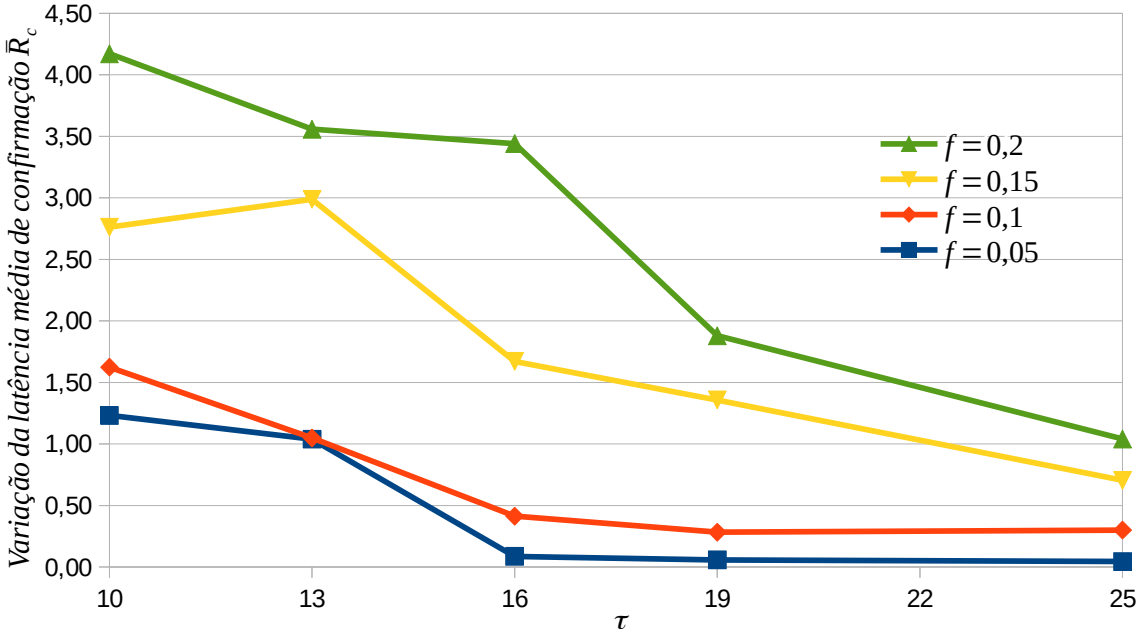

Figura 6.11 – Variação na latência de confirmação é menos acentuada para τ mais elevado na presença de adversários

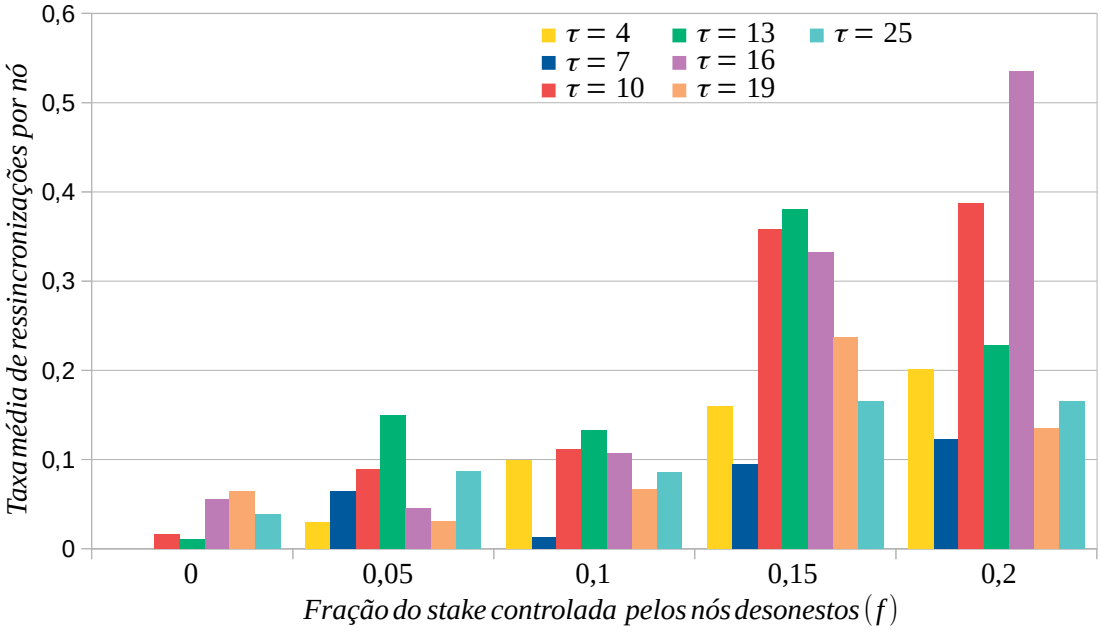

Figura 6.12 – Fração f aumenta a taxa média de ressincronizações devido ao consequente aumento dos falsos positivos

médio de cada transação é dado pelo parâmetro Tx_{size}.

$$Trans/s = \frac{block_{size}}{Tx_{size} \times T} \tag{6.2}$$

Para realizar a estimativa, considerou-se $Block_{size} = 1MB$ e $Tx_{size} = 400$ bytes. Estes valores estão coerentes, por exemplo, com os números obtidos na blockchain do Bitcoin (BLOCKCHAIN.INFO, 2020). Desta forma, tomando como base a variação

do tempo de duração da rodada T, o gráfico da Fig. 6.13 mostra o número de transações confirmadas por segundo esperado para o CPoS.

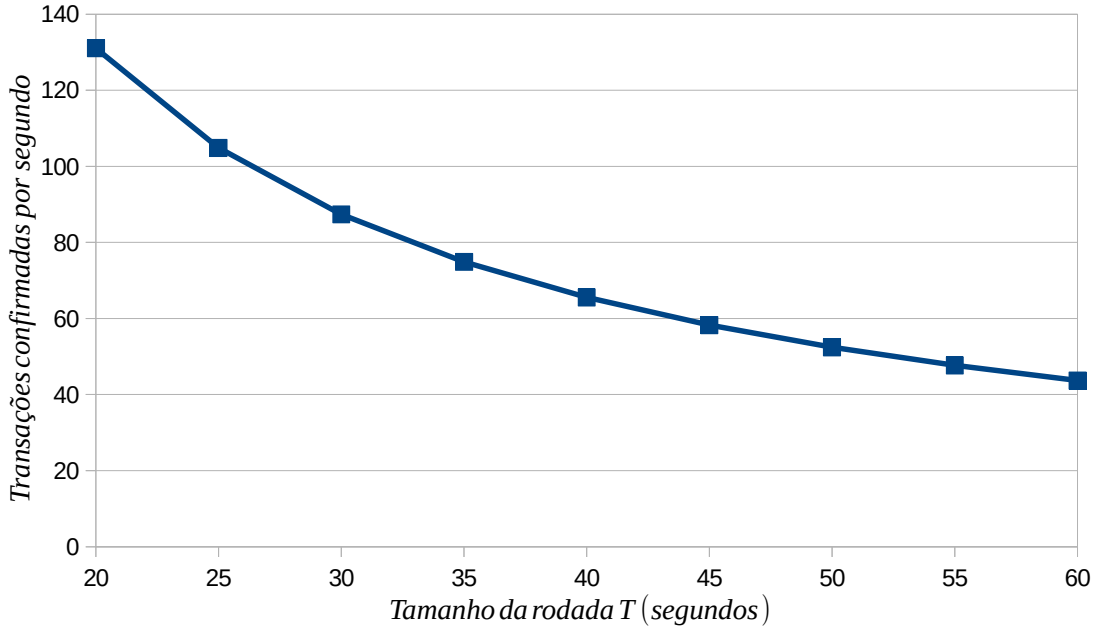

Figura 6.13 – O tempo de duração da rodada influencia o número de transações confirmadas por segundo.

Sabe-se que a escolha de τ está associada diretamente ao número de blocos produzidos em uma rodada do CPoS. Sendo assim, caso τ seja pequeno ($\tau < 10$) passa a ser significativa a probabilidade de ocorrência de rodadas em que nenhum nó seja sorteado para produzir novos blocos, o que é chamado de lacuna. Assim, para um valor de τ pequeno, é possível dizer que ele pode influenciar no número de transações confirmadas por segundo do CPoS, pois caso existam lacunas o mecanismo não terá o bloco dessa rodada específica para confirmar. Porém, assumindo que serão utilizados valores de τ suficientemente grandes ($\tau \geq 10$), é possível afirmar que existe uma alta probabilidade dos nós produzirem novos blocos em todo início de rodada e, portanto, o único parâmetro controlável do CPoS que pode influenciar no número de transações confirmadas por segundo passa a ser o tempo de duração da rodada (T). A título de comparação, a blockchain do Bitcoin tem uma taxa teórica média de 7 transações por segundo (NGUYEN et al., 2019).

Desta forma, espera-se que a taxa de indexação dos novos blocos produzidos seja próxima de um bloco por rodada. Nestas condições, a taxa de blocos confirmados por rodada também será próxima de um, já que é esperado que em todo início de rodada o CPoS seja capaz de confirmar um novo bloco. Mais precisamente, é esperado que o mecanismo confirme em todo início de rodada o bloco que foi produzido R_c rodadas antes.

6.5 Análise do comportamento da rede

A partir dos resultados apresentados nesta seção, é possível afirmar que a latência de confirmação do mecanismo de consenso CPoS está associado ao valor de τ, já que, por meio do aumento deste parâmetro, é possível reduzir o tempo necessário para que um bloco seja confirmado, sem alterar o limiar de segurança ϵ estabelecido. Além disso, o aumento do parâmetro τ torna o protocolo mais robusto no que tange à sua capacidade de manter a latência de confirmação de blocos próxima do valor esperado, mesmo com a presença de adversários que controlam uma fração f do *stake* ou quando blocos são perdidos devido ao aumento da taxa média de atrasos.

Por estes motivos, é recomendado que o CPoS utilize o maior valor de τ possível que não traga um impacto muito forte no tráfego de rede. Assim, é importante conhecer os impactos causados por este aumento, que estão relacionados com o tráfego de rede gerado em cada nó e ao atraso envolvido na propagação do bloco na rede *peer-to-peer*.

6.5.1 Tempo médio de transmissão dos blocos na rede

Um ponto que deve ser avaliado, é o tempo médio que os blocos levam desde que são criados até o momento em que são recebidos pelos nós que compõem a rede *peer-to-peer*. Este tempo foi medido através das execuções do protocolo, utilizando $\epsilon = 10^{-6}$, $T = 20s$. Os resultados dessas execuções foram obtidos para diferentes números de nós igualmente distribuídos entre os *hosts* da topologia mostrada na Fig. 6.1. Esses resultados são apresentados no gráfico da Fig. 6.14. O gráfico mostra que o tempo médio de transmissão

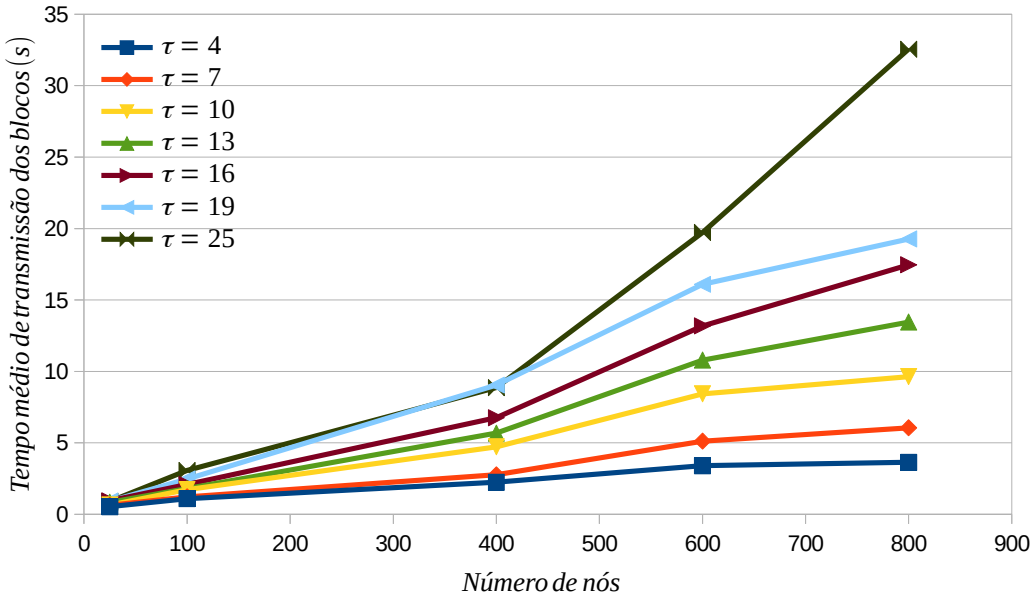

Figura 6.14 – Tempo médio de transmissão dos blocos cresce em função do aumento do parâmetro τ e do tamanho da rede.

dos blocos cresce a partir do aumento do parâmetro τ, para uma rede com o mesmo número de nós. Isso ocorre devido ao aumento do número de blocos que cada nó deve transmitir na rede a cada rodada para um valor de τ maior, o que colabora para que o tempo de permanência de um bloco na fila de transmissão de cada nó também seja maior.

Por outro lado, quando o número de nós na rede cresce, o número de saltos até que um bloco seja recebido por um nó também cresce, o que justifica o aumento do tempo médio de transmissão dos blocos para este caso, considerando agora um mesmo valor de τ. Como exemplo, o gráfico mostra que para $\tau = 25$, já não é possível garantir que o tempo de transmissão médio dos blocos seja menor que a duração da rodada definida ($T = 20s$), a partir de uma rede com 600 nós. Portanto, em caso de redes maiores, é preciso avaliar o aumento da duração de rodada T ou diminuição de τ para aceitação de um bloco.

6.5.2 Tráfego médio gerado por nó

O tráfego médio gerado em cada nó depende do parâmetro τ, ou seja, do número médio de blocos produzidos por rodada, e também do número de *peers* com que cada nó se conecta. Para esta análise, é apresentado no gráfico da Fig. 6.15 o comportamento do tráfego médio por nó, medido a partir de execuções práticas do protocolo para $N = 25$, $\epsilon = 10^{-6}$ e $T = 20s$, considerando que cada nó está alocado em um *host* da topologia apresentada na Fig. 6.1. De acordo com o gráfico, é possível notar que o tráfego médio

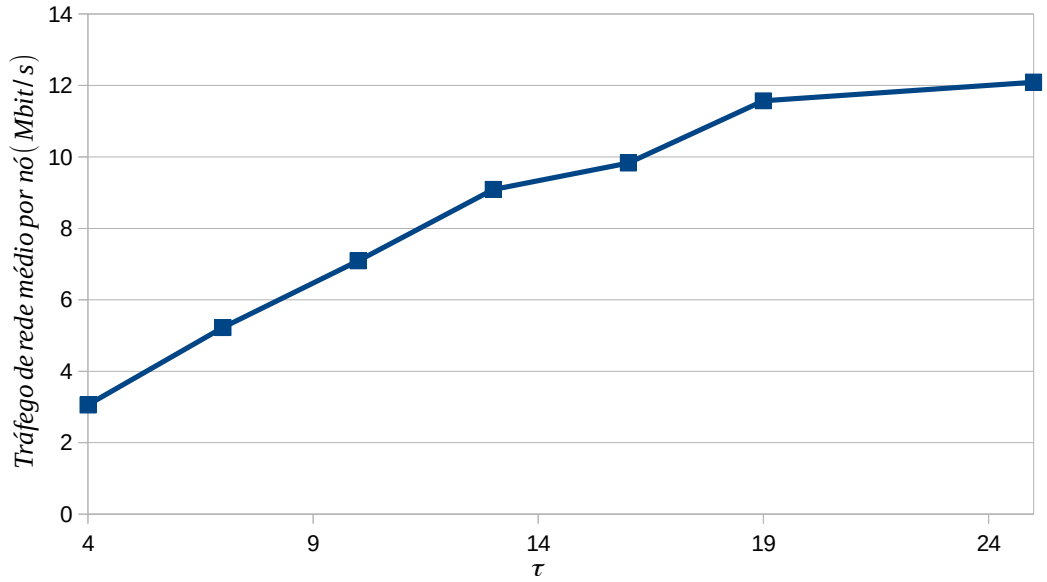

Figura 6.15 – Comportamento do tráfego médio de rede por nó em função de τ para $T = 20s$ e $\epsilon = 10^{-6}$

por nó aumenta em função de τ, como esperado. Assim, o crescimento do tráfego de rede é um limitante que impede que o valor de τ seja aumentado de modo arbitrário, pois o tráfego de rede gerado pode dificultar a participação de alguns nós no consenso.

Alguns valores típicos dos principais parâmetros envolvidos no CPoS são mostrados na tabela 6.1. Essa tabela relaciona a latência de confirmação esperada (R_c) com aquela que é esperada na presença de um conluio que envolva uma fração f do stake, além do tráfego médio por nó. Os valores são tabelados para diferentes valores de τ com $\epsilon = 10^{-6}$ e podem ser utilizados como um ponto de partida para os desenvolvedores de novas aplicações nas configurações dos parâmetros.

Tabela 6.1 – Comparativo entre os principais parâmetros do CPoS

τ	ϵ	Latência R_c	Latência com conluio ($f \leq 0,2$)	Tráfego médio por nó (Mbit/s)
4	10^{-6}	$4,12$	$10,12$	$3,06$
7	10^{-6}	$2,66$	$8,16$	$5,22$
10	10^{-6}	$2,28$	$7,23$	$7,09$
13	10^{-6}	$1,92$	$6,20$	$9,09$
16	10^{-6}	$1,70$	$5,69$	$9,83$
19	10^{-6}	$1,54$	$3,89$	$11,57$
25	10^{-6}	$1,42$	$2,74$	$12,09$

Apesar do tráfego médio gerado por nó aumentar em função de τ não é esperado que ele aumente a partir do crescimento do número de nós na rede. Isso ocorre porque os nós individualmente transmitem seus blocos apenas para seus *peers* e, como é esperado que haja uma formação de *peers* pseudoaleatória (seção C.2 do anexo C), pode-se dizer que a rede mesmo que cresça estará com suas conexões de rede bem distribuídas entre os diferentes nós, reduzindo a possibilidade de aumento de tráfego em alguns nós particulares ou mesmo do tráfego médio gerado por nó.

Ainda assim, é importante avaliar alternativas para reduzir o tráfego de rede por nó. Essa redução, se possível, trará impactos positivos em relação ao desempenho do mecanismo, já que a partir de um tráfego médio menor, é possível reduzir também a duração da rodada de produção dos blocos. A próxima seção traz uma discussão sobre como o tráfego de rede poderia ser reduzido.

6.5.3 Proposta de redução do tráfego de rede

Uma alternativa para o controle do tráfego é a não transmissão de todos os blocos com suas respectivas transações a cada rodada. Ao invés disso, em um primeiro momento, são transmitidos pelos nós sorteados apenas os cabeçalhos dos blocos e somente aquele nó que produzir o bloco com o menor *hash* de prova (nó vencedor) entre todos os sorteados deve transmitir o bloco completo na rede. Nesta nova abordagem, há um risco de que o nó vencedor não divulgue o bloco com as respectivas transações e, neste caso, é necessário definir um novo critério a ser seguido quando essa situação ocorrer.

É preciso definir o momento que um determinado nó sorteado é declarado vencedor. No cenário atual, os nós decidem durante a rodada de criação qual bloco deve ser mantido na sua visão local, a partir da aplicação dos critérios estabelecidos na seção

4.4. Portanto, no início de uma rodada, todas as decisões de aceitação já foram tomadas na rodada anterior e o nó pode iniciar um novo ciclo de produção. Nesta nova abordagem, é conveniente que esta estratégia seja mantida, ou seja, que o envio do bloco vencedor seja realizado em um momento posterior à transmissão dos cabeçalhos, mas ainda dentro da mesma rodada. Uma das formas de atingir este objetivo é por meio da divisão da rodada em duas fases distintas, como mostrado na Fig. 6.16.

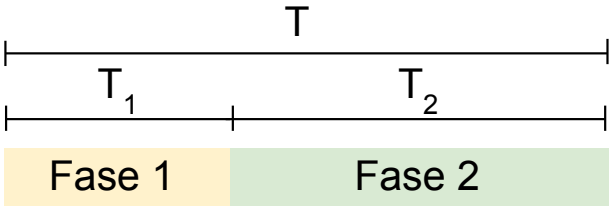

Figura 6.16 – Nós sorteados enviam os cabeçalhos de seus blocos na fase 1. Nó vencedor envia o bloco completo na fase 2

Na fase 1, os nós sorteados enviam os respectivos cabeçalhos dos blocos associados aos seus melhores *hashes* de prova. Já no início da fase 2, o nó sorteado, dono do bloco com o menor *hash* de prova, transmite o seu bloco com todas as transações selecionadas. A partir deste cenário e considerando a situação ideal em que o nó vencedor sempre divulga seu bloco no início da fase 2, foram realizadas novas execuções do mecanismo com o objetivo de avaliar o novo tráfego de rede para $N = 25$ nós que foram igualmente distribuídos entre os *hosts* da topologia mostrada na Fig. 6.1. Além disso, foi utilizado $\epsilon = 10^{-6}$ e $T = 20s$.

Os resultados desta avaliação estão apresentados no gráfico da Fig. 6.17. Primeiramente, nota-se que a partir desta nova abordagem, o tráfego médio por nó é consideravelmente reduzido quando comparado com a implementação atual. Isso ocorre devido ao fato do cabeçalho do bloco ser muito menor que o bloco completo, ou seja, com todas as transações. Além disso, o novo tráfego se mantém quase constante em função de τ, mostrando que essa abordagem pode ser efetiva no controle do tráfego de rede mesmo para valores de τ mais elevados. Este segundo fato tem grande relevância para o CPoS, já que a partir desta nova estratégia é possível elevar o valor de τ, o que melhora a latência de confirmação, sem aumentar com isso o valor do tráfego médio de rede por nó.

Uma solução intermediária entre a abordagem atual e a nova proposta apresentada acima pode ser baseada em uma nova condição, que permite a transmissão de todos os k blocos com os menores *hashes* de prova. Escolhendo $k > 1$ e menor que τ, é possível reduzir a probabilidade de não haver blocos completos divulgados na rodada, já que é pouco provável que nenhum dos k melhores blocos (aqueles com menores *hashes* de prova) sejam divulgados na rede durante a fase 2. Idealmente, quando todos os k blocos são transmitidos, os nós da rede escolhem aquele que tem o menor *hash* de prova para permanecer em suas visões locais. Entretanto, caso esse bloco não seja transmitido, o bloco

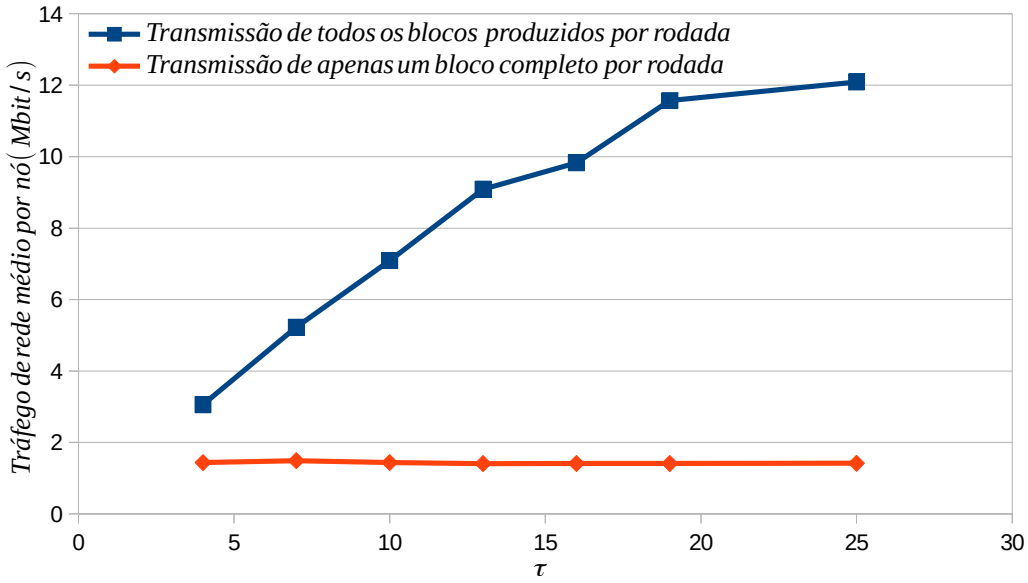

Figura 6.17 – O tráfego de rede é praticamente constante em função τ quando só cabeçalhos de blocos candidatos são transmitidos e apenas o bloco vencedor é transmitido por completo.

com o segundo menor *hash* de prova é o escolhido pelos nós. Neste sentido, os nós sempre procuram manter em suas visões locais o bloco de menor *hash* de prova que tenha sido transmitido na rede durante a fase 2.

Sendo assim, as execuções mostradas anteriormente foram realizadas novamente com os mesmos parâmetros e com a nova premissa de que os k blocos com os menores *hashes* de prova são transmitidos durante a fase 2. Os resultados dessa nova análise são mostrados no gráfico da Fig. 6.18, que foram obtidos por meio da execução do CPoS em uma rede com $N = 25$ nós igualmente distribuídos entre os *hosts* da topologia mostrada na Fig. 6.1, com $\epsilon = 10^{-6}$. Nota-se, a partir do gráfico da Fig. 6.18, que essa proposta também reduz o tráfego de rede por nó quando comparado com a proposta atual (transmissão de todos os blocos), pois o número de blocos completos divulgados na rede é consideravelmente menor que o número de blocos produzidos em uma rodada ($k < \tau$). Além disso, o tráfego também não varia em função de τ para um mesmo número de blocos completos transmitidos (k), como já foi mostrado anteriormente para $k = 1$ (gráfico da Fig. 6.17). Esse tráfego passa a depender mais fortemente do parâmetro k.

Por fim, o gráfico da Fig. 6.19 mostra que, com o aumento do número de blocos completos transmitidos, é possível reduzir a probabilidade de ocorrência de rodadas sem blocos, fato que ocorre quando os nós desonestos produzem todos os k blocos com os menores *hashes* de prova, e não os divulgam. Nota-se, que a partir de $k = 4$, a probabilidade de nenhum dos k blocos com os menores *hashes* de prova serem divulgados é menor que 1% quando o conluio de nós desonestos é capaz de controlar até 30% do *stake* total da rede ($f = 0,3$).

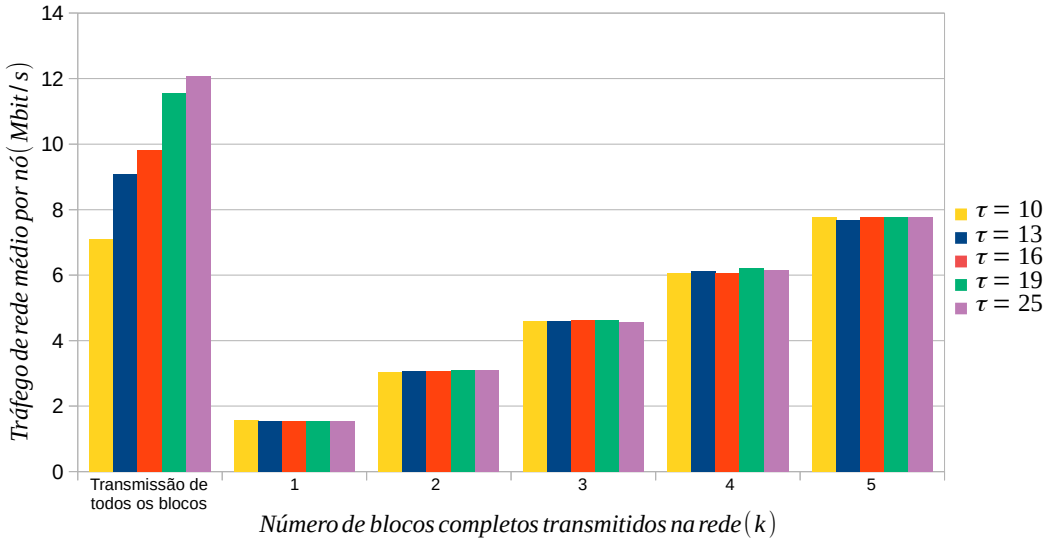

Figura 6.18 – Transmissão dos k blocos com os menores *hash* de prova também pode reduzir o tráfego de rede desde que k seja significativamente menor que τ

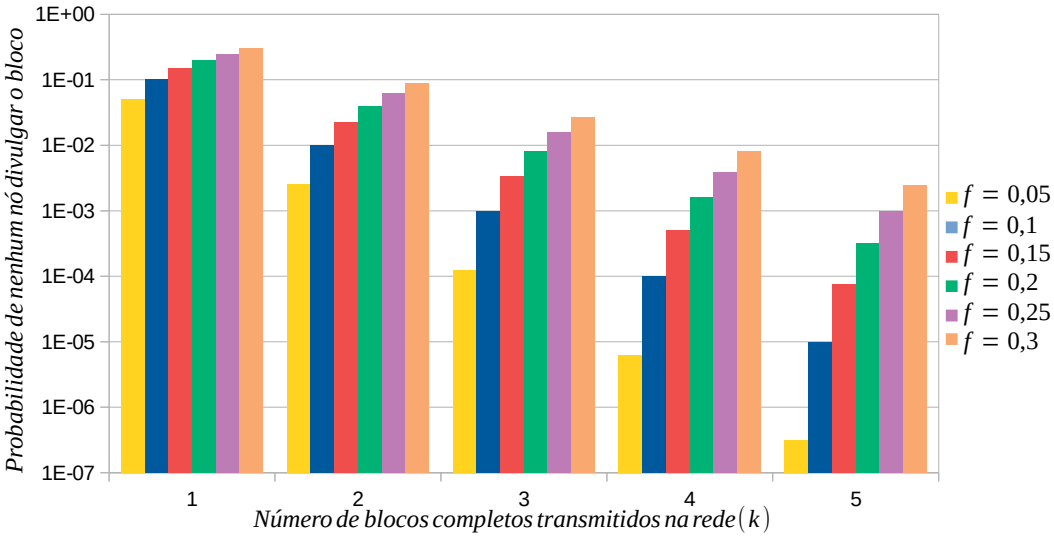

Figura 6.19 – Probabilidade de nenhum bloco ser divulgado diminui com o aumento do número de blocos completos (k) que podem ser transmitidos a cada rodada

6.6 Conclusões

Neste capítulo foi possível validar os modelos teóricos do mecanismo CPoS apresentados no capítulo anterior, através da sua execução em um ambiente descentralizado. Os resultados mostraram que o CPoS é capaz de operar em uma rede descentralizada, mesmo na presença de nós desonestos ou que apresentem latência elevada, o que provoca perda de blocos gerando visões diferentes da blockchain em alguns nós. Nestes cenários adversos, o aumento do valor do parâmetro τ (número esperado de sorteios bem sucedidos por rodada) se mostrou importante, já que desta maneira o protocolo torna-se mais robusto contra oscilações na latência de confirmação dos blocos. Apesar disso, este parâmetro não

pode ser aumentado indefinidamente, já que seu aumento está ligado a um maior tráfego de rede. Neste sentido, foi apresentada uma nova proposta que pode reduzir o tráfego em cada nó, por meio de uma nova estratégia que transmite, em um primeiro momento, apenas os cabeçalhos dos blocos sorteados. Logo depois, apenas o bloco do nó vencedor é transmitido.

A análise inicial mostrou que o objetivo de reduzir o tráfego pode ser alcançado com essa nova técnica, porém novos desafios devem ser tratados. O principal deles está associado à possibilidade do nó vencedor não divulgar o bloco, o que pode dificultar a continuidade do mecanismo em uma nova rodada. Para contornar essa situação, uma proposta intermediária foi apresentada, a qual busca reduzir a possibilidade do nó vencedor não enviar o bloco, através de uma nova política que permite que os k blocos com os menores *hashes* de prova sejam transmitidos por seus criadores.

O número de transações confirmadas por segundo também foi avaliado no protocolo CPoS. Este parâmetro mostrou que este mecanismo probabilístico pode apresentar elevadas taxas de transações confirmadas por segundo, desde que o tempo de duração da rodada seja reduzido.

No próximo capítulo a segurança contra reversões do consenso obtido pelo CPoS é comparada com a obtida pelo protocolo PoW tradicional. Além disso, é apresentada uma comparação preliminar entre a latência de confirmação do CPoS com a do protocolo Casper, ressaltando a possibilidade do CPoS alcançar um tempo de confirmação menor que este último a partir do ajuste adequado de seus parâmetros de controle. Algumas comparações qualitativas com os mecanismos de consenso FPC (Fast Probabilistic Consensus) e Algorand também são apresentadas.

7 Comparações entre o CPoS e outros mecanismos

Neste capítulo será feita uma comparação mais detalhada entre o mecanismo CPoS proposto e outros mecanismos já consolidados na literatura como o Proof-of-Work (PoW), o *Fast Probabilistic Consensus* (FPC), o Casper e o Algorand.

7.1 Proof-of-Work

7.1.1 Segurança

Tschorsch et al. (TSCHORSCH; SCHEUERMANN, 2016) analisa a segurança do PoW do Bitcoin no que tange à sua capacidade de evitar que reversões da cadeia principal ocorram. Uma reversão de cadeia ocorre no PoW quando um grupo de adversários, a partir da soma de seus esforços computacionais, é capaz de produzir um *fork* na blockchain e, em algum momento, a cadeia gerada a partir deste ponto de *fork*, torna-se maior que a cadeia original seguida pelo restante da rede, como mostra a Fig. 7.1.

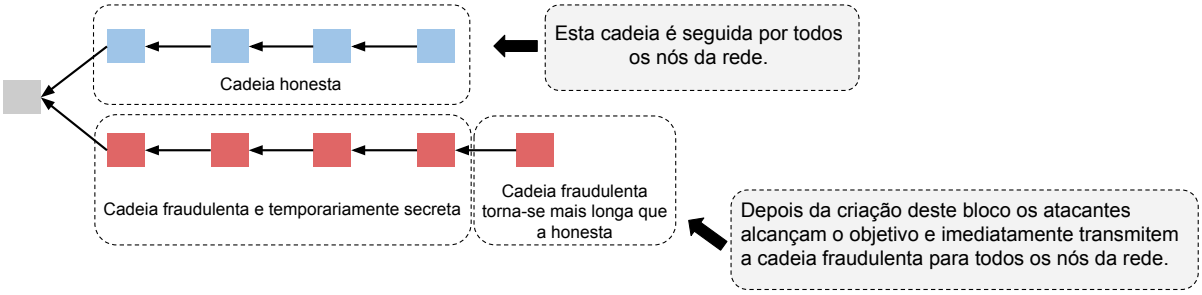

Figura 7.1 – Uma reversão de longo alcance ocorre no PoW quando os adversários produzem uma cadeia mais longa a partir de um *fork*

Quando isso ocorre, todas as transações presentes na cadeia honesta, que estão em blocos a partir do ponto de *fork*, são removidas da blockchain junto com seus blocos, o que caracteriza um reversão bem sucedida na cadeia original. Assim, todas as transações revertidas são invalidadas e os recursos financeiros utilizados a priori como entrada destas transações podem ser utilizados novamente, o que pode caracterizar um ataque conhecido como *double-spending*.

A técnica utilizada pelo PoW para aumentar a segurança contra esse tipo de ataque consiste em esperar até que um determinado bloco receba novas confirmações, ou seja, até que novos sucessores do bloco em questão sejam produzidos para que ele enfim seja considerado confirmado. Essa espera por mais blocos de confirmação aumenta a

segurança, pois os adversários precisarão ser capazes de reverter mais blocos da blockchain para alcançar o bloco em questão, ao ponto que o restante dos nós da rede também estão trabalhando para produzir novos blocos na cadeia honesta.

De acordo com Tschorsch et al., este ataque pode ser modelado por uma distribuição binomial do tipo *random walk*, assumindo que a taxa de *hashes* e, portanto, a dificuldade para produção de um bloco, seja mantida constante. A Eq. 7.1 representa este modelo, onde z_i é a distância em blocos entre a cadeia fraudulenta e a original seguida pela rede na rodada i. No PoW é considerado que os atacantes controlam uma fração f do poder computacional total da rede, o que é equivalente a dizer que f é a probabilidade de eles produzirem um bloco na cadeia que estão construindo. Quando isso ocorre a distância entre as cadeias diminui em uma unidade. Já os nós honestos têm probabilidade $1 - f$ de criar um novo bloco e elevar a distância entre as cadeias em uma unidade.

$$z_{i+1} = \begin{cases} z_i + 1 \text{ } com \text{ } probabilidade \text{ } 1 - f \\ z_i - 1 \text{ } com \text{ } probabilidade \text{ } f \end{cases} \tag{7.1}$$

O interesse deste modelo está em verificar a probabilidade de se obter $z = -1$ em função de f, o que caracteriza uma reversão bem sucedida. É mostrado que se $f > 0,50$, o ataque será bem sucedido em algum momento, independentemente do número de confirmações que o bloco tenha recebido, ou seja, o sucesso passa a não depender do valor de z. Esse ataque é conhecido na literatura como ataque de 51%. A situação pode ser comparada ao problema da ruina de Gambler (William Feller, 1968), que considera a probabilidade de um jogador terminar sem nenhum dinheiro em um dado jogo de cara ou coroa, dado que ele começa com uma quantidade de créditos inicial z. Considerando as probabilidades das possíveis saídas iguais a f e $1 - f$, a probabilidade do jogador terminar sem dinheiro é dada pela Eq. 7.2.

$$q_z = \begin{cases} 1 \text{ } se \text{ } z < 0 \text{ } ou \text{ } f > 0,5 \\ (f/(1-f))^z \text{ } se \text{ } z \geq 0 \text{ } e \text{ } f \leq 0,5 \end{cases} \tag{7.2}$$

A Eq. 7.2 fornece a probabilidade de melhor caso, já que assume que os adversários iniciam o trabalho na cadeia fraudulenta apenas depois que a distância entre elas já está estabelecida (distância z). Todavia, é necessário encontrar a probabilidade de reversão em uma situação mais real, onde os adversário iniciam as tentativas de produzir blocos na cadeia fraudulenta no momento em que o *fork* entre as duas cadeias ocorre, enquanto os nós honestos trabalham para produzir os k blocos necessários para que a confirmação ocorra. Tschorsch et al., considera que os m blocos produzidos pelos adversários na cadeia fraudulenta sigam uma distribuição binomial negativa. Além disso, é assumido que os atacantes já dispõe de um bloco pré minerado (aquele que originou o

fork), consequentemente $z = k - m - 1$. A partir destas considerações, a Eq. 7.3 apresenta a probabilidade geral de reversão no PoW.

$$\begin{cases} 1 \; se \; f \geq 0,5 \\ 1 - \sum_{m=0}^{k} \binom{m+k-1}{m} (f^m(1-f)^k - f^k(1-f)^m) \; se \; f < 0,5 \end{cases} \tag{7.3}$$

Já o gráfico da Fig. 7.2 apresenta a probabilidade de reversão dada pela Eq. 7.3 para diferentes valores de f e k. Nota-se, que a medida que a fração f associada ao poder computacional controlado pelos adversários aumenta, a reversão torna-se mais provável, convergindo para 1 quando $f = 0,5$. Entretanto, quanto mais blocos são aguardados pelos nós honestos para que a confirmação ocorra (aumento do parâmetro k), mais lento é o crescimento da probabilidade de reversão em função de f. Esse resultado reforça que para garantir a segurança do PoW é necessário que sejam criados alguns sucessores para o bloco que se deseja confirmar.

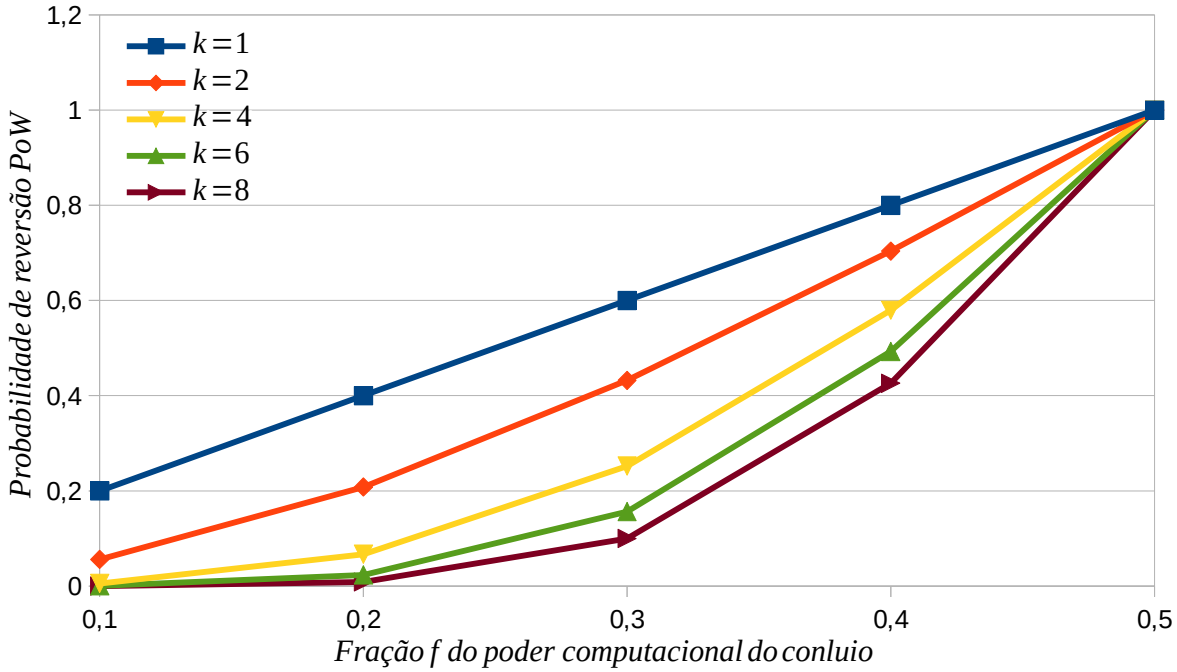

Figura 7.2 – Probabilidade de reversão no PoW converge para 1 a medida que a fração do poder computacional controlado pelo conluio aumenta

Por outro lado, no CPoS a segurança não está alicerçada na dificuldade dos adversários produzirem uma cadeia mais longa que aquela seguida pelos nós honestos, a partir de um ponto de *fork*. A cadeia mais longa no CPoS é pouco relevante, já que o objetivo do mecanismo é garantir probabilisticamente que as confirmações dos blocos não sejam realizadas em conflito com a visão da cadeia correta, ou seja, aquela seguida pelos nós honestos. O ataque começa com um conluio que é capaz de confirmar um bloco de posição (altura) k na mesma rodada (ou menor) em que os nós honestos confirmam um

outro bloco de mesma posição k na cadeia correta. Essa probabilidade já foi analisada na seção 5.3 para diferentes valores de τ e $\epsilon = 10^{-6}$.

A tabela 7.1 mostra uma comparação direta entre a probabilidade de ocorrer uma reversão na cadeia honesta no PoW e a probabilidade de blocos conflitantes serem confirmados no CPoS. A comparação é feita para $\tau = 10$ e $\epsilon = 10^{-6}$, e o parâmetro f tem

Tabela 7.1 – Comparação entre a probabilidade de uma reversão no PoW com uma confirmação conflitante no CPoS

f	k e Δ_r	Prob. reversão no PoW	Prob. conflito no CPoS
$0,1$	1	$0,2$	$2,75 \times 10^{-13}$
$0,1$	2	$0,056$	$1,60 \times 10^{-15}$
$0,1$	4	$0,0055$	$1,54 \times 10^{-15}$
$0,1$	6	$0,0006$	$1,33 \times 10^{-15}$
$0,2$	1	$0,4$	$3,72 \times 10^{-9}$
$0,2$	2	$0,208$	$9,26 \times 10^{-12}$
$0,2$	4	$0,0667$	$6,53 \times 10^{15}$
$0,2$	6	$0,0233$	$3,99 \times 10^{-15}$
$0,3$	1	$0,6$	$6,55 \times 10^{-7}$
$0,3$	2	$0,4320$	$2,40 \times 10^{-8}$
$0,3$	4	$0,2521$	$1,43 \times 10^{-10}$
$0,3$	6	$0,1564$	$1,03 \times 10^{-10}$
$0,4$	1	$0,8000$	$1,97 \times 10^{-5}$
$0,4$	2	$0,7040$	$3,67 \times 10^{-6}$
$0,4$	4	$0,5796$	$3,22 \times 10^{-7}$
$0,4$	6	$0,4930$	$2,18 \times 10^{-7}$
$0,5$	1	1	$2,24 \times 10^{-4}$
$0,5$	2	1	$1,19 \times 10^{-4}$
$0,5$	4	1	$5,27 \times 10^{-5}$
$0,5$	6	1	$1,14 \times 10^{-5}$

relação com os dois mecanismos. No PoW ele indica a razão entre o poder computacional controlado pelo conluio e toda a capacidade computacional da rede. Já no CPoS ele indica a razão entre o *stake* controlado pelo conluio e o total de *stake* disponível no período. De acordo com a tabela, é possível notar claramente que a probabilidade de uma reversão ocorrer no PoW é bem mais elevada que a probabilidade do CPoS confirmar um bloco de maneira conflitante.

Apesar da diferença existente entre os dois mecanismos no que tange à confirmação de blocos, a reversão de blocos no PoW e a confirmação de blocos conflitantes no CPoS são situações que levam as transações dos blocos envolvidos a um conflito e, por esse motivo, a comparação é válida. Pelos dados da tabela, é possível afirmar que o CPoS apresenta menos risco de ter um ataque iniciado. Além disso, a tabela mostra que a probabilidade de conflito em confirmações no CPoS, para um mesmo valor de f, diminui à medida que aumenta o intervalo de rodadas (Δ_r). Isso ocorre porque, à medida que o número de rodadas aumenta, a média de blocos recebidos em cada rodada se aproxima ainda mais de τ, o que diminui a probabilidade de ocorrência de sorteios bem sucedidos suficientes para confirmar blocos em cadeias conflitantes.

Sendo assim, caso seja de interesse elevar a segurança do CPoS contra confirmações conflitantes para além dos valores apresentados na tabela, é recomendado aguardar mais algumas rodadas após a confirmação do bloco. A partir dessa espera é possível dizer que a probabilidade de um bloco confirmado em uma determinada visão ser revertido por um bloco confirmado em outra visão, diminui. Essa espera é importante, principalmente, para fazer com que possíveis nós honestos que foram levados a produzir blocos em uma visão desonesta percebam que devem trocar de visão, por meio de uma ressincronização. A existência de nós honestos que contribuem com a evolução de uma cadeia desonesta pode ocorrer em um determinado momento caso tais nós honestos percam a sincronização com a visão honesta seguida pelo restante da rede, sendo, portanto, levados a acreditar na visão produzida pelos atacantes.

Essa situação não será mantida por muitas rodadas no caso em que a fração do *stake* controlada pelos atacantes seja minoria ($f < 0,5$), já que os nós honestos perceberão a existência de uma visão melhor assim que novos pedidos de ressincronização ocorrerem, ou seja, quando a média calculada na visão atual (visão do atacante) ficar abaixo do limiar de ressincronização $\alpha^{(x)}$. Entretanto, quando os atacantes dispõem da maioria do *stake* ($f > 0,5$), situação não desejada e nem esperada na prática (como nos demais mecanismos de consenso), a visão deles passa a ser a principal e, ao longo das próximas rodadas, o restante da rede irá reverter para tal visão, a partir de novos pedidos de ressincronização.

7.1.2 Desempenho energético

O CPoS não depende de um alto esforço computacional em sua operação, o que permite que seu consenso seja alcançado a partir de um menor consumo de energia elétrica, como será mostrado nesta seção por meio de comparações com o PoW do Bitcoin.

Os mineradores do Bitcoin têm uma capacidade agregada de $153,83 \times 10^{18}$ *hashes* por segundo com 9972 nós (COIN.DANCE, 2021). Desta forma, em uma primeira comparação busca-se estimar a taxa de *hashes* calculados pelo CPoS em uma rede com o mesmo número de nós, utilizando $T = 14$ segundos. Esse tamanho de rodada facilita a comparação com o mecanismo Casper, pois ele utiliza esse mesmo valor, como será mostrado mais adiante. O CPoS calcula *hashes* em três fases distintas de sua operação. Assim, cada fase e seus respectivos volumes de dados são definidos a seguir.

- **Fase 1:** destina-se ao sorteio criptográfico da rodada r, onde cada nó i calcula o *hash* do nó $U_i^{(r)}$ e o *hash* de sua assinatura $H(\sigma_i^{(r)})$.
- **Fase 2:** destina-se à transmissão dos blocos da rodada r, onde apenas os nós sorteados calculam seus *hashes* de prova ($H_t^{(r)}$). Nesta fase é assumido que os nós sorteados criam apenas um *hash* de prova, ou seja, são sorteados apenas uma vez. Isso é

razoável para uma rede com um grande *stake* total e que está bem distribuído, fatos que são esperados na prática.

- **Fase 3:** destina-se a verificação dos blocos transmitidos na rodada r, onde é esperado que cada nó i calcule τ *hashes* do nó e τ *hashes* de prova. Em média τ blocos são transmitidos na rede.

A tabela 7.2 mostra essa taxa em cada uma das fases de produção do mecanismo, considerando dois valores diferentes para τ. Na Fase 1, cada um dos 9.972 nós calcula 2

Tabela 7.2 – Número de *hashes* por segundo calculados pelo CPoS em uma rede com 9.972 nós.

τ	Fase 1 (H/s)	Fase 2 (H/s)	Fase 3 (H/s)
10	$1.424,57$	$0,71$	$14.245,71$
100	$1.424,57$	$7,14$	$142.457,14$

hashes por rodada. Logo, para uma rodada com duração $T = 14s$, temos $1.424,57$ *hashes* por segundo. A Fase 2 depende do valor τ, já que computa os *hashes* calculados durante a fase de envio dos blocos. Sendo assim, é necessário considerar todos os *hashes* de prova dos nós que foram sorteados na rodada. Como na média o número de nós sorteados é τ e foi assumido que cada nó é sorteado apenas uma vez, a taxa de *hashes* da Fase 2 é dada por τ/T. Por fim, na Fase 3, todos os nós verificam os blocos transmitidos na rodada. Cada nó deve calcular τ *hashes* de prova e τ *hashes* do nó e, portanto, a taxa de *hashes* dessa fase é dada por $9.972 \times 2 \times \tau/T$. A taxa de *hashes* por segundo calculada por um mecanismo de consenso é diretamente proporcional ao consumo de energia elétrica dos nós que participam desse consenso. Portanto, a partir desta primeira análise, é possível notar que para $\tau = 10$, o CPoS exigiria na ordem de 10^{14} menos esforço computacional e consumo de energia elétrica, mesmo quando comparado com a fase de cálculo mais intenso (Fase 3).

A segunda análise supõe como condição hipotética que o tamanho da rede do CPoS será agora equivalente ao tamanho atual da rede do Bitcoin, mas considerando que todos os nós que hoje formam os pools de mineração estão descentralizados, ou seja, cada um dos nós está empregando seus esforços computacionais de maneira individual. Assim, tomando como base que o pool *SlushPool* tem aproximadamente 144.000 participantes e representa $4,5\%$ do poder de mineração de toda a rede do Bitcoin (SLUSHPOOL.COM, 2021), é possível extrapolar este fato para estimar que a rede do Bitcoin formada pelos 9.972 nós da primeira análise é equivalente a $3,2 \times 10^6$ nós individuais. A tabela 7.3 mostra a nova taxa de *hashes* calculados por segundo pelos nós do CPoS, utilizando exatamente o número de nós encontrados por meio da extrapolação. Nota-se a partir da tabela que, apesar do aumento no número de *hashes* calculados por segundo nesta nova configuração

Tabela 7.3 – Número de *hashes* por segundo calculados pelo CPoS em uma rede com $3,2 \times 10^6$ nós.

τ	Fase 1 (H/s)	Fase 2 (H/s)	Fase 3 (H/s)
10	$457.142,86$	$0,71$	$4,57 \times 10^6$
100	$457.142,86$	$7,14$	$4,57 \times 10^7$

de rede, ainda assim o CPoS exige 10^{12} menos esforço computacional e consumo de energia elétrica quando comparado ao PoW do Bitcoin para $\tau = 10$.

É desejável que o CPoS tenha um grande número de nós e um *stake* bem distribuído entre eles para dificultar conluios e, consequentemente, ataques. Porém, diferentemente do PoW do Bitcoin, o CPoS não necessita de um número gigantesco de nós para operar satisfatoriamente e não incentiva uma corrida por taxas de cálculo de *hashes* cada vez maiores.

7.2 Fast Probabilistic Consensus (FPC)

Um participante no mecanismo FPC depende da opinião emitida por um conjunto aleatório de *peers* para formar a sua própria opinião a respeito do valor de um determinado bit. Capossele et al. (CAPOSSELE; MUELLER; PENZKOFER., 2019) define a taxa de integridade do FPC como uma medida da capacidade do mecanismo manter a opinião inicial da maioria dos nós honestos ao final do consenso. Segundo estes autores, esta taxa é dependente da proporção f de nós desonestos, onde para $f = 0,2$ a taxa de integridade é no máximo igual a $0,6$, o que ocorre próximo a um número de rodadas $l = 5$.

Esse resultado mostra que os adversários conseguem influenciar os nós honestos emitindo opiniões contrárias àquela que é dada pela maioria dos nós no início do consenso. Por outro lado, no FPC o consenso em direção ao valor correto do bit ocorre quando a opinião inicial emitida pela maioria simples dos nós (51%), no início do consenso, é igual à opinião de todos os nós honestos ao final do consenso, fato que não ocorre quando há falhas de integridade. Portanto, na presença destas falhas, os nós honestos são conduzidos pelos atacantes a uma nova opinião e, quando o consenso é finalizado, a opinião do atacante terá prevalecido.

Já no CPoS os adversários foram analisados a partir da formação de um conluio que controla uma fração f do *stake* total disponível na rede em um determinado período. Neste sentido, apesar dos adversários atrasarem o consenso, a Seção 5.3 mostra que, para $f \leq 0,3$, a probabilidade do conluio confirmar um bloco conflitante é baixa e está próxima do limiar de segurança ϵ estabelecido. Porém, quando f cresce, é esperado um aumento da influência do conluio no crescimento da cadeia correta e, quando ele detém a maioria do *stake*, sua cadeia receberá a maioria dos sorteios bem sucedidos e os nós honestos

reverterão suas visões para a nova cadeia proposta.

7.3 Casper

7.3.1 Segurança

O Casper utiliza um complexo esquema de punições com o objetivo de evitar que o grupo de validadores realize atividades contrárias àquelas determinadas pelo mecanismo. Essas punições garantem o aumento da confiabilidade e do desempenho do consenso, já que os validadores são impactados por perdas financeiras, caso alguma regra do mecanismo seja violada. Entretanto, apesar de possível, não é garantido pelo mecanismo que os membros do conjunto de validadores sejam alterados dentro de um período pré definido. Este fato colabora para que estes membros sejam publicamente conhecidos durante um intervalo de tempo suficientemente grande, permitindo assim que tentativas de ataques e de subornos sejam direcionadas a eles, com o objetivo de dificultar ou direcionar a convergência dos votos para um certo valor. Como consequência, o desempenho do mecanismo pode ser reduzido, principalmente quando os validadores que são donos de grandes quantidades de *stake* são afetados por essas tentativas de ataques. Quando esses validadores são impedidos de distribuir seus votos na rede, torna-se mais difícil alcançar pelo menos 2/3 do *stake* total para garantir a convergência em direção a um mesmo valor.

Já o CPoS não conta com esquemas de punições e tampouco com grupos de validadores, já que não implementa um comitê de confirmação descentralizado. Ao invés disso, a convergência do consenso e a segurança contra conflitos são garantidos por meio da análise local que qualquer nó faz em relação a sua visão atual da blockchain. Essa análise é realizada a partir da média de sorteios bem sucedidos que é calculada através dos blocos recebidos a cada rodada. Essa média e o fato do número de nós sorteados a cada rodada ser controlado pelo parâmetro τ, são suficientes para que os nós decidam se suas visões locais são corretas, ou seja, se são seguidas pelos outros nós da rede. Comparando com o Casper, o CPoS oferece uma forma mais simples para confirmar os blocos, pois esta decisão é tomada localmente pelo nó, sem que seja necessária a implementação de uma fase destinada à votação. Portanto, mesmo na presença de adversários, não é necessário utilizar um esquema de punições no CPoS haja vista que as decisões de outros nós não afetam diretamente a convergência do consenso probabilístico, que é realizada de maneira descentralizada pelos nós.

O esquema de sorteios do CPoS é parcialmente baseado no mecanismo Algorand e, por isso, é esperado que os nós sorteados para produção de novos blocos sejam diferentes a cada rodada, o que dificulta as tentativas de ataques direcionados a estes nós da forma como pode ocorrer no Casper. Além disso, os nós sorteados em uma rodada não são conhecidos até que os blocos criados por eles sejam divulgados na rede, preservando

assim a identidade dos mesmos. Esse fato evita que conluios sejam formados, a partir do conhecimento de quais nós serão sorteados em rodadas futuras.

7.3.2 Latência de confirmação

A latência de confirmação dos blocos no Casper depende da velocidade com que os *checkpoints* são finalizados. A Fig. 7.3 mostra o melhor caso do mecanismo Casper que ocorre quando a maioria dos nós compartilham a mesma visão da blockchain, a partir de uma forte sincronização entre os nós. A figura mostra que nestas condições, o Casper utiliza dois períodos para que um determinado *checkpoint* possa ser confirmado.

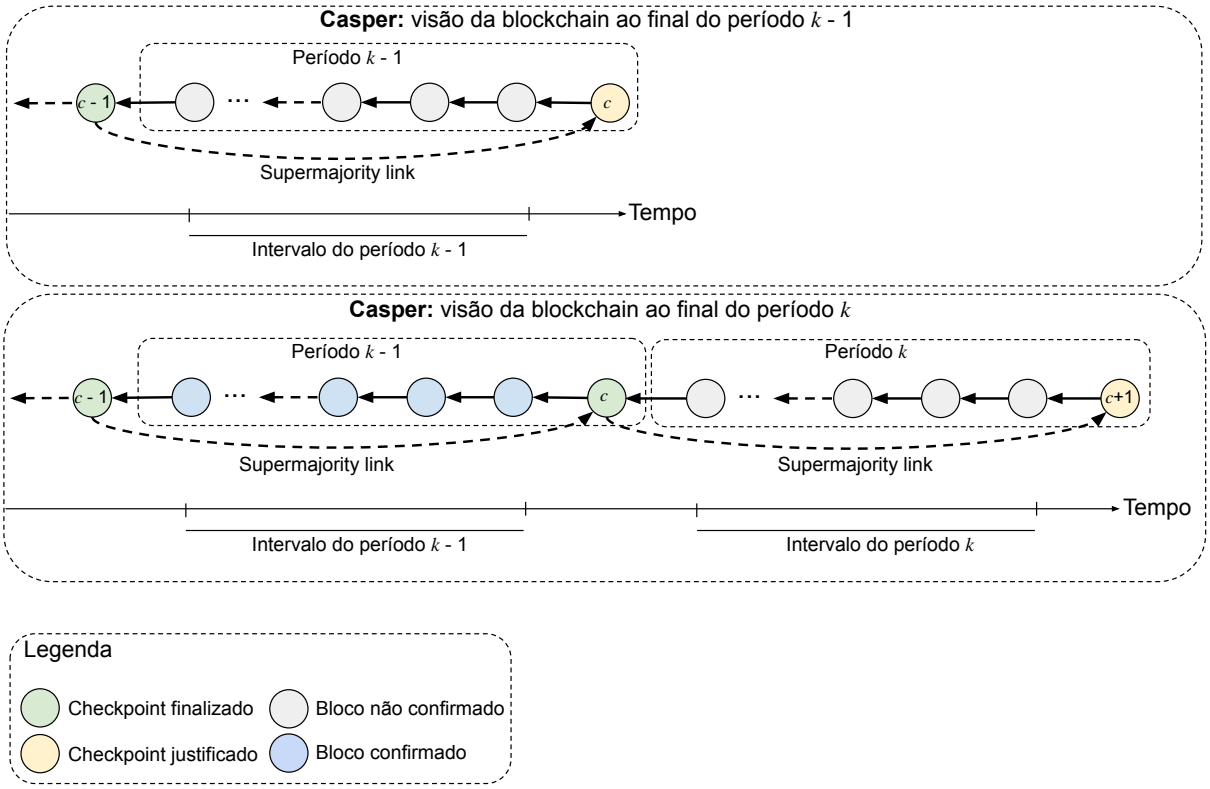

Figura 7.3 – Casper utiliza dois períodos para finalizar um *checkpoint* no melhor caso

Ao final do período $k-1$, os validadores enviam votos na direção do link $(c-1) \rightarrow c$, com o objetivo de justificar o próximo *ckeckpoint* da blockchain (*checkpoint* c). Como a análise é de melhor caso, os votos emitidos convergem para este único link, alcançando assim a condição limite para torná-lo supermajoritário (2/3 dos votos), o que é suficiente para justificar o *checkpoint* c. O segundo período é utilizado para finalizar o checkpoint c por meio da emissão de novos votos, para que um novo link supermajoritário com origem em c seja formado $(c \rightarrow (c+1))$. Este novo link é a condição necessária para que o *checkpoint* c seja confirmado, o que promove a confirmação de todos os blocos do período $k-1$.

Neste sentido, a latência de confirmação no Casper depende da posição do bloco dentro do período, ou seja, blocos produzidos no início do período apresentam uma latência de confirmação maior, enquanto aqueles blocos produzidos mais ao final do período são confirmados mais rapidamente. Assim, a latência de confirmação L_i referente ao i-ésimo bloco produzido dentro do período é dada pela Eq. 7.4.

$$L_i = ((Epoch - i) + Epoch) \times T \quad (1 \leq i \leq Epoch) \tag{7.4}$$

Os parâmetros usados na Eq. 7.4 são definidos a seguir.

- L_i: latência de confirmação do i-ésimo bloco do período;
- $Epoch$: número de blocos em um período;
- T: tempo de rodada (Casper);

Na prática o Casper utiliza períodos com 50 blocos ($Epoch = 50$) (BV, 2018), onde no melhor caso é possível assumir que estes blocos são produzidos em 50 rodadas. A literatura indica ainda que os validadores do Casper são capazes de propor um novo bloco a cada 14 segundos, ou seja, $T = 14$ s (BV, 2018). Nestas condições, é possível calcular a latência média ($\bar{L}$) para que um bloco seja confirmado no Casper. A tabela 7.4 apresenta uma comparação entre a latência esperada para o CPoS (R_c) e a latência média observada no Casper nas condições descritas acima, utilizando $T = 14$ s para os dois mecanismos. A

Tabela 7.4 – Latência de confirmação esperada no CPoS é menor quando comparada a do Casper.

τ	ϵ	R_c	$R_c(segundos)$	$\bar{L}(segundos)$	$R_c/\bar{L}$ (%)
10	10^{-6}	$2,96$	$41,44$	1043	$3,97$
13	10^{-6}	$2,65$	$37,10$	1043	$3,56$
16	10^{-6}	$2,25$	$31,50$	1043	$3,02$
19	10^{-6}	$2,02$	$28,28$	1043	$2,71$
25	10^{-6}	$1,75$	$24,50$	1043	$2,35$

tabela mostra que o CPoS oferece uma grande redução na latência de confirmação quando comparado ao Casper. Vale ressaltar ainda que, uma vez calibrados os parâmetros τ e ϵ, a latência esperada R_c do CPoS passa a ser a mesma para qualquer bloco produzido, fato que também pode ser considerado uma vantagem significativa em relação ao Casper, haja vista que mesmo no seu melhor caso, o Casper ainda produz blocos com tempo de confirmação maior que a média $\bar{L}$ apresentada.

Por outro lado, considerando que o tamanho médio das transações e do bloco são constantes nos dois mecanismos, pode-se dizer que o número de transações confirmadas por segundo passa a depender apenas do tamanho da rodada de produção (T). Sendo

assim, se for considerada apenas a primeira abordagem do CPoS, onde todos os blocos produzidos são transmitidos na rede com todas as suas respectivas transações, mostrou-se não ser possível adotar $T = 14s$ no CPoS, pois o tráfego de rede gerado em cada nó exige uma rodada mais longa. Como o Casper produz apenas um bloco em cada rodada, ele pode utilizar uma rodada menor, já que apenas um bloco deve ser transmitido entre todos os validadores em cada rodada. Nestas condições, é possível dizer que o número de transações confirmadas por segundo do Casper é mais alto que o do CPoS. Porém, assumindo a proposta alternativa do CPoS, onde é possível transmitir apenas os k blocos associados aos melhores *hashes* de prova, pode-se reduzir o seu tráfego de rede, o que reduz o tempo de trasmissão dos blocos, permitindo uma redução no tamanho da rodada. Com essa nova premissa, o CPoS pode produzir um número de transações confirmadas por segundo próximo ao do Casper, mas com uma latência bem menor como mostrado na Tabela 7.4.

Por fim, segundo estudos realizados por Moindrot et al. (MOINDROT; BOURNHONESQUE, 2017), quando no Casper a fração de validadores desconectados da rede se aproxima de 1/3 do *stake* total da rede, o número de *checkpoints* finalizados e de blocos confirmados por segundo decresce rapidamente para zero. No CPoS também é observada uma redução no número de blocos confirmados por segundo devido a um aumento na latência de confirmação, quando uma fração $f > 0$ de participantes deixa de contribuir com a visão seguida pela rede. Mas, ainda assim, o mecanismo continua a confirmação de blocos na cadeia correta mesmo que o conluio controle 1/3 do *stake* ($f \leq 0,33$). O CPoS só irá reduzir seu número de blocos confirmados por segundo de maneira mais significativa quando a fração f se aproximar de $0,5$. Entretanto esta redução pode ser atenuada com o aumento do valor de τ, como mostra a Seção 6.3. A capacidade de continuar a confirmação de blocos está associada ao valor de τ, haja vista que o aumento deste parâmetro melhora a margem de confirmação $M^{(x)}$, o que colabora para que o CPoS seja mais robusto às ações dos adversários.

7.4 Algorand

A principal diferença entre o CPoS e o Algorand está em como cada um deles confirma os blocos. O Algorand é um mecanismo de consenso imediato, baseado em um esquema de votação descentralizado. Os votos enviados pelo comitê são utilizados para que os nós dá rede possam confirmar o próximo bloco da blockchain, algo que ocorre de maneira determinística sempre que pelo menos 2/3 dos votos são enviados na direção de um único bloco. Já o CPoS é um mecanismo que utiliza um algoritmo de sorteio criptográfico derivado do Algorand; entretanto, este mecanismo não utiliza comitê de validação para que seus blocos sejam confirmados. Ao invés disso, os critérios estabelecidos definem uma forma para que cada nó possa mensurar a sua visão local e, com isso, confirmar novos

blocos ou solicitar uma ressincronização de cadeia. Essa novidade traz ao CPoS um maior controle da sua latência de confirmação, o que o torna menos sujeito a ataques e aumenta sua eficiente, já que não requer nenhum esquema para criar, manter, gerenciar e eliminar grupos de confirmação.

7.5 Conclusões

A primeira comparação mostrou que o mecanismo CPoS oferece maior segurança contra problemas ligados a confirmações conflitantes ou reversões quando comparado com o PoW. Esta melhor segurança é alcançada a partir da utilização de um novo critério de evolução da blockchain, o qual não está amparado no tamanho da cadeia, mas sim na visão atual que cada nó tem da blockchain, visão esta que pode ser mensurada a partir do número de sorteios recebidos em cada rodada. Também foi mostrado através de duas análises distintas, que o CPoS requer menos poder computacional e, consequentemente, consome muito menos energia elétrica quando comparado ao PoW.

O FPC também é probabilístico, mas depende da opinião inicial da maioria dos nós, a qual deve ser mantida até o final do consenso. Caso contrário, o FPC pode apresentar uma falha de integridade. O CPoS é um mecanismo que não depende da opinião inicial dos nós. Assim, é possível afirmar que um bloco confirmado no CPoS é sempre válido, desde que nós adversários não tenham uma fração significativa do stake capaz de causar um conflito durante o consenso.

O CPoS também foi comparado com o mecanismo Casper no que tange o seu desempenho e segurança. Nesta comparação, destacou-se alguns pontos qualitativos, principalmente relacionados a uma maior simplicidade do CPoS nas confirmações de blocos, já que ele é um mecanismo que não utiliza comitês de validação. Além disso, foi mostrado que o mecanismo é capaz de oferecer uma latência de confirmação consideravelmente menor que a do Casper e constante para todo bloco produzido.

8 Conclusões

O consenso baseado em prova que utiliza o *stake* como forma de controlar as ações dos participantes apresenta características desejáveis para as blockchains públicas. Entre as principais características estão a possibilidade de se obter um desempenho melhor, além de reduzir o consumo de energia elétrica, quando comparado ao *Proof-of-Work*. Em linhas gerais, os mecanismos tolerantes a falhas bizantinas (BFTs) que implementam comitês de validação controlados pelo *stake* estão entre os que apresentam melhores desempenhos. Entretanto, a operação destes mecanismos é em geral complexa, já que é necessário garantir a convergência de um esquema de votação, para que a confirmação dos blocos ocorra.

O CPoS foi proposto como uma alternativa aos mecanismos baseados em Proof-of-Stake que utilizam comitês de confirmação descentralizados. Como não implementa esquemas de confirmação de blocos por meio de comitês em nenhuma de suas fases, a operação do CPoS torna-se mais simples. Por outro lado, ele exige que o *stake* total W de cada período seja precisamente calculado, evitando distorções na definição da probabilidade p de ocorrer um sorteio bem sucedido. Além disso, o *stake* do usuário não pode ser movimentado até o final do período atual, o que contribui para que a probabilidade de sucesso seja mantida constante no período.

O CPoS é um mecanismo que oferece parâmetros configuráveis, que influenciam diretamente a latência de confirmação e a segurança, já que o parâmetro de segurança ϵ permite definir o nível de segurança para uma aplicação específica e, consequentemente, tem-se o tempo de confirmação correspondente. Por outro lado, é possível reduzir o tempo de confirmação mantendo o nível de segurança constante, mas aumentando o número médio de blocos produzidos a cada rodada (parâmetro τ). A partir da alteração destes parâmetros, o CPoS mostrou-se capaz de oferecer uma latência de confirmação menor que a do mecanismo Casper, o que mostra a viabilidade de seu consenso sem comitê de confirmação e a importância de ter um consenso mais ajustável.

Resultados experimentais mostraram que o CPoS pode confirmar entre 40 e 130 transações por segundo, valor uma ordem de magnitude superior ao do mecanismo PoW utilizado no Bitcoin, por exemplo. Além disso, foi possível mostrar por meio de uma comparação teórica entre CPoS e PoW, que o critério contra confirmações conflitantes utilizado pelo CPoS garante maior segurança às transações já confirmadas.

Os experimentos também mostraram que o CPoS é tolerante a falhas, já que o consenso foi alcançado em cenários com perda de blocos e presença de adversários que controlam menos de 50% do *stake* total. Com relação aos adversários, mostrou-se que a

probabilidade de eles confirmarem um bloco conflitante é baixa; porém o conluio pode dificultar a convergência do consenso. Ainda sim, mostrou-se que é possível aumentar a robustez do mecanismo contra esse tipo de ataque, utilizando valores de τ mais elevados.

Por fim, o CPoS é um mecanismo mais simples que aqueles que utilizam comitês de confirmação, já que não precisa criar e gerenciar tais comitês e não possui sistemas de punições como parte de seu consenso. Ao invés disso, os nós são capazes de confirmar blocos, a partir do fortalecimento de suas visões locais, o que ocorre quando novos blocos que estendem a visão da blockchain local são recebidos.

Trabalhos Futuros

É necessário implementar uma política de distribuição de recompensas para os participantes do consenso, o que permitiria que os nós fossem incentivados a participar, aumentando a descentralização da rede e com isso a segurança do mecanismo, como descrito no Capítulo 4. Tal implementação poderia ser utilizada como prova de conceito para avaliar a capacidade de descentralização desta nova política, quando comparada com outros mecanismos da literatura. Neste sentido, a distribuição proporcional da recompensa entre todos os nós que depositaram algum *stake* dentro de um período específico é uma opção promissora, pois garantiria que nós com poucos recursos financeiros recebessem recompensas apenas por participar do consenso, mesmo que não fossem sorteados para produzir os blocos. Além disso, também poderia haver uma segunda recompensa associada à produção dos blocos. Essa recompensa teria a função de estimular a participação no sorteio criptográfico a cada rodada, o que facilitaria a manutenção do número de blocos produzidos próximo de τ.

Assim, uma nova implementação deve garantir que todos os nós que participaram do consenso durante o período E sejam recompensados de maneira proporcional à quantidade de *stake* depositada por meio de transações confirmadas no período $E-1$ ou antes. Essas recompensas podem ser solicitadas pelos nós a partir do início do período $E+1$, por meio da criação de novas transações que indicam o valor da recompensa de maneira proporcional ao valor depositado no período $E-1$. Os nós da rede que recebem em seus *pools* essas transações de resgate podem atestar a validade do pedido, a partir de consultas ao histórico de transações que foram confirmadas no período $E-1$. Deste modo, é possível confirmar se, de fato, o dono do endereço em questão reservou a quantidade de *stake* informada.

É preciso realizar também uma análise de segurança mais aprofundada no CPoS, onde seria avaliada a sua capacidade em manter uma única visão da blockchain em função do número de *peers* que os nós mantêm durante a execução. Essa análise poderia trazer conclusões sobre a probabilidade de um ataque do tipo *Eclipse* ocorrer à medida

que o número de *peers* definidos aleatoriamente muda. Além disso, é preciso realizar uma elaboração mais completa da prova de segurança do mecanismo de consenso proposto.

Por fim, é preciso otimizar processos que não estão diretamente associados às políticas de consenso, mas que influenciam no desempenho do mecanismo. Essas questões envolvem, por exemplo, a implementação de um mecanismo eficiente para verificação e criação de novos blocos, o que otimiza a validação das transações destes blocos e, consequentemente, o seu encaminhamento na rede.

Referências

ANTONOPOULOS, A. M. *Mastering Bitcoin: Programming the Open Blockchain*. 2. ed. [S.l.]: O'Reilly, 2017. 615 p. ISBN 978-1491954386. Citado 3 vezes nas páginas 23, 28 e 29.

ARATANI, L. *Electricity needed to mine bitcoin is more than used by 'entire countries'*. 2021. (acessado em 10/04/2021). Disponível em: <https://www.theguardian.com/technology/2021/feb/27/bitcoin-mining-electricity-use-environmental-impact>. Citado na página 33.

AUBLIN, P. L.; MOKHTAR, S. B.; QUEMA, V. RBFT: Redundant byzantine fault tolerance. In: *Proceedings - International Conference on Distributed Computing Systems*. Washington,DC, United States: IEEE Computer Society, 2013. p. 297–306. ISBN 9780769550008. Disponível em: <https://ieeexplore.ieee.org/document/6681599>. Citado na página 23.

BASHIR, I. *Mastering Blockchain Second Edition*. 2. ed. Packt Publishing, 2018. 647 p. ISBN 978-1-78883-904-4. Disponível em: <www.packtpub.com>. Citado 2 vezes nas páginas 17 e 25.

BAXENDALE, G. CAN BLOCKCHAIN REVOLUTIONISE EPRs? *Itnow*, v. 58, n. 1, p. 38–39, 2016. ISSN 17465710. Disponível em: <https://ieeexplore.ieee.org/document/8138652>. Citado na página 21.

BEN-OR, M. Another advantage of free choice (Extended Abstract). In: *Proceedings of the Second Annual ACM Symposium on Principles of Distributed Computing*. New York,NY, United States: Association for Computing Machinery, 1983. p. 27–30. ISBN 0897911105. Disponível em: <https://dl.acm.org/doi/10.1145/800221.806707>. Citado na página 27.

BENTOV, I. et al. Proof of Activity. *ACM SIGMETRICS Performance Evaluation Review*, v. 42, n. 3, p. 34–37, 2014. ISSN 0163-5999. Disponível em: <https://dl.acm.org/doi/10.1145/2695533.2695545>. Citado 2 vezes nas páginas 29 e 47.

BISSIAS, G. et al. An Analysis of Attacks on Blockchain Consensus. *ArXiv*, 2016. (acessado em 05/06/2021). Disponível em: <http://arxiv.org/abs/1610.07985>. Citado na página 76.

BLOCKCHAIN.INFO. *Hashrate Distribution*. 2020. 1–2 p. (acessado em 10/04/2021). Disponível em: <https://blockchain.info/de/pools>. Citado 2 vezes nas páginas 33 e 109.

BLOOM, B. H. Space/time trade-offs in hash coding with allowable errors. *Communications of the ACM*, v. 13, n. 7, p. 422–426, 1970. ISSN 15577317. Disponível em: <https://dl.acm.org/doi/10.1145/362686.362692>. Citado na página 154.

BUTERIN, V.; GRIFFITH, V. Casper the friendly finality gadget. *arXiv*, 2017. ISSN 23318422. Disponível em: <http://arxiv.org/abs/1710.09437>. Citado 3 vezes nas páginas 43, 67 e 69.

BV, S. . C. *Consensus, Casper and Cryptoeconomics in 15 minutes or less*. 2018. (acessado em 08/03/2021). Disponível em: <https://medium.com/@jpa_of_snc/

consensus-casper-and-cryptoeconomics-in-15-minutes-or-less-c7ca2427bf88>. Citado na página 127.

CACHIN, C. *Architecture of the Hyperledger Blockchain Fabric*. Rüschlikon, Zürich,Switzerland, 2016. (acessado em 10/12/2020). Disponível em: <http://bytacoin.io/main/Hyperledger.pdf>. Citado na página 23.

CAPOSSELE, A.; MUELLER, S.; PENZKOFER., A. Robustness and efficiency of leaderless probabilistic consensus protocols within byzantine infrastructures. *arXiv*, 2019. ISSN 23318422. Disponível em: <http://arxiv.org/abs/1911.08787>. Citado na página 124.

CASTRO, M.; LISKOV, B. Practical Byzantine Fault Tolerance and Proactive Recovery. *ACM Transactions on Computer Systems*, v. 20, n. 4, p. 398–461, 2002. ISSN 07342071. Disponível em: <https://dl.acm.org/doi/10.1145/571637.571640>. Citado na página 95.

Certicom Research. *SEC2: Recommended Elliptic Curve Domain Parameters (Version 1.0)*. Mississauga, ON, Canada, 2010. 3–18 p. Disponível em: <http://www.secg.org/sec2-v2.pdf>. Citado na página 24.

CHEN, L. et al. On security analysis of proof-of-elapsed-time (PoET). In: *Stabilization, Safety, and Security of Distributed Systems*. Boston, MA, United States: Springer International Publishing, 2017. p. 282–297. ISBN 978-3-319-69084-1. ISSN 16113349. Disponível em: <http://link.springer.com/10.1007/978-3-319-69084-1_19>. Citado 3 vezes nas páginas 29, 46 e 47.

CHRISTIDIS, K.; DEVETSIKIOTIS, M. Blockchains and Smart Contracts for the Internet of Things. *IEEE Access*, v. 4, p. 2292–2303, 2016. ISSN 21693536. Disponível em: <http://ieeexplore.ieee.org/document/7467408/>. Citado na página 55.

COIN.DANCE. *Bitcoin Statistics*. 2021. (acessado em 16/03/2021). Disponível em: <https://coin.dance/stats>. Citado na página 122.

CORPORATION, I. *Software Guard Extensions Programming Reference*. 2013. (acessado em 10/04/2021). Disponível em: <https://software.intel.com/sites/default/files/managed/48/88/329298-002.pdf>. Citado 2 vezes nas páginas 23 e 46.

DEIRMENTZOGLOU, E.; PAPAKYRIAKOPOULOS, G.; PATSAKIS, C. A survey on long-range attacks for proof of stake protocols. *IEEE Access*, v. 7, p. 28712–28725, 2019. ISSN 21693536. Disponível em: <https://ieeexplore.ieee.org/document/8653269/>. Citado 3 vezes nas páginas 27, 29 e 54.

EYAL, I. et al. Bitcoin-NG: A scalable blockchain protocol. In: *Proceedings of the 13th USENIX Symposium on Networked Systems Design and Implementation, NSDI 2016*. Santa Clara, CA, United States: USENIX Association, 2016. p. 45–59. ISBN 9781931971294. Disponível em: <https://www.usenix.org/conference/nsdi16/technical-sessions/presentation/eyal>. Citado na página 34.

FISCHER, M. J.; LYNCH, N. A.; PATERSON, M. S. Impossibility of Distributed Consensus with One Faulty Process. *Journal of the ACM (JACM)*, v. 32, n. 2, p. 374–382, 1985. ISSN 1557735X. Disponível em: <https://dl.acm.org/doi/10.1145/3149.214121>. Citado 3 vezes nas páginas 17, 26 e 51.

FOUNDATION, A. S. *Documentation Kafka*. [S.l.], 2017. (acessado em 10/04/2021). Disponível em: <https://kafka.apache.org/documentation/>. Citado na página 23.

GILAD, Y. et al. Algorand: Scaling Byzantine Agreements for Cryptocurrencies. In: *SOSP 2017 - Proceedings of the 26th ACM Symposium on Operating Systems Principles*. New York, NY, USA: ACM, 2017. p. 51–68. ISBN 9781450350853. Disponível em: <https://dl.acm.org/doi/10.1145/3132747.3132757>. Citado 4 vezes nas páginas 39, 59, 61 e 64.

GILBERT, S.; LYNCH, N. Perspectives on the CAP Theorem. *Computer*, v. 45, n. 2, p. 30–36, 2012. ISSN 0018-9162. Disponível em: <http://ieeexplore.ieee.org/document/6122006/>. Citado na página 26.

GREVE, F. et al. Blockchain e a Revolução do Consenso sob Demanda. In: *Minicursos do Simpósio Brasileiro de Redes de Computadores e Sistemas Distribuídos*. Campos de Jordão, SP, Brasil: Sociedade Brasileira de Computação, 2018. p. 52. Disponível em: <http://143.54.25.88/index.php/sbrcminicursos/article/view/1770>. Citado 3 vezes nas páginas 17, 20 e 25.

HEILMAN, E. et al. Eclipse Attack on Bitcoin's Peer-to-Peer Network. In: *Proceedings of the 24th USENIX Conference on Security Symposium*. Washington,DC, United States: USENIX Association, 2015. p. 129–144. ISBN 9781931971232. Disponível em: <https://dl.acm.org/doi/10.5555/2831143.2831152>. Citado na página 75.

HYPERLEDGER. *An Introduction to Hyperledger [White Paper]*. 2020. 33 p. (acessado em 20/01/2021). Disponível em: <https://www.hyperledger.org/wp-content/uploads/2018/08/HL_Whitepaper_IntroductiontoHyperledger.pdf>. Citado na página 23.

JYOTHI, B. S.; JANAKIRAM, D. SyMon: A practical approach to defend large structured P2P systems against Sybil Attack. *Peer-to-Peer Networking and Applications*, v. 4, n. 3, p. 289–308, 2011. ISSN 19366442. Disponível em: <http://link.springer.com/10.1007/s12083-010-0084-0>. Citado na página 28.

KANG, J. et al. Toward Secure Blockchain-Enabled Internet of Vehicles: Optimizing Consensus Management Using Reputation and Contract Theory. *IEEE Transactions on Vehicular Technology*, v. 68, n. 3, p. 2906–2920, 2019. ISSN 00189545. Disponível em: <https://ieeexplore.ieee.org/document/8624307/>. Citado na página 21.

KIAYIAS, A. et al. Ouroboros: A provably secure proof-of-stake blockchain protocol. In: *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*. Santa Barbara, CA, United States: Springer International Publishing, 2017. v. 10401 LNCS, p. 357–388. ISBN 9783319636870. ISSN 16113349. Disponível em: <http://link.springer.com/10.1007/978-3-319-63688-7_12>. Citado na página 39.

KING, S. *Primecoin: Cryptocurrency with prime number proof-of-work. http://primecoin.io/bin/primecoin-paper.pdf*. 2013. (acessado em 08/05/2020). Disponível em: <http://primecoin.io/bin/primecoin-paper.pdf>. Citado na página 29.

KING, S.; NADAL, S. *PPCoin: Peer-to-Peer Crypto-Currency with Proof-o-Stake*. 2012. (acessado em 09/05/2020). Disponível em: <https://peercoin.net/assets/paper/peercoin-paper.pdf>. Citado na página 18.

KWON, J. *TenderMint : Consensus without Mining*. Online, 2014. (acessado em 10/12/2020). Disponível em: <tendermint.com/docs/tendermint.pdf>. Citado na página 41.

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, 1982. ISSN 15584593. Disponível em: <https://dl.acm.org/doi/10.1145/357172.357176>. Citado 6 vezes nas páginas 17, 27, 39, 51, 76 e 156.

LIN, I.-C.; LIAO, T.-C. A survey of blockchain security issues and challenges. *International Journal of Network Security*, v. 19, p. 653–659, 09 2017. Citado na página 17.

MAEHARA, Y. et al. Avaliação de mecanismos de consenso para blockchains em busca de nova estratégia mais eficiente e segura. In: *Simposio Brasileiro de Segurança da Informação e de Sistemas Computacionais (SBSEG), 18.* Natal, RN, Brasil: Sociedade Brasileira de Computação, 2018. Disponível em: <https://sol.sbc.org.br/index.php/sbseg/article/view/4267>. Citado na página 29.

MARATHE, N.; GANDHI, A.; SHAH, J. M. Docker swarm and kubernetes in cloud computing environment. In: *Proceedings of the International Conference on Trends in Electronics and Informatics, ICOEI 2019*. Tirunelveli, India: IEEE, 2019. v. 2019-April, p. 179–184. ISBN 9781538694398. Disponível em: <https://ieeexplore.ieee.org/document/8862654/>. Citado na página 98.

MCABEE, A.; TUMMALA, M.; MCEACHEN, J. Military Intelligence Applications for Blockchain Technology. In: *Proceedings of the 52nd Hawaii International Conference on System Sciences*. Honolulu, Hawaii, USA: University of Hawaii, 2019. Disponível em: <http://hdl.handle.net/10125/60038>. Citado na página 21.

MERKLE, R. C. A certified digital signature. In: *Advances in Cryptology — CRYPTO' 89 Proceedings*. New York, NY, United States: Springer New York, 1990. p. 218–238. ISBN 978-0-387-34805-6. Disponível em: <http://link.springer.com/10.1007/0-387-34805-0_21>. Citado 2 vezes nas páginas 22 e 66.

MICALI, S.; RABIN, M.; VADHAN, S. Verifiable random functions. In: *40th Annual Symposium on Foundations of Computer Science*. New York, NY, United States: IEEE Comput. Soc, 1999. p. 120–130. ISBN 0-7695-0409-4. ISSN 02725428. Disponível em: <http://ieeexplore.ieee.org/document/814584/>. Citado na página 61.

MILLER, A. et al. Permacoin: Repurposing bitcoin work for data preservation. In: . San Jose, CA, United States: IEEE, 2014. p. 475–490. ISBN 9781479946860. ISSN 10816011. Disponível em: <http://ieeexplore.ieee.org/document/6956582/>. Citado 3 vezes nas páginas 29, 48 e 49.

MOINDROT, O.; BOURNHONESQUE, C. *Proof of Stake Made Simple with Casper*. Stanford, CA, United States, 2017. 1–7 p. Disponível em: <https://www.scs.stanford.edu/17au-cs244b/labs/projects/moindrot%5C_bournhonesque.pdf>. Citado 3 vezes nas páginas 74, 128 e 159.

NAKAMOTO, S. *Bitcoin: A Peer-to-Peer Electronic Cash System*. bitcoin.org, 2009. 1–9 p. (acessado em 20/05/2020). Disponível em: <https://bitcoin.org/bitcoin.pdf>. Citado 4 vezes nas páginas 18, 29, 30 e 32.

NGUYEN, C. T. et al. Proof-of-Stake Consensus Mechanisms for Future Blockchain Networks: Fundamentals, Applications and Opportunities. *IEEE Access*, v. 7, p. 85727–85745, 2019. ISSN 21693536. Disponível em: <https://ieeexplore.ieee.org/document/8746079/>. Citado 6 vezes nas páginas 10, 21, 32, 39, 87 e 110.

NGUYEN, G. T.; KIM, K. A survey about consensus algorithms used in Blockchain. In: *Journal of Information Processing Systems*. Seoul, Korea: JiPS, 2018. v. 14, n. 1, p. 101–128. ISSN 2092805X. Disponível em: <http://jips-k.org/journals/jips/digital-library/manuscript/file/22872/JIPS-2018-14-1-101.pdf>. Citado na página 17.

NIST. *Federal Information Processing Standard (FIPS 186-3) – Digital Signature Standard (DSS)*. Gaithersburg, MD, United States, 2009. 119 p. Disponível em: <http://nvlpubs.nist.gov/nistpubs/FIPS/NIST.FIPS>. Citado na página 24.

POPOV, S. *The Tangle*. Berlin, Germany: [s.n.], 2018. (acessado em 17/01/2021). Disponível em: <https://assets.ctfassets.net/r1dr6vzfxhev/2t4uxvsIqk0EUau6g2sw0g/45eae33637ca92f85dd9f4a3a218e1ec/iota1_4_3.pdf>. Citado na página 38.

POPOV, S.; BUCHANAN, W. J. FPC-BI: Fast Probabilistic Consensus within Byzantine Infrastructures. *Journal of Parallel and Distributed Computing*, v. 147, p. 77–86, 2021. ISSN 0743-7315. Disponível em: <https://www.sciencedirect.com/science/article/pii/S0743731520303634>. Citado 3 vezes nas páginas 36, 37 e 94.

POPOV, S. et al. *The Coordicide*. Berlin, Germany: [s.n.], 2020. (acessado em 20/10/2020). Disponível em: <https://files.iota.org/papers/20200120_Coordicide_WP.pdf>. Citado 4 vezes nas páginas 38, 150, 152 e 153.

PRADEEP, V. *Ethereum's Memory Hardness Explained — VIJAY PRADEEP*. 2017. (acessado em 05/01/2021). Disponível em: <https://www.vijaypradeep.com/blog/2017-04-28-ethereums-memory-hardness-explained/>. Citado 2 vezes nas páginas 10 e 36.

SLUSHPOOL.COM. *Number of Active workers*. 2021. (acessado em 10/03/2021). Disponível em: <https://slushpool.com/stats/?c=btc>. Citado na página 123.

SONNINO, A.; DANEZIS, G. SybilQuorum: Open Distributed Ledgers Through Trust Networks. *ArXiv*, jun 2019. Disponível em: <http://arxiv.org/abs/1906.12237>. Citado na página 28.

STRUCKHOFF, R. *The Iroha Project to Bring Mobility to Blockchain with Simple APIs*. 2016. (acessado em 20/01/2021). Disponível em: <https://www.altoros.com/blog/the-iroha-project-to-bring-mobility-to-blockchain-with-simple-apis/>. Citado na página 23.

TANENBAUM, A. S.; Maarten Van Steen. *Sistemas Distribuídos princípios e paradigmas*. [S.l.]: Pearson, 2007. 416 p. ISBN 978-8576051428. Citado na página 146.

TSCHORSCH, F.; SCHEUERMANN, B. Bitcoin and beyond: A technical survey on decentralized digital currencies. *IEEE Communications Surveys and Tutorials*, v. 18, n. 3, p. 2084–2123, 2016. ISSN 1553877X. Disponível em: <http://ieeexplore.ieee.org/document/7423672/>. Citado 4 vezes nas páginas 18, 24, 34 e 118.

VASIN, P. *BlackCoin's Proof-of-Stake Protocol V2*. Online, 2013. 2 p. (acessado em 10/05/2020). Disponível em: <https://blackcoin.org/blackcoin-pos-protocol-v2-whitepaper.pdf>. Citado na página 38.

William Feller. *An Introduction to Probability Theory and its Applications*. 3 rd. ed. [S.l.]: Willey, 1968. v. 1. 509 p. ISBN 978-0471257080. Citado na página 119.

William Stallings. *Cryptography and Network Security*. 6. ed. [S.l.]: Pearson, 2013. 752 p. ISBN 978-0133354690. Citado na página 24.

WOOD, G. *Ethereum: a secure decentralised generalised transaction ledger*. Online, 2014. 1–32 p. (acessado em 10/03/2021). Disponível em: <https://gavwood.com/paper.pdf>. Citado na página 35.

XIAO, Y. et al. A Survey of Distributed Consensus Protocols for Blockchain Networks. v. 4, 2020. Disponível em: <http://arxiv.org/abs/1904.04098>. Citado 3 vezes nas páginas 10, 54 e 55.

ANEXO A – Detalhamento dos protocolos

A.1 Produção de blocos

O diagrama de sequência da Fig. A.1 mostra as mensagens do protocolo entre as instâncias das classes que participam do processo de produção de blocos em cada um dos nós do CPoS. É apresentado como um nó cálcula a probabilidade p em um período, assim como sua participação no sorteio, utilizando seu *stake* w_i. Após iniciada, a thread de mineração (*Node Thread Miner*) inicia as interações com instâncias da classe *Sortition* e *Database Interface*. A partir da informação sobre qual é o período atual, o objeto *sortition* calcula o *stake* total do período (W), consultando o valor das transações do tipo 2 (método *db.stake(epoch:E,node:all)*) realizadas e confirmadas no período anterior. Com isso, a probabilidade de sucesso p é calculada e mantida durante todo o período que se inicia. O *stake* w_i associado ao endereço Id_i é obtido de maneira semelhante pelo objeto *sortition*.

Logo após a definição do *stake*, é iniciado um novo período no CPoS, onde primeiramente o nó aguarda o início de uma nova rodada, permanecendo no estado *sleep*. Para isso, ele calcula o tempo faltante para o início de uma nova rodada por meio da diferença entre o tempo de uma rodada T e o transcorrido desde o início da última execução até o momento atual ($T - (time.now() - lastRoundTime)$).

No início de uma nova rodada, o nó calcula a rodada atual (r) e obtém os parâmetros necessários para a execução do sorteio criptográfico. Caso o nó seja sorteado ($j \geq 1$) após o retorno do método *execSortition*, um novo bloco é criado e divulgado para os *peers*. Ao final de cada rodada é verificado se o período atual já está finalizado e, neste caso, um novo período é iniciado. Caso contrário, o nó volta ao estado de sleep até que uma nova rodada seja iniciada. A tabela A.1 descreve o que cada um dos métodos realiza durante a produção de blocos quando chamados por meio das mensagens indicadas no protocolo da Fig. A.1.

A.2 Recebimento de blocos pela rede

Na Fig. A.2 é apresentado o protocolo de recebimento e validação de um bloco. Após iniciada, a thread de escuta (*Thread Listen*) implementa um *socket* de rede que é responsável por aguardar novas transmissões de blocos (*waitingNewBlocksFromPeers()*). Quando uma nova conexão é iniciada e aceita, a *Thread* em questão cria um novo *Worker*, ou seja, uma nova *Thread* que deve processar o recebimento do novo bloco até que o mesmo seja aceito ou rejeitado. Os *bits* do bloco recebido da rede são fornecidos como

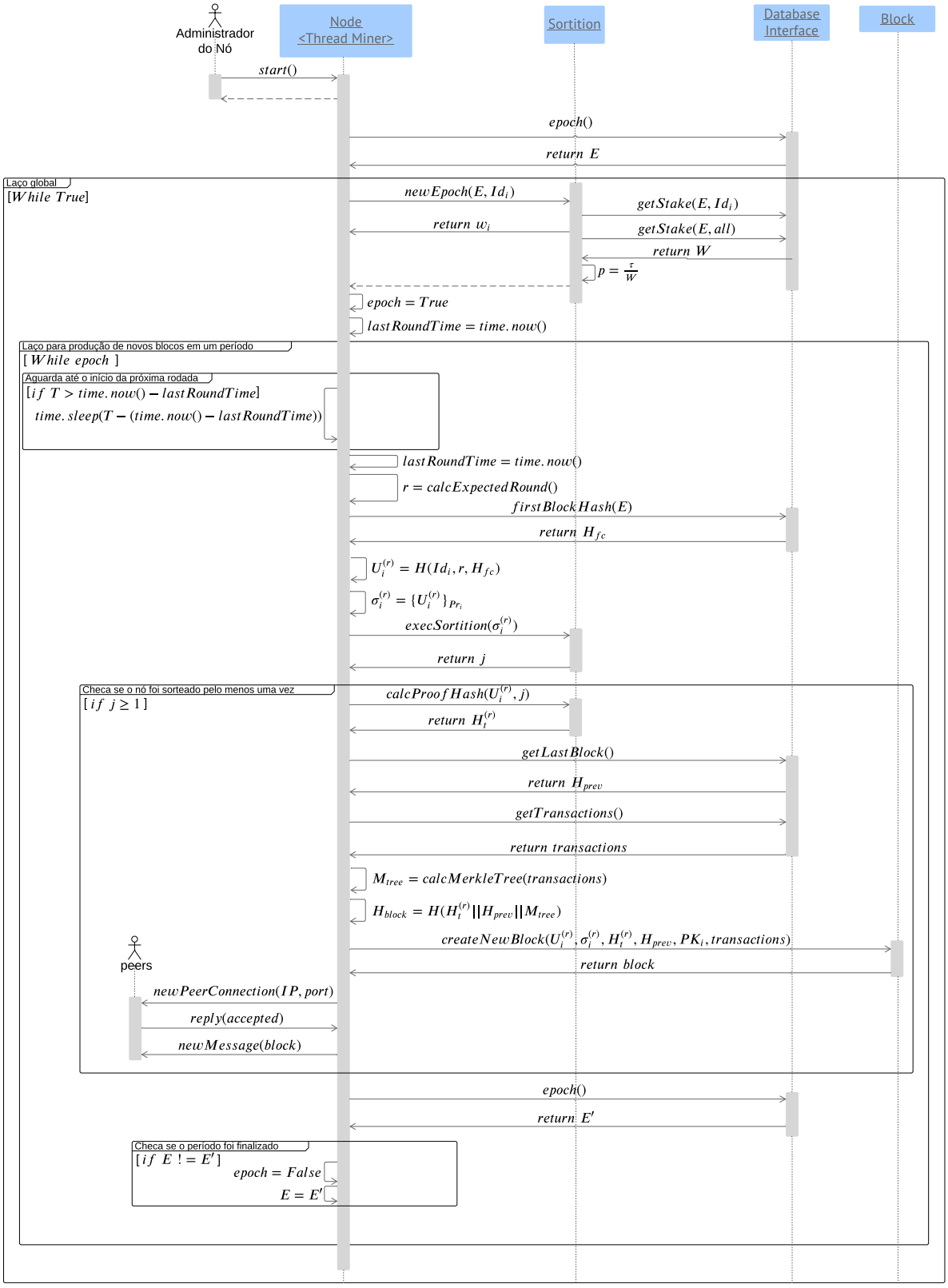

Figura A.1 – Protocolo de produção de blocos dentro do período na visão de um nó

Tabela A.1 – Descrição dos principais métodos utilizados durante a produção de blocos.

Método	**Classe**	**Descrição**
epoch()	Database Interface	Retorna o período atual.
newEpoch(E, Id_i)	Sortition	Envia ao objeto Sortition o período atual e o endereço do nó
getStake(E, $[Id_i : all]$)	Sortition	Retorna o *stake* total do nó representado pelo endereço Id_i ou de todos os nós da rede (all).
time.now()	Node <Thread Mine>	Retorna o tempo UTC atual.
time.sleep()	Node <Thread Mine>	Protocolo entra em estado de *sleep* pelo tempo determinado no parâmetro.
calcExpectedRound()	Node <Thread Mine>	Calcula a rodada esperada com o relógio local do nó.
firstBlockHash(E)	Database Interface	Retorna o primeiro bloco do período atual (H_{fc}) passado como parâmetro.
execSortition($\sigma_i^{(r)}$)	Sortition	Retorna o número de vezes que o nó foi sorteado na rodada r.
calcProofHash($U_i^{(r)}$,j)	Sortition	Retorna o menor *hash* de prova (hash de prioridade) do nó sorteado.
getLastBlock()	Database Interface	Retorna o último bloco da blockchain local.
getTransactions()	Database Interface	Retorna o conjunto de transações para serem inseridas no novo bloco.
calcMerkleTree(transactions)	Node <Thread Mine>	Calcula a árvore de Merkle a partir do conjunto de transações fornecido como parâmetro.
$H_{block} = H(H_t^{(r)}\|\|H_{prev}\|\|M_{tree})$	Node <Thread Mine>	Calcula o *hash* do novo bloco produzido.
createNewBlock($U_i^{(r)}$,$\sigma_i^{(r)}$,$H_t^{(r)}$,H_{prev},PK_i, $transactions$)	Block	Cria um novo bloco a partir dos parâmetros passados como parâmetro.
newPeerConnection(IP,port)	Node <Thread Mine>	Cria uma nova conexão com um *Peer* para transmitir o bloco produzido.
reply(accepted)	Node <Thread Listen> (*Peer*)	Retorno do *Peer* aceitando a conexão.
newMessage(block)	Node <Thread Listen>	Envia o bloco produzido para o *Peer*.

parâmetro da chamada ao método *createNewBlock(newMsg.block)* da classe *Block*. Este método retorna um novo objeto da classe *Block*, que é utilizado em algumas verificações que definem o destino do bloco em questão. Primeiramente, a partir da chamada ao método *checkBlockRound(block.round)*) da classe *Validation*, é verificado se o bloco tem uma rodada válida. Caso seja recebido *status* = *True*, é verificado se o bloco é sucessor do último bloco da visão local do nó e, logo depois, caso o bloco seja realmente o próximo da cadeia, o objeto *block* é fornecido como entrada do método *checkAcceptanceBlock(block)*, que tem a função de validar se o bloco de fato foi criado a partir de um sorteio válido e se o nó criador utilizou corretamente o seu *stake*. Caso as verificações indiquem que o bloco seja válido, o mesmo é inserido na blockchain através da chamada *insertBlock(block)*. Por fim, a tabela A.2 descreve cada um dos métodos mostrados no protocolo da Fig. A.2.

A.3 Protocolo de confirmação e ressincronização

A Fig. A.3 apresenta o protocolo que descreve as ações tomadas pelo nó quando uma ressincronização é detectada no início de uma rodada. De acordo com o diagrama, após o início de uma nova rodada a thread principal chama o método *updateBlockView* que pertence a um objeto do tipo *Blockchain control*. Este método, por sua vez, busca o próximo bloco a ser confirmado na base de dados (chamada *firstBlockUnconfirmed*) e, assim, calcula a média de sorteios bem sucedidos que confirma a visão atual do nó e que envolva o bloco retornado (chamada *calcAverage(r, block.numSucess, block.round)*). A partir desta média, o método *updateBlockView* é capaz de decidir se o bloco é considerado confirmado (se $\bar{s}_v^{(r)} \geq \bar{s}_{min}^{(r)}$). Caso o bloco atenda a condição de confirmação, o método *confirmBlock* é chamado, alterando para confirmado o estado do bloco no banco de dados

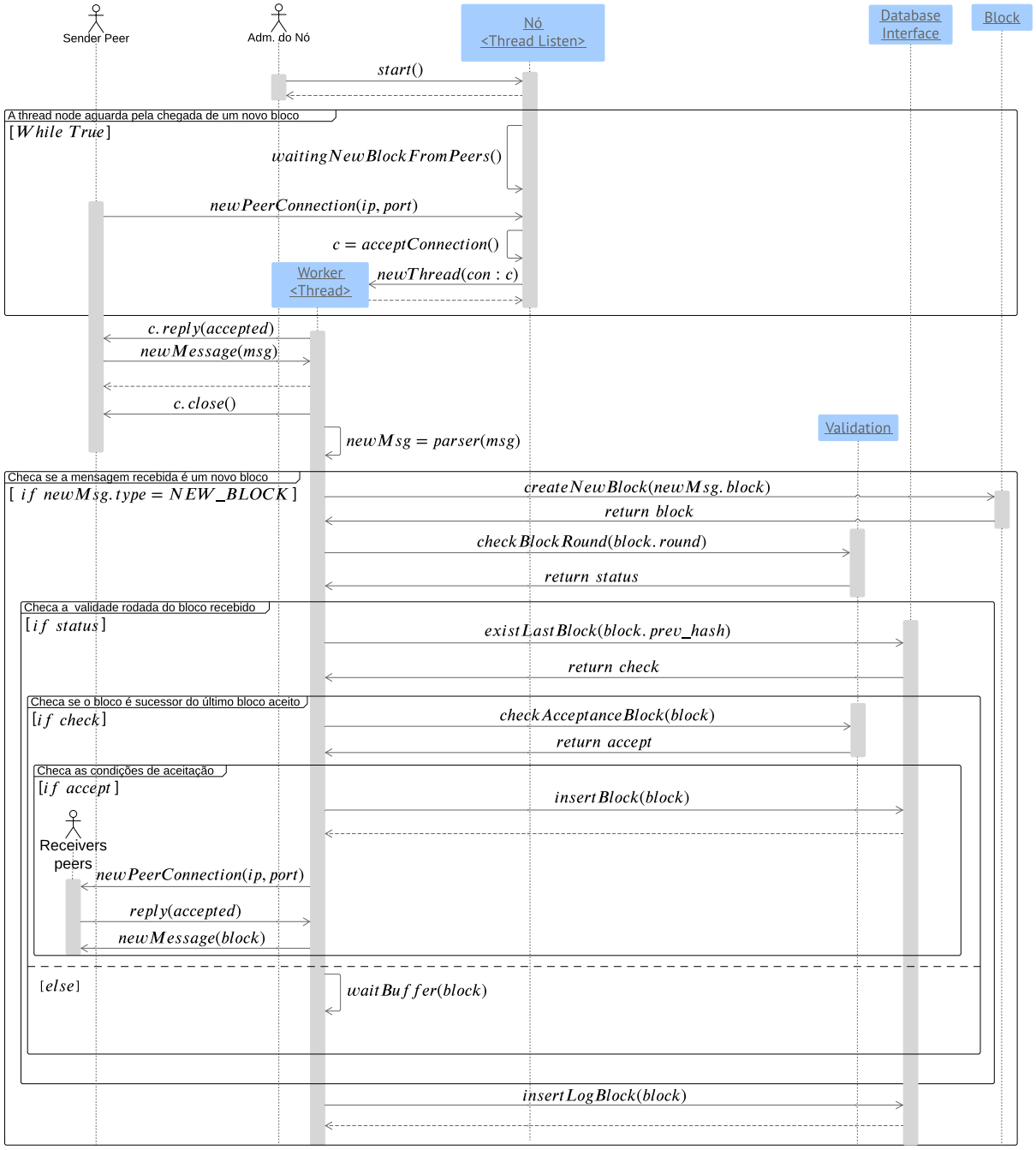

Figura A.2 – Protocolo de recebimento de um bloco no CPoS

que armazena a visão local da blockchain. Caso o bloco não seja confirmado, é verificado se a visão atual ainda deve ser considerada válida, ou se o nó deve disparar uma solicitação de ressincronização, baseado na condição $\bar{s}_v^{(r)} < \alpha^{(r)}$.

Caso seja necessário, a solicitação de ressincronização é realizada a partir de um sinal (*syncsignal(bc)*) transmitido para a thread de sincronização que, por sua vez, sai do estado de *idle* (*wait_for_syncsignal*) e inicia a solicitação dos blocos para os *peers*. Para isso, o nó envia mensagens do tipo *requestBlock()*, fornecendo como parâmetro inicial a constante $k = LASTID$ para que os *peers* enviem primeiramente o último bloco conhecido.

Tabela A.2 – Descrição dos principais métodos utilizados durante a chegada de de novos blocos.

Método	**Classe**	**Descrição**
waitingNewBlockFromPeers()	Node <Thread Listen>	Coloca a Thread de Listen no estado de espera por um novo bloco.
newPeerConnection(ip,port)	*Peer*	*Peers* solicita uma nova conexão com o socket da Thread Listen.
acceptConnection()	Node <Thread Listen>	Aceita uma nova conexão com o *Peer* solicitante.
newThread(c)	Node <Thread Mine>	Cria uma nova Thread Worker responsável por tratar da nova conexão.
c.reply(accepted)	Node <Thread Worker>	Informa o *Peer* que o seu pedido de conexão foi aceito.
newMessage(msg)	*Peer*	Envia um novo bloco através dos sockets estabelecidos na conexão.
createNewBlock(block)	Block	Cria um novo bloco a partir dos parâmetros passados como parâmetro.
checkBlockRound(block.round)	Validation	Retorna se a rodada do bloco recebido é válida.
existLastBlock(block.prev_hash)	Database Interface	Retorna verdadeiro se o bloco recebido é sucessor do último bloco aceito pelo nó.
checkAcceptanceBlock(block)	Validation	Retorna verdadeiro se o bloco foi produzido por um nó sorteado na rodada a partir de um *stake* válido.
insertBlock(block)	Database Interface	Insere o novo bloco recebido na cópia local que o nó tem da blockchain.
waitBuffer(block)	Node <Thread Mine>	Armazena bloco em um buffer na memória por algumas rodadas. Esse bloco pode ser utilizado caso ocorra uma ressincronização
insertLogBlock(block)	Database Interface	Insere o novo bloco recebido em um log local.

Estas solitações têm como objetivo encontrar uma nova cadeia, que seja capaz de melhorar a média de sorteios bem sucedidos do nó e, com isso, substituir sua cadeia atual. A thread de sincronização faz uma comparação entre a cadeia atual do nó e a recebida pelos *peers*. Isso é feito através da chamada *checkNewChain(newchainlist)*. Neste processo de comparação, o método calcula a média de cada cadeia a partir do bloco que vem logo após o ponto de *fork* que deu origem a elas, até o último bloco de cada cadeia, como mostrado na Fig. A.4.

De acordo com a figura, as cadeias podem ter tamanhos diferentes em relação ao número de blocos, já que podem haver rodadas sem a produção de blocos em qualquer uma das cadeias para o intervalo solicitado. Outro ponto importante, é que o nó muda sua visão apenas quando encontra uma cadeia com melhor média no intervalo de rodadas atual e, portanto, esta ação não depende do número de blocos das cadeias envolvidas. Neste sentido, caso a nova cadeia tenha média melhor, ela substitui a cadeia atual do nó até o ponto de *fork* que deu origem as duas visões. No diagrama de sequência, estes passos são mostrados primeiramente a partir da chamada do método *getSubChain(firstround, r)*, que recebe como entradas a rodada de criação do primeiro bloco recebido e a rodada atual r. Com base nestas entradas, o método *getSubChain* busca a cadeia atual do nó, onde a rodada de cada bloco deve pertencer ao intervalo $[firstround, r]$. A média destas cadeias são então comparadas através da chamado do método *compareSubChain(newchainlist, localchainlist)*. A nova cadeia é inserida, substituindo a atual, se o resultado da comparação retornar *True*, indicando que a média da nova cadeia é melhor que a atual. Neste caso, a inserção ocorre através do método *changesubchain(newchainlist)*.

Por fim, uma alteração de cadeia pode provocar uma mudança no período atual. Portanto, sempre que ocorre uma ressincronização com substituição de uma parte da visão local da blockchain, o nó verifica se é necessário mudar o período atual. Caso a alteração

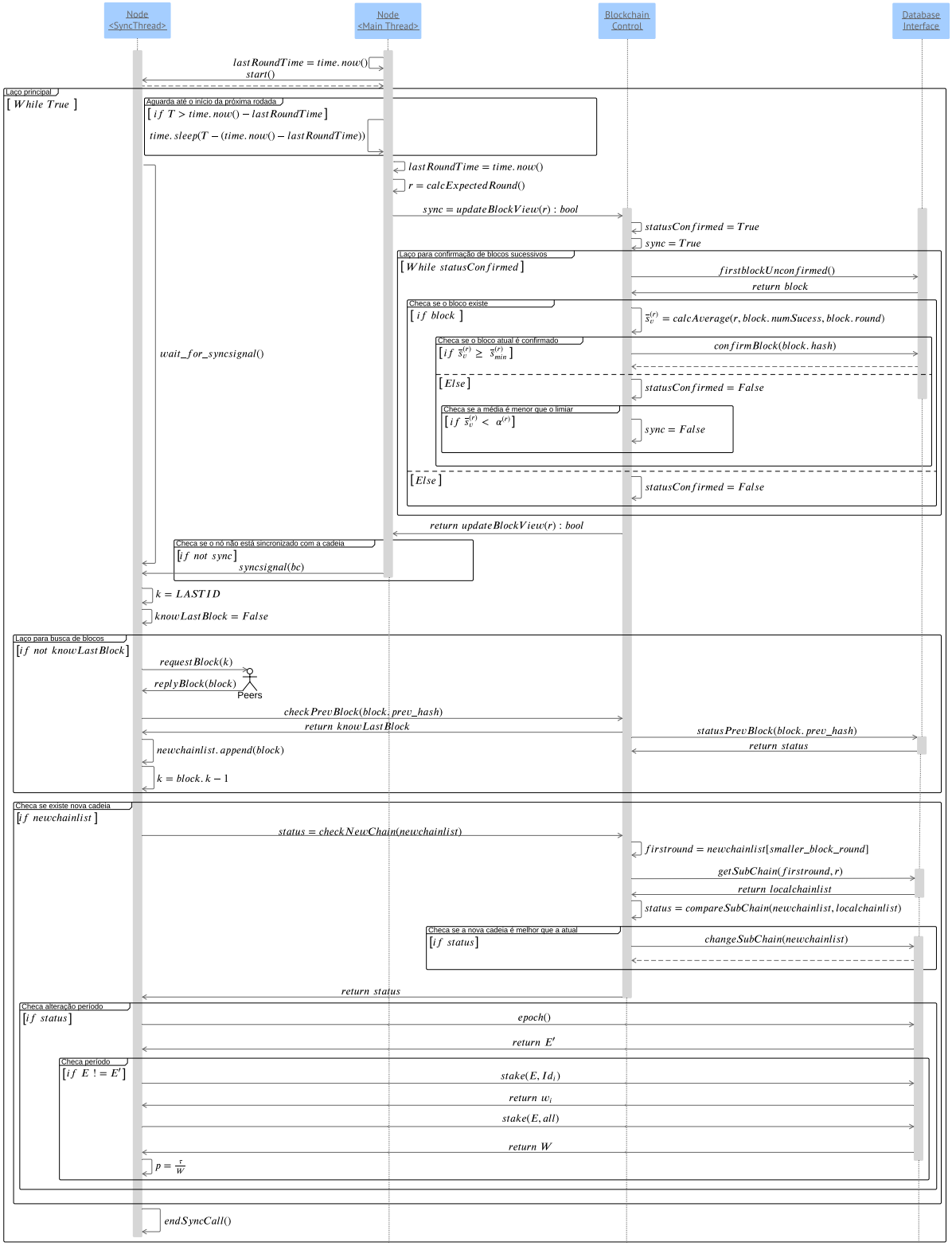

Figura A.3 – Protocolo de confirmação e ressincronização

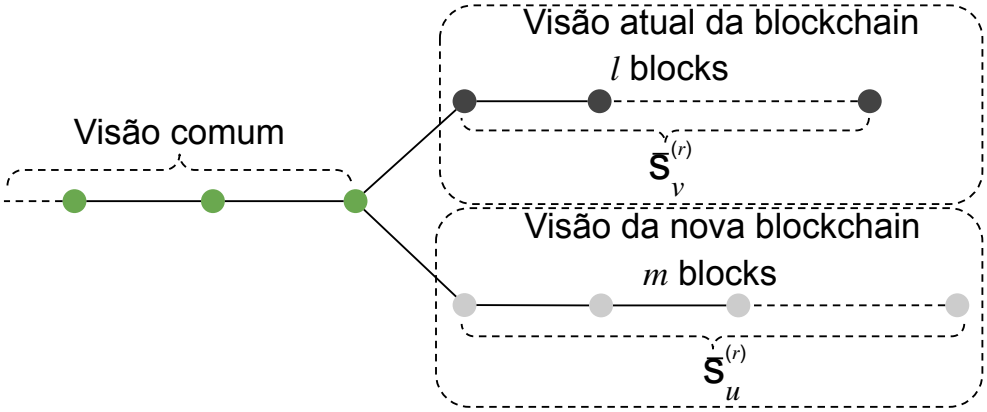

Figura A.4 – Comparação entre as médias da cadeia atual e a recebida

seja necessária, ele solicita novamente o valor de seu próprio *stake* w_i, além de recalcular a probabilidade de sucesso p, a partir da quantidade de *stake* total W presente no novo período. A tabela A.3 descreve cada um dos métodos mostrados no protocolo da Fig. A.3.

Tabela A.3 – Descrição dos principais métodos utilizados durante a confirmação e ressincronização de blocos.

Método	**Classe**	**Descrição**
updateBlockView(r):bool	Blockchain Control	Retorna verdadeiro se ocorrerem mudanças na visão local que o nó tem da blockchain.
firstBlockUnconfirmed()	Database Interface	Retorna o bloco mais antigo que ainda não foi confirmado na visão local que o nó tem da blockchain.
calcAverage(r, *block.numSucess*, *block.round*)	Blockchain control	Calcula a média $\bar{s}_v^{(r)}$ na rodada r para o próximo bloco que deve ser confirmado.
confirmBlock(block.hash)	Database Interface	Confirma o bloco com *hash* igual ao que foi passado como parâmetro.
syncSignal(bc)	Node <Sync Thread>	Ativa a Thread de sincronismo para atuar em uma nova ressincronização.
requestBlock(*altura*)	*Peer*	Solicita ao *Peer* o seu bloco de altura k.
checkPrevBlock(block.prev_hash)	Blockchain control	Retorna verdadeiro se o nó tem em sua visão local da blockchain o bloco com *hash* igual aquele passado como parâmetro.
newchainlist.append(block)	Node <Sync Thread>	Armazena na estrutura newchainlist o bloco retornado pelo *Peer*.
getSubChain(firsround, r)	Database Interface	Retorna parte da cadeia local atual do nó do bloco com rodada *firstblock* até o bloco com rodada r.
compareSubChain(newchainlist,localchainlist)	Database Interface	Compara a média da subcadeia recebida por meio das solicitações enviadas aos *Peers* com a subcadeia atual local do nó.
changeSubChain(newchainlist)	Database Interface	Troca a subcadeia original do nó pela subcadeia recebida por meio das solicitações enviadas aos *Peers*.
endSyncCall()	Node <Sync Thread>	Finaliza o trabalho da Thread de sincronismo, levando ela novamente para o estado de espera por um novo pedido de ressincronização.

Por fim, ressaltamos que o protocolo da Fig. A.3 também é utilizado quando um novo nó entra no CPoS com o objetivo de participar do consenso. Para um nó que acabou de entrar no mecanismo o processo de sincronismo será disparado rapidamente, pois esse nó conhece a princípio apenas o bloco gênesis e, por isso, irá em um primeiro momento produzir blocos a partir dele. Como todos os outros nós estão empregando seus *stakes* em uma outra visão, o nó entrante não será capaz de calcular uma média a partir de sua visão local que seja maior que o mínimo necessário para mantê-la, o que levará o nó a um pedido de ressincronização. É esperado que a ressincronização de um nó entrante ocorra logo após a primeira rodada de participação.

ANEXO B – Intervalo de tolerância

O CPoS foi apresentado como um mecanismo síncrono onde, em cada rodada, um novo ciclo para produção de blocos tem início. Assim, todo nó tenta criar um bloco e associar a ele uma rodada atual calculada a partir de seu relógio local. Além disso, os nós da rede são capazes de validar se a rodada de um bloco corresponde à rodada esperada, calculada também a partir de sua fonte de relógio local. Todos os cálculos para definir a rodada de um bloco e sua posterior validação são realizadas com base na referência de tempo UTC (*Coordinated Universal Time*).

Em uma rede descentralizada, onde cada nó utiliza seu próprio relógio para definir a duração da rodada esperada, é possível que não exista concordância sobre ela, já que os osciladores de *quartzo* utilizados na produção do *clock* de cada relógio possuem impurezas, provocando pequenas variações na frequência que define a unidade de tempo em cada relógio (TANENBAUM; Maarten Van Steen, 2007). Este fato resulta em diferenças de tempo entre os relógios dos nós. Por este motivo, é razoável definir um intervalo de tolerância para o CPoS para que blocos possam ser aceitos mesmo que suas rodadas não sejam exatamente iguais à rodada esperada, calculada pelos nós verificadores. A partir da Fig. B.1 é iniciada uma análise que permite definir um intervalo de tolerância capaz de atender as necessidades do mecanismo CPoS sem incorrer em problemas de segurança ou de desempenho.

Primeiramente, para cada transmissão realizada são analisados os limites temporais para que a origem e o destino possam concordar sobre a rodada. Na Fig. B.1(a) existem dois nós que trocam blocos entre si com diferentes atrasos de propagação pela rede. O nó dois está adiantado em relação ao tempo real $+\delta_2$, enquanto o nó um está atrasado $-\delta_1$. É esperado que os nós transmitam o bloco logo no início de cada rodada em que forem sorteados. Assim, quando o nó adiantado transmite um bloco em direção ao atrasado, para que exista concordância de rodada entre eles, é necessário que o atraso de transmissão Δ_{t2} seja maior que a diferença de tempo entre os dois nós ($\delta_2 - (-\delta_1) = \delta_2 + \delta_1$) e menor que a soma desta diferença com o período T de cada rodada. Logo, é necessário que $(\delta_2 + \delta_1) < \Delta_{t2} < (\delta_2 + \delta_1 + T)$.

Em linhas gerais, este intervalo para Δ_{t2} é igualmente válido para as situações mostradas em (b) e (c), considerando sempre os casos em que a transmissão do bloco ocorra do nó mais adiantado para o mais atrasado. Portanto, é apresentado na Eq. B.1 o intervalo para o atraso de transmissão (Δ_{t2}) capaz de garantir concordância de rodada

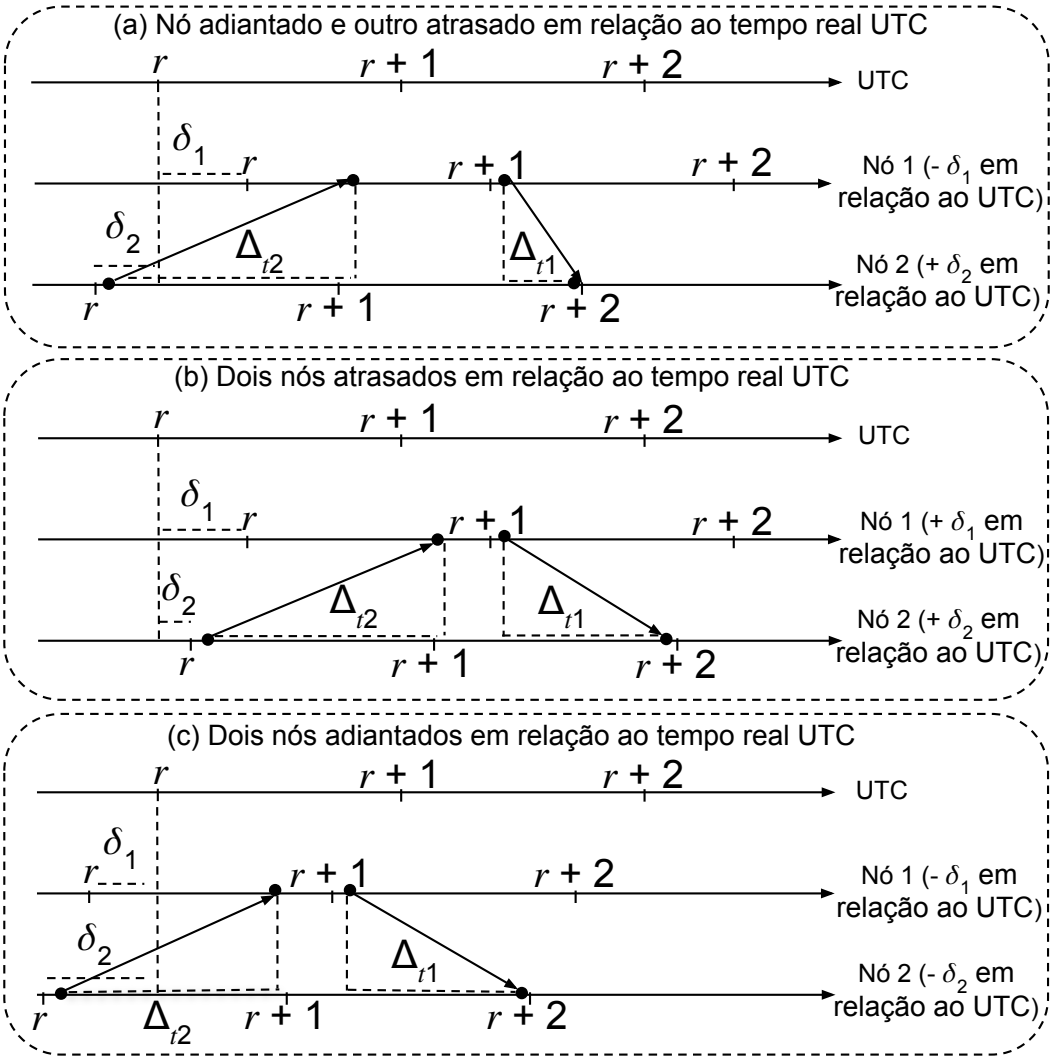

Figura B.1 – Análise dos limites temporais para que exista concordância em relação a rodada atual

entre os nós, para transmissões com origem em nós mais adiantados que os destinos.

$$|\delta_2 - \delta_1| < \Delta_{t2} < T + |\delta_2 - \delta_1| \tag{B.1}$$

Ainda na mesma Fig. B.1(a), podemos ver que um bloco transmitido pelo nó 1 (atrasado) no início de $r+1$ só chegará no nó 2 na mesma rodada se $0 < \Delta_{t1} < (\delta_1 - \delta_2 + T)$. Esse intervalo também é válido nas situações em que os dois nós estão atrasados em relação ao referencial UTC (Fig. B.1 (b)) e a transmissão parte do nó mais atrasado, ou quando os dois nós estão adiantados (Fig. B.1 (c)). Assim, para transmissões do nó mais atrasado em direção ao mais adiantado, não existem valores mínimos para Δ_{t1}, já que a rodada $r+1$ é iniciada sempre antes nos destinos; entretanto, o tempo máximo de transmissão é menor que o tempo de duração de uma rodada (T). Neste sentido, a Eq. B.2 define o limite máximo para que um bloco enviado por um nó atrasado seja recebido ainda dentro

da rodada de criação por um nó adiantado para todos os casos possíveis.

$$\Delta_{t1} < T - |\delta_2 - \delta_1| \tag{B.2}$$

De acordo com a Eq. B.2, se $\Delta_{t1} > T - |\delta_2 - \delta_1|$ significa que o nó destino (adiantado) já estará em uma nova rodada na chegada do bloco e, portanto, ele irá considerar o bloco atrasado. Ainda assim, não se justifica a utilização de tolerância para aceitar blocos atrasados, já que é esperado que a quantidade de tempo $T - |\delta_2 - \delta_1|$ esteja próximo de T para quase todos os casos. A afirmação acima é verdadeira para atrasos de relógio muito menores que o tempo de duração da rodada, o que é esperado na prática. Logo, mesmo a partir de diferentes fontes de relógio é esperado que os nós consigam transmitir blocos que sejam capazes de alcançar os nós da rede ao ponto de não serem considerados atrasados, para isso basta que o tempo de duração da rodada seja definido adequadamente, buscando suportar os maiores atrasos de transmissão da rede, e que os nós sincronizem seus relógios com uma fonte de tempo UTC confiável de maneira periódica. Além disso, como foi definido na seção 4.4, blocos com rodadas mais antigas têm preferência no critério de aceitação e, um atacante pode de maneira proposital divulgar um bloco na rede apenas quando ele já está atrasado, o que provoca reversões na blockchain caso uma tolerância para blocos atrasados seja utilizada.

Por outro lado, para transmissões de blocos a partir de um nó adiantado, o limite superior do atraso de transmissão é beneficiado pela diferença de tempo, se tornando maior que o valor esperado T ($\Delta_{t2} < T + |\delta_2 - \delta_1|$). Neste cenário, é possível que o atraso de transmissão seja maior que a duração de uma rodada e os nós ainda estarem em concordância sobre a rodada atual. Entretanto, as transmissões neste sentido também possuem um limite inferior ($|\delta_2 - \delta_1|$). Como as rodadas dos nós mais adiantados começam antes, é necessário um tempo mínimo de propagação para que o bloco seja recebido pelo nó atrasado apenas depois que ele tenha trocado para a mesma rodada. Diferentemente do caso anterior, onde a duração da rodada exerce controle sobre o limite superior, neste caso o limite inferior não é controlado por um parâmetro definido pelo mecanismo. Não é possível assumir que o limite inferior será atendido, já que variações no tempo de transmissão são esperadas devido a fatores externos como, por exemplo, mudanças no desempenho da rede. Por este motivo, é justificada a utilização de uma tolerância capaz de garantir a aceitação de blocos com rodadas adiantadas, ou seja, aqueles enviados por nós com relógios adiantados e que chegam no destino com um tempo menor que o limite inferior.

Portanto, um bloco com rodada de criação x é aceito como válido por um nó qualquer se x estiver no intervalo $[r; r + tol]$, onde r é a rodada atual calculada pelo nó e tol é um número fixo de rodadas definido pelo sistema. Este intervalo garante que blocos com rodadas adiantadas sejam aceitos por nós atrasados até um limite de tol rodadas.

O parâmetro *tol* é definido para todos os nós que compõem a rede do CPoS e como sua utilização infuencia apenas blocos adiantados, ou seja, aqueles que são recebidos com rodadas maiores que as calculadas pelos nós destinatários, é possível afirmar que as decisões tomadas por meio dos critérios de aceitação (Seção 4.4) não são impactadas pelo seu uso. Assim, a visão local da blockchain de um nó evolui a cada nova rodada e nunca pode ser revertida pela substituição de algum de seus blocos que foram aceitos em rodadas já finalizadas. Desta forma, a utilização de um intervalo de tolerância com *tol* rodadas não traz impactos à confirmação probabilística (Seção 4.5), sendo que a única maneira possível de um nó mudar de visão é a partir de um novo pedido de ressincronização, como mostrado na Seção 5.2.

ANEXO C – Processo de conexão de um nó com a rede *peer-to-peer*

No CPoS, um nó (ou *peer*) é um *host* controlado por um participante e que mantém informações sobre a blockchain que está em constante evolução ao longo do tempo. Neste sentido, para que a visão da *blockchain* possa ser única, nós trocam informações a cada rodada, ou seja, compartilham blocos conhecidos. Assim, é necessário que cada nó estabeleça um canal de comunicação com um pequeno conjunto de *peers*. Para isso, cada nó participa de um processo chamado *autopeering* que é adaptado para o CPoS a partir do mecanismo descrito por Popov et al. (POPOV et al., 2020). O processo de *peering* é em geral vulnerável ao *eclipse attack*, que ocorre quando todos os *peers* de um nó são controlados por um atacante, que passa a determinar quais dos blocos produzidos são enviados para ele. De acordo com Popov et al. (POPOV et al., 2020), é importante definir um mecanismo capaz de garantir que os nós possam se conectar na rede por meio de um processo automatizado e de escolha aleatória, onde os nós não são capazes de controlar com quais *peers* as conexões são estabelecidas. Este processo é chamado de *autopeering* e nesta seção ele é descrito a partir de um conjunto de adaptações que garantem sua operação no CPoS.

C.1 Peer discovery

Primeiramente no CPoS será estabelecida uma lista de nós confiáveis que acompanha o código fonte do mecanismo. Esta é uma abordagem comum em muitos sistemas distribuídos como ponto de partida para que novos nós tenham um conjunto inicial de *peers* para inicializar o protocolo de *peering* (POPOV et al., 2020). Desta forma, o novo nó envia mensagens do tipo *peer discovery* aos nós desta lista que respondem com o subconjunto de *peers* conhecidos por eles. O CPoS compartilha as chaves públicas dos nós confiáveis para evitar que as mensagens de *peer discovery* enviadas para estes nós sofram tentativas de *hijacking attack* . Portanto, estes novos *peers* são a princípio inseridos em um conjunto de *peers* não verificados. Apenas depois da confirmação da chave pública utilizada pelos dois nós envolvidos na comunicação é que o *peer* passa a ser considerado no conjunto dos nós verificados. Assim, o processo prevê um esquema de autenticação mútua para evitar este tipo de ataque, por meio de um protocolo do tipo *ping-pong*. Neste protocolo, um *peer* X envia uma mensagem assinada do tipo *ping* contendo sua chave pública para outro *peer* Y. O *peer* Y responde a mensagem inicial com uma mensagem assinada do tipo *pong* agora contendo a sua chave pública. Ambos os *peers* verificam a

assinatura aplicando a correspondente chave pública e na sequência adicionam o *peer* descoberto ao seu conjunto local de *peers* verificados. O protocolo *ping-pong* é importante, pois é uma forma de associar uma identidade a um único ip, evitando que um mesmo nó assine com a mesma chave privada diferentes endereços ips.

A descoberta de novos *peers* e posterior confirmação de suas chaves públicas garantem apenas que os nós possam aprender de maneira eficiente o maior número possível de *peers*, porém estes ainda não são considerados em sua lista de vizinhos, ou seja, como pertencentes ao conjunto de *peers* com quem ele troca novos blocos produzidos. Além disso, em um sistema distribuído, o número de *peers* muda constantemente, à medida que nós saiam ou entrem na rede. Portanto, o nó envia mensagens do tipo *peer discovery* periodicamente para seus *peers*, com o objetivo de manter atualizada sua lista local. Quando, a partir do envio destas mensagens, novos *peers* são encontrados, o nó adiciona os mesmos à lista de não verificados e, logo depois, envia novas mensagens do tipo *peer discovery* destinadas a eles, com o objetivo de descobrir um novo conjunto *peers*. Ao conjunto formado por todos os *peers* presentes na lista de não verificados, o nó inícia o protocolo *ping-pong*, buscando confirmar quais *peers* são realmente donos das chaves públicas divulgadas. Os nós que respondem corretamente são inseridos no conjunto de *peers* verificados, como ilustrado na Fig. C.1.

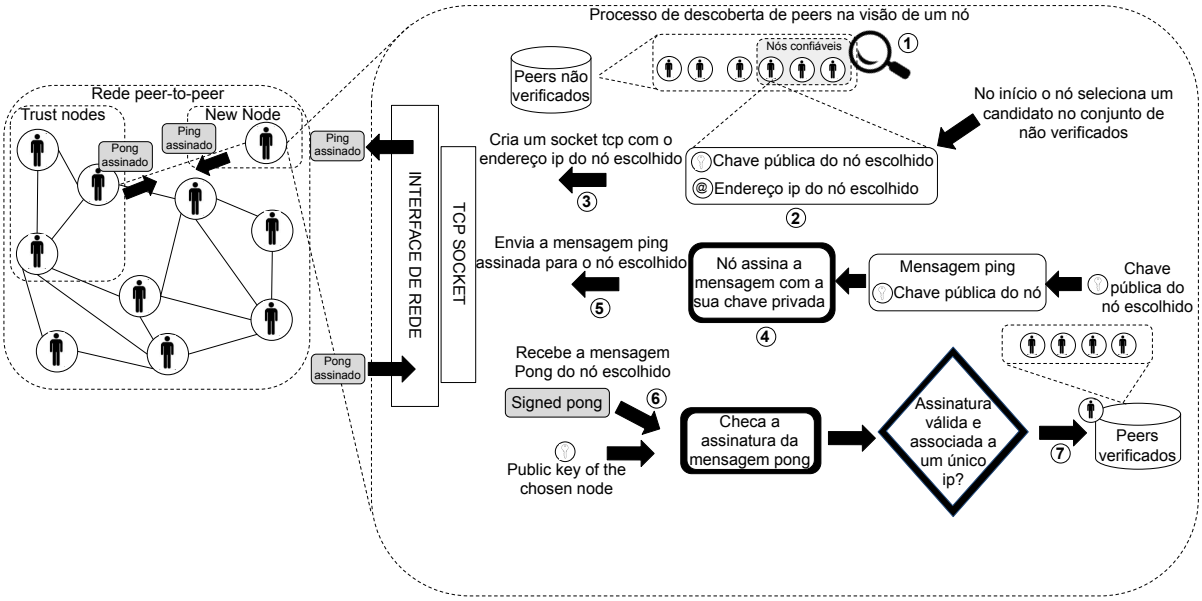

Figura C.1 – Nós iniciam e mantêm seu conjunto de *peers* atualizado por meio de *discovery message*

De acordo com a figura, o nó inícia o processo escolhendo um *peer* do conjunto de não verificados e no passo 2 consulta seu endereço ip e chave pública. Inicialmente, apenas os *peers* enviados com o código fonte do CPoS fazem parte deste conjunto, porém quando novos *peers* são descobertos, o protocolo *ping-pong* é estendido a eles. O nó cria um *socket tcp* com o *peer* escolhido no passo 3 e envia uma mensagem assinada contendo

sua chave pública, como mostrado no passo 4. O *peer* escolhido ao receber a mensagem, envia como resposta a sua chave pública através de uma mensagem assinada do tipo *pong*. No passo 6, o nó valida a assinatura do *peer* escolhido, ou seja, ele verifica se a mensagem *pong* foi de fato assinada pela chave privada associada à chave pública divulgada pelo *peer*, e se nenhuma outra mensagem foi assinada anteriomente com a mesma chave, utilizando outro endereço ip. Caso as condições sejam atendidas, o nó adiciona o novo *peer* ao seu conjunto de verificados.

C.2 Seleção de vizinhos

O mecanismo de seleção de vizinhos é parte importante do processo de *autopeering*. Os vizinhos são *peers* com quem o nó estabelece conexões e a partir delas envia e recebe os blocos produzidos pela rede. Para evitar que um atacante consiga disseminar *peers* controlados por ele e, com isso, possa aplicar ataques do tipo *eclipse*, o CPoS torna pseudoaleatório e verificável o processo de seleção de vizinhos, a partir de uma adaptação do esquema proposto por Popov et al. (POPOV et al., 2020). Desta forma, um nó envia uma mensagem do tipo *peering request* para um candidato a *peer*, e o mesmo é capaz de verificar se a solicitação é de fato aleatória. Além disso, baseado em cálculos feitos com parâmetros conhecidos apenas por ele, o *peer* informa se aceita ou não o nó, respondendo à solicitação com uma mensagem do tipo *peering reply*.

Deste modo, um nó i, representado por sua identidade Id_i, escolhe os potenciais candidatos a *peers*, medindo à distância d (Eq. C.1) entre ele e um *peer* (Id_j) já verificado pelo protocolo *ping-pong*. Como parâmetro não controlado pelo nó i na Eq. C.1, utiliza-se a rodada atual r, que pode ser validada por todos os nós da rede. Além disso, o nó i deve ter pelo menos um *stake* ($w_i \geq 1$) associado à identidade Id_i durante o período (E) em que ele inícia o processo de *peering*. Este *stake* deve ser enviado ao Id_i através de uma transação realizada no período anterior ($E-1$), como explicado na seção 4.3. Isso é necessário para evitar que um atacante produza várias identidades (*sybil attack*) e com elas aumente as chances de aplicar um ataque do tipo *eclipse* na rede. Caso não seja exigido que os nós tenham *stake* associado às suas identidades durante o processo de *peering*, um atacante pode por meio de força bruta, criar identidades próximas da vítima em relação à distância d. Assim, o atacante pode controlar todos os *peers* da vítima, mesmo a partir de um esquema aleatório para formação de *peers*. Neste sentido, é fundamental que as *Sybil* identidades sejam evitadas por meio da associação do *stake*.

A distância d é calculada através de uma operação xor ($\oplus$) entre os bits de duas diferentes saídas de uma função *hash* de 256 bits.

$$d(Id_i, Id_j, r) = hash_{256}(Id_i) \oplus hash_{256}(Id_j||r) \tag{C.1}$$

Todos aqueles *peers* que apresentarem distância em relação ao nó i menor que um valor θ definido pelo mecanismo são ordenados de maneira crescente em relação à distância. Para estes *peers*, o nó envia uma mensagem do tipo *peering request* primeiramente aos nós mais próximos. O processo é reiniciado a cada nova rodada e termina quando o número de *peers* desejado (k) é obtido pelo nó i. O estabelecimento de *peers* é descrito por meio do algoritmo 2.

Algoritmo 2: Select neighbors $(Id_i, r, k, Verified\ \ set\ \ V)$

```
Q ← sortByDistanceAsc(V, Id_i, r)
Peers ← []
foreach t ∈ Q do
    peerRequest ← sendPeerRequest(t)
    if peerRequest.accepted then
        append(Peers, t)
        if |Peers| ≥ k then
            return Peers
end
return Peers
```

Cada *peer* que recebe uma mensagem do tipo *peering request* decide se aceita o nó i como vizinho a partir de um cálculo local de distância que apenas ele pode executar. Para isso, um *peer*, representado pela sua identidade Id_j, sorteia um número pseudoaleatório (ζ_j) e mantém este valor em segredo. Por questões de segurança este número deve ser alterado periodicamente pelos *peers* da rede. Deste modo, o *peer* calcula a distância d entre ele e o nó solicitante ($d(Id_i, Id_j, \zeta_j)$) por meio da Eq. C.1, utilizando seu parâmetro secreto ζ_j. Assim, mesmo que o nó i consiga atender os requisitos para que uma conexão seja estabelecida, ela ainda pode não ser aceita pelo *peer*. Diz-se que um *peer* aceita um novo pedido de conexão se ao menos uma das duas condições abaixo é satisfeita (POPOV et al., 2020):

- A tentativa de conexão está mais próxima que a de algum vizinho já conhecido;
- O *peer* que recebe a conexão ainda não tem um número suficiente de vizinhos;

Além das condições acima, para cada nova mensagem *peer request* recebida, o *peer* verifica se a requisição atende a condição $d(Id_i||Id_j||r) < \theta$ e se o dono da identidade Id_i tem algum *stake*. Para θ pequeno torna-se mais díficil para um atacante decidir em quais nós se conectar; entretanto, a fase de *peering* do protocolo será mais lenta.

C.3 Controle das transmissões na rede *peer-to-peer*

A partir do processo aleatório de formação dos *peers* descrito nesta seção, é esperado que um determinado bloco seja recebido por um nó i, a partir de diferentes

peers. Cada transmissão passa por um conjunto de nós diferentes antes de alcançar o nó i, criando estruturas redundantes e aleatórias na rede *peer-to-peer*. Os objetivos destas duas características da rede são evitar que um adversário possa executar o *eclipse attack* e garantir que o esquema de transmissão de blocos seja resiliente à falhas nos nós que compõem essa estrutura.

Todavia, esta estrutura produz muitas vezes transmissões desnecessárias, já que um determinado nó acaba recebendo um mesmo bloco por meio de diferentes *peers*. Por este motivo, O CPoS envia na mensagem de transmissão do bloco, um filtro de Bloom (BLOOM, 1970), capaz de indicar para quais nós da rede aquele bloco já foi transmitido. Os filtros de Bloom são estruturas de dados projetadas para informar de maneira rapida e eficiente, se um determinado dado está presente em um conjunto, como representado na Fig. C.2. De acordo com a figura, um determinado valor é inserido como entrada de k funções *hash* distintas, onde a saída de cada uma delas corresponde a posição do vetor que deve ter seu valor alterado para um. Desta forma, ao final deste processo, diz-se que o valor de entrada está representado no filtro através das k posições do vetor selecionadas pelas funções *hash*.

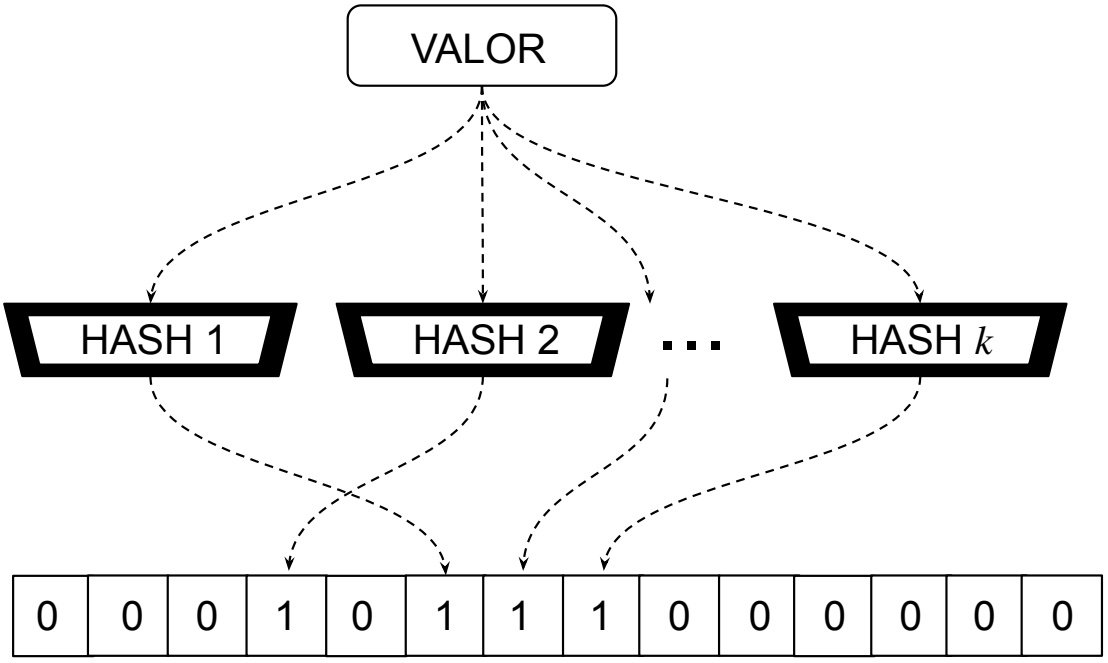

Figura C.2 – Filtro de Bloom

É possível consultar se um valor está no filtro, a partir de sua inserção como entrada no mesmo conjunto de funções *hash* utilizado no processo anterior. Caso a saída de alguma função *hash* aponte para um índice do vetor cujo valor seja zero, é possível afirmar que o elemento não está inserido no filtro de Bloom. Por outro lado, se todos as saídas das funções indicarem índices com valores iguais a um, o elemento de entrada

pode estar no filtro, já que falsos positivos são possíveis. Os falsos positivos ocorrem a partir de valores, que quando aplicados ao conjunto de funções hash, levam ao mesmo subconjunto de índices do vetor. Portanto, estes valores são representados pelos mesmos bits do vetor, provocando o falso positivo, caso algum deles não tenha sido inserido no filtro. A probabilidade de falsos positivos pode ser controlada por meio do uso de filtros com tamanhos adequados ao número previsto de entradas, e também pelo número de funções *hash* utilizadas.

Portanto, antes de executar uma nova transmissão, um nó qualquer realiza uma consulta ao filtro de Bloom que acompanha a mensagem recebida, com o objetivo de verificar quais de seus *peers* já receberam o bloco a partir de transmissões passadas. Para os *peers* já representados no filtro, o nó não envia o bloco, colaborando para que o tráfego de rede seja reduzido. Por outro lado, caso algum de seus *peers* não esteja no filtro, ele é atendido com uma nova transmissão e seu endereço é inserido no filtro que acompanha o bloco transmitido.

ANEXO D – Falhas Bizantinas

Nas blockchains públicas, o consenso deve ser alcançado mesmo na presença de nós desonestos ou que apresentem falhas. Ou seja, é considerado o modelo de falhas Bizantinas proposto por Lamport et al. (LAMPORT; SHOSTAK; PEASE, 1982). O modelo proposto descreve as falhas Bizantinas como uma condição inerente aos sistemas computacionais distribuídos, onde os processos podem falhar de modo arbitrário, não responder, ou mesmo subverter as ações que devem ser tomadas, por meio de comportamentos maliciosos que dificultam o consenso. Neste sentido, o consenso em uma blockchain pública deve ser alcançado em uma infraestrutura Bizantina desafiadora, onde as falhas são imprevisíveis e não existem relações de confiança entre os participantes.

O problema teórico dos generais Bizantinos é parte do modelo teórico proposto por Lamport et al. para avaliar o comportamento de um sistema distribuído que busca atingir o consenso sobre uma ação, em um ambiente sujeito às falhas Bizantinas. Assim, o problema considera que algumas tropas do exército Bizantino são posicionados nos arredores de uma cidade inimiga, onde a comunicação entre eles pode ser realizada apenas por meio de mensagens transmitidas por mensageiros. É assumido que cada tropa é comandada por um general que pode apresentar comportamento desonesto ou falhar (não enviar a mensagem), porém, uma vez que o general entrega a mensagem ao mensageiro, a mesma é recebida de maneira íntegra pelo general de destino. Além disso, o problema considera que as mensagens têm atraso de comunicação limitado e são enviadas apenas no modo *unicast*. A solução para o problema é encontrar um algoritmo que seja capaz de conduzir os generais a um consenso sobre atacar ou não a cidade inimiga. Em outras palavras, o algoritmo deve garantir que:

- todos os generais honestos decidam a mesma ação: atacar ou não a cidade inimiga;
- um número pequeno de traídores não seja capaz de conduzir os generais honestos a ações diferentes, fazendo com que apenas parte dos generais decida atacar a cidade.

Uma solução apresentada por Lamport et al. para o problema consiste em transmitir mensagens orais. Essa solução faz uso de três suposições:

- toda mensagem enviada é entregue corretamente pelo mensageiro;
- o general destinatário sabe quem enviou a mensagem;
- a ausência de uma mensagem pode ser detectada.

Cada general i envia uma mensagem m_i para os outros, indicando sua decisão sobre o ataque. Os generais desonestos podem mudar suas decisões a cada novo envio, buscando dificultar o consenso. Um exemplo com 4 generais é apresentado na Fig. D.1.

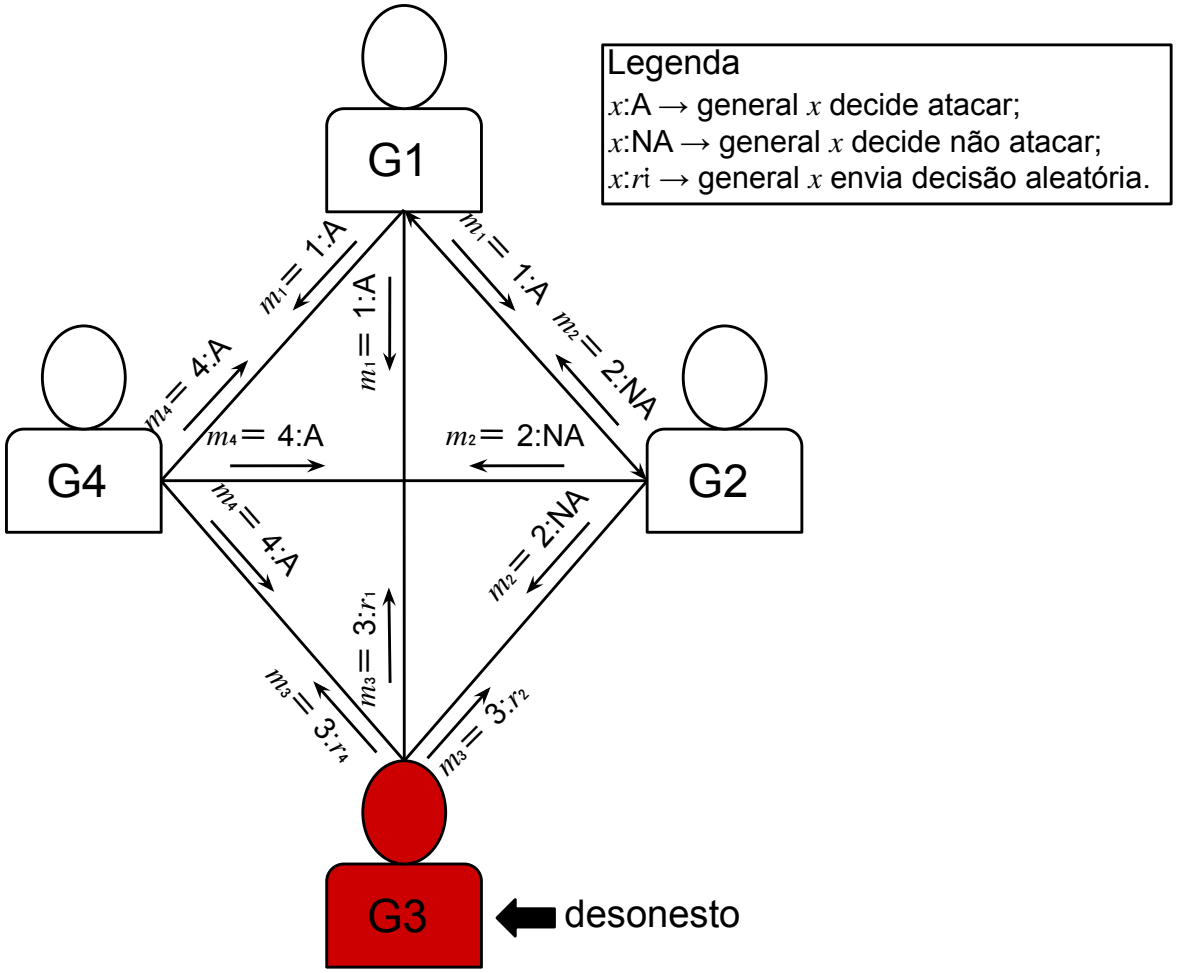

Figura D.1 – Mensagens orais

Primeiramente, cada processo (general) cria um vetor v_i com os valores recebidos durante a transmissão das mensagens, como mostrado abaixo:

$$v_1 = (1{:}\text{A}, 2{:}\text{NA}, 3 : r_1, 4{:}A)$$
$$v_2 = (1{:}\text{A}, 2{:}\text{NA}, 3 : r_2, 4{:}\text{A})$$
$$v_3 = (1{:}\text{A}, 2{:}\text{NA}, 3 : r_3, 4{:}\text{A})$$
$$v_4 = (1{:}\text{A}, 2{:}\text{NA}, 3 : r_3, 4{:}\text{A})$$

Logo após, os vetores resultantes são transmitidos entre os generais interessados no consenso. A partir do conjunto de vetores recebidos é construído o vetor resultado R, onde sua i-ésima posição é dada pelo valor que foi recebido na maioria das vezes, como mostrado a seguir:

$G1$	$G2$	$G4$
$G2$ $(1{:}\text{A}; 2{:}\text{NA}; 3{:}r_2; 4{:}\text{A})$	$G1$ $(1{:}\text{A}; 2{:}\text{NA}; 3{:}r_1; 4{:}\text{A})$	$G1$ $(1{:}\text{A}; 2{:}\text{NA}; 3{:}r_1; 4{:}\text{A})$
$G3$ $(1{:}r_5; 2{:}r_6; 3{:}r_7; 4{:}r_8)$	$G3$ $(1{:}r_9; 2{:}r_{10}; 3{:}r_{11}; 4{:}r_{12})$	$G2$ $(1{:}\text{A}; 2{:}\text{NA}; 3{:}r_2; 4{:}\text{A})$
$G4$ $(1{:}\text{A}; 2{:}\text{NA}; 3{:}r_3; 4{:}\text{A})$	$G4$ $(1{:}\text{A}; 2{:}\text{NA}; 3{:}r_3; 4{:}\text{A})$	$G3$ $(1{:}r_{13}; 2{:}r_{14}; 3{:}r_{15}; 4{:}r_{16})$
$R = (1{:}\text{A}; 2{:}\text{NA}; -; 4{:}\text{A})$	$R = (1{:}\text{A}; 2{:}\text{NA}; -; 4{:}\text{A})$	$R = (1{:}\text{A}; 2{:}\text{NA}; -; 4{:}\text{A})$

Nota-se que o vetor R mostra que os processos $G1$, $G2$ e $G4$ já estão com suas ações acordadas, independentemente das mensagens desonestadas enviadas pelo processo $G3$. Lamport et al. mostra que para um conjunto de f generais traidores, o consenso sobre a ação a ser tomada pode ser encontrado através do algoritmo descrito acima, se existirem no mínimo $3f + 1$ generais participantes.

Este resultado teórico é utilizado como premissa para as implementações práticas dos mecanismos de consenso para sistemas distribuídos, onde estão incluídos os mecanismos para blockchains públicas. Estes mecanismos são conhecidos por tolerar falhas Bizantinas e, portanto, são chamados na literatura de BFT (*Byzantine Fault Tolerance*). Neste sentido, qualquer mecanismo de consenso para blockchain pública deve ser implementado com premissas que garantam que ele seja BFT.

ANEXO E – Argumentos preliminares de prova de segurança

De acordo com Moindrot et al. (MOINDROT; BOURNHONESQUE, 2017), a segurança de um mecanismo de consenso pode ser associada à sua capacidade de evitar que dois blocos conflitantes sejam confirmados em cadeias diferentes. No CPoS, a média de sorteios bem sucedidos calculada é utilizada para evitar a confirmação de blocos conflitantes, já que essa média garante, com a probabilidade que for necessária, que não existe uma outra visão capaz de confirmar um bloco de mesmo índice na rodada x. Sendo C_1 e C_2 duas visões distintas da blockchain e assumindo que o número de sorteios bem sucedidos de uma rodada é de fato controlado pelo parâmetro τ, para que exista ${}_vB_r \in C_1$ e ${}_uB_r \in C_2$ capazes de serem confirmados na rodada x, é necessário que $\bar{s}_v^{(x)} \geq s_{min}^{(x)}$ e $\bar{s}_u^{(x)} \geq s_{min}^{(x)}$ ao final da rodada x. Além disso, $s_{min}^{(x)}$ depende do intervalo entre a rodada atual (x) e a rodada de criação do bloco, e seu valor também tem relação com τ, ou seja, podemos escrever que em uma rodada x, $s_{min}^{(x)} = \tau - \delta$. Considerando $\delta = 0$, é necessário que a soma das médias atenda à inequação E.1 para que ambas as visões confirmem os blocos ${}_vB_r$ e ${}_uB_r$ ao final da rodada x:

$$\bar{s}_v^{(x)} + \bar{s}_u^{(x)} \geq 2\tau \tag{E.1}$$

Por outro lado, as duas médias dependem do número de sorteios bem sucedidos recebidos no intervalo $[r+1; x]$, resultando em:

$$\bar{s}_v^{(x)} = \frac{s_v^{(r+1)} + ... + s_v^{(x)}}{\Delta_r} \tag{E.2}$$

$$\bar{s}_u^{(x)} = \frac{s_u^{(r+1)} + ... + s_u^{(x)}}{\Delta_r} \tag{E.3}$$

Na Eq. E.4, é mostrada a soma das Eqs. E.2 e E.3, utilizando a hipótese que o número de sorteios bem sucedidos de uma rodada é aproximadamente τ.

$$\frac{s_v^{(r+1)} + s_v^{(r+2)} + ... + s_v^{(x)}}{\Delta_r} + \frac{s_u^{(r+1)} + s_u^{(r+2)} ... + s_u^{(x)}}{\Delta_r} = \frac{\tau + ... + \tau}{\Delta_r} = \tau \tag{E.4}$$

Portanto, assumindo que o número esperado de sorteios bem sucedidos por rodada seja de fato τ, passa a ser improvável que a inequação E.1 seja satisfeita. Desta maneira, a probabilidade do conflito acontecer passa a ser controlada pelo parâmetro τ

escolhido. À medida que o intervalo de rodadas Δ_r cresce, a distribuição de probabilidade garante que a média de sorteios observada pela rede se aproxime ainda mais do valor esperado τ. Desta forma, é também improvável que $\bar{s}_v^{(x)} + \bar{s}_u^{(x)} \geq 2\tau - 2\delta$, para δ suficientemente pequeno e diferente de zero. Portanto, as premissas descritas nessa seção podem ajudar na construção de um modelo de prova mais formal, pois elas mostram que o número esperado de sorteios bem sucedido converge para τ, o que dificulta a obtenção do número mínimo de sorteios necessário para que confirmações conflitantes ocorram.

ANEXO F – Avaliação da incidência de *forks* em mecanismos de consenso baseados em cadeia

O CPoS, em suas primeiras versões, era um mecanismo de consenso que trabalhava apenas com propriedades ligadas à evolução da cadeia como forma de garantir sua segurança contra reversões. Assim como no PoW, a cadeia com o maior número de blocos era a que deveria ser seguida, pois tratava-se da cadeia que estava sendo considerada pela maioria dos nós da rede. Neste sentido, o CPoS contava apenas com as duas primeiras condições de aceitação de blocos definidas na seção 4.4 do capítulo 4. Assim, não existia o conceito de *hash* de menor valor como forma de definir qual bloco deveria permanecer na visão local de cada nó, o que colaborava para o prolongamento dos *forks* para além de uma rodada.

Este cenário inicial motivou o desenvolvimento de um modelo probabilístico para análise da ocorrência dos *forks*, o qual pode ser utilizado por qualquer mecanismo de consenso probabilístico que seja baseado no estado da cadeia (*Chain-based*), como é o caso do *Proof-of-Work* e algumas propostas iniciais do *Proof-of-Stake*.

F.1 Probabilidade de produzir c cadeias em uma rodada qualquer

Nesta seção, é definida a probabilidade de existirem exatamente c cadeias produzidas em uma rodada r qualquer. Em outras palavras, é derivada uma expressão para calcular a probabilidade de k nós serem sorteados na rodada r, o que é equivalente a calcular a existência de k cadeias, já que cada nó sorteado produz uma nova cadeia na blockchain, considerando que na versão inicial do CPoS não existe o conceito de *hash* de menor valor. Algumas premissas básicas mostradas abaixo são necessárias para tornar o problema mais tratável do ponto de vista de matemático:

- Todos os nós têm a mesma visão da blockchain.
- O número N representa a quantidade de nós que estão dentro da tolerância de tempo e, portanto, os blocos criados por eles podem gerar novos *forks*.
- As cadeias disponíveis na rodada r para mineração são apenas aquelas produzidas pelos nós sorteados na mais recente rodada que tenha tido algum nó sorteado. Neste caso, o modelo não utiliza intervalo de tolerância, ou seja, $tol = 0$.

- No início da primeira rodada analisada ($r = 0$) existe apenas uma cadeia disponível (a blockchain não contém *forks*).

Conhecer essa probabilidade é importante para o nosso modelo, pois a probabilidade de ocorrer um *fork* depende do número de cadeias válidas que os nós têm à disposição para tentar criar os novos blocos no início de cada rodada. Neste sentido, são definidas duas novas notações que suportam o cálculo da probabilidade de sorteios bem sucedidos e de *forks*:

- $P_c^{(r)}$ **:** probabilidade de haver c cadeias disponíveis no início da rodada r. De forma equivalente, é a probabilidade de ser sorteado c vezes durante a rodada $r - 1$;
- $P_f^{(r)}$ **:** probabilidade de ocorrência de *forks* na rodada r.

A probabilidade de um nó ser sorteado na rodada r é p, como já definido na seção 4.2 do capítulo 4 deste trabalho. A Fig. F.1 mostra a variação possível do parâmetro c na rodada $r = 0$ para $N = 3$.

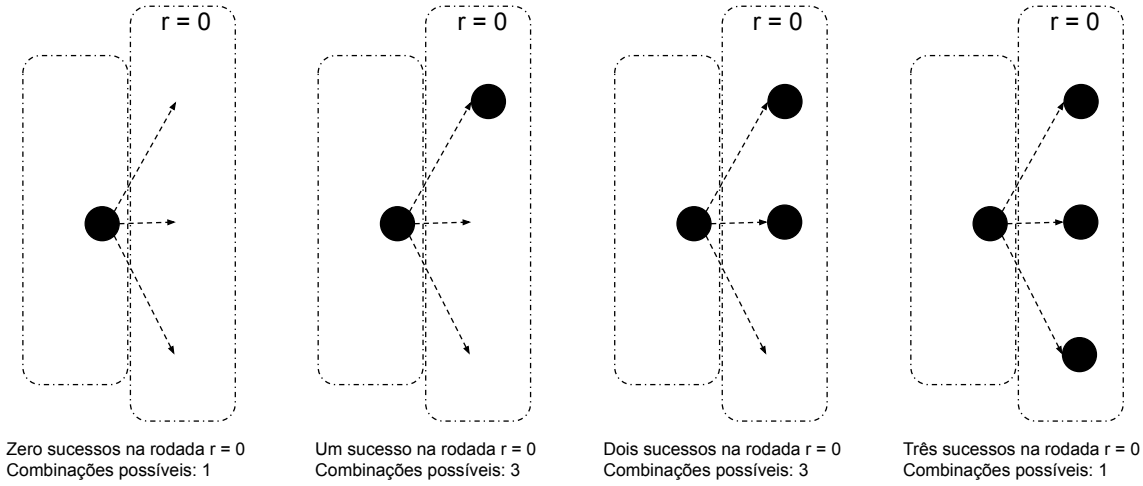

Figura F.1 – Número de sucessos possíveis na rodada inicial

A rodada $r = 0$ é a rodada inicial de análise e não será necessariamente baseada no primeiro bloco gerado (bloco gênesis). Basta que antes dessa rodada exista apenas uma cadeia disponível, como definido nas premissas deste modelo. Na rodada $r = 0$ com três nós participantes, o número de sucessos possíveis (ou cadeias) varia entre zero e três. As combinações possíveis para cada quantidade de nós sorteados são descritas abaixo de cada configuração da Fig. F.1. No início da rodada $r = 0$, com apenas uma cadeia disponível, o número de sucessos possíveis varia entre zero e N, onde N é o número de nós ($N = 3$). No início da rodada $r = 1$ espera-se, portanto, que o valor de c varie entre zero e N^2, já que para cada sorteio da rodada $r = 0$ é possível que até N nós sejam sorteados na rodada $r = 1$.

A probabilidade de ser sorteado em uma rodada qualquer depende do número de sorteios da rodada anterior, ou seja, quanto maior o número de cadeias disponíveis, maior será o número de tentativas e maior será a probabilidade de um nó ser sorteado nas próximas rodadas. Na rodada inicial ($r = 0$) sempre existe apenas uma cadeia disponível no seu início. Logo, nessa rodada, a probabilidade de se obter c novas cadeias a partir de N sorteios é dada pela Eq. F.1.

$$P_c^{(0)} = \binom{N}{c}(1-p)^{N-c}p^c \tag{F.1}$$

Em uma rodada r qualquer, a probabilidade de c sucessos muda em função do número de sucessos que foram obtidos na rodada anterior $(r-1)$. Assim, a probabilidade de ocorrência de c sucessos na rodada atual depende de cada um dos x sucessos obtidos na rodada $r-1$ com probabilidade de ocorrência $P_x^{(r-1)}$, que seja capaz de produzir c sucessos na rodada r. Nesse sentido, é necessário que o número de sucessos x gerados na rodada anterior, ofereça o mínimo necessário de cadeias para que seja possível produzir c sucessos na rodada atual. Como exemplo, na Fig. F.1, apenas quando 3 sucessos ocorrem na rodada $r = 0$ é que existe uma probabilidade diferente de zero de se obter 9 sucessos na rodada $r = 1$. Esse número mínimo de sucessos necessários na rodada anterior para que seja possível produzir c cadeias na rodada atual é dado por $S_{min} = max\left(1, \left\lceil \frac{c}{N} \right\rceil\right)$.

Seja i o número de sorteios bem sucedido (cadeias produzidas) na rodada $r-1$. É possível que c cadeias sejam geradas na rodada r se $i \geq S_{min}$ onde, para cada uma destas i cadeias, é possível obter até N novos sorteios (todos os nós participam do sorteio em todas as cadeias disponíveis), resultando em iN sorteios realizados. Assim, a nova distribuição binomial para calcular a probabilidade de c sucessos bem sucedidos quando i cadeias estão disponíveis é dada por $\binom{iN}{c}(1-p)^{iN-c}p^c$. Além disso, diferentemente do que ocorre para $r = 0$ quando sempre existe uma cadeia no início desta rodada, a ocorrência de i cadeias no início da rodada r depende da probabilidade de que i sorteios tenham ocorrido na rodada anterior ($P_i^{(r-1)}$). Portanto, a nova distribuição binomial para c sorteios bem sucedidos com i cadeias disponíveis é dada por $\binom{iN}{c}(1-p)^{iN-c}p^c P_i^{(r-1)}$.

Na rodada $r-1$, é possível que diferentes quantidades de sorteios bem sucedidos sejam capazes de produzir c cadeias na rodada r (qualquer quantidade que atenda a condição $i \geq S_{min}$). Além disso, i é no máximo N^r (na rodada $r-1$), o que permite concluir que o número i de cadeias que podem produzir c sorteios bem sucedidos na rodada r está no intervalo $S_{min} \leq i \leq N^r$. Cada i neste intervalo contribui com uma probabilidade individual e, portanto, a probabilidade total para todos os i sorteios bem sucedidos possíveis é dada por $\sum_{i=S_{min}}^{N^r}\binom{iN}{c}(1-p)^{iN-c}p^c P_i^{(r-1)}$.

Todo este desenvolvimento foi para valores não nulos de i. Mesmo para o caso de $c = 0$, o início deste somatório também será em $i = 1$, que é o menor valor que i pode assumir. Portanto, este somatório não está considerando a probabilidade de não ocorrência

de sorteios bem sucedidos na rodada $r-1$, ou seja, $P_0^{(r-1)}$. A não ocorrência de sucessos na rodada $r-1$ é um caso particular, pois faz com que todos os sorteios bem sucedidos ocorridos na rodada anterior $(r-2)$ sejam ainda válidos no início da rodada r. Essa mesma ideia é aplicada caso não existam sucessos na rodada $r-2$, onde nesse caso os sucessos da rodada $r-3$ ainda são válidos. Esta regressão continua até que seja encontrada uma rodada com algum sorteio bem sucedido ou a rodada inicial $(r=0)$.

Desta forma, a probabilidade de ocorrência de c sucessos na rodada r utiliza o número de sucessos da rodada $r-2$, ponderado pela probabilidade de nenhum sorteio bem sucedido ocorrer na rodada $r-1$ $(P_0^{(r-1)})$. Assim, o cálculo da probabilidade de c sucessos na rodada r utilizando o número de sucessos da rodada $r-2$ é dado por $P_c^{(r)}P_0^{(r-1)}$ e, $P_c^{(r)}$ com zero sorteios bem sucedidos na rodada $r-1$, é o mesmo que calcular a probabilidade $P_c^{(r-1)}$ com a rodada atual sendo r, porém com a premissa de que durante a rodada $r-1$ zero sucessos ocorreram $(P_c^{(r)}P_0^{(r-1)} = P_c^{(r-1)}P_0^{(r-1)})$. Além disso, existe a probabilidade da rodada $r-2$ também não ter nenhum sucesso, o que corresponde a calcular $P_c^{(r-2)}$ na rodada atual r, sabendo que durante as rodadas $r-1$ e $r-2$ nenhum sucesso ocorreu $(P_c^{(r-2)}P_0^{(r-2)}P_0^{(r-1)})$. Essas contribuições individuais são somadas no intervalo entre as rodadas $r-1$ e a última rodada capaz de produzir a quantidade S_{min} de sucessos para obter a probabilidade total de c sucessos na rodada r quando nenhum sorteio bem sucedido ocorre na rodada $r-1$. É necessário, portanto, definir esta rodada mínima possível. A Eq. F.2 define o termo R_{min} como sendo a rodada minima capaz de produzir os S_{min} sucessos.

$$\begin{aligned} R_{min} &= 1, \quad se\ c = 0\ ou\ c = 1 \\ R_{min} &= \lceil \log_N c \rceil, \quad se\ c > 1 \end{aligned} \tag{F.2}$$

Assim, para cada rodada $t \geq R_{min}$ é calculada a probabilidade de se obter c sucessos a partir dos sucessos desta rodada $(P_c^{(t)})$ com a respectiva probabilidade de todas as rodadas maiores que t apresentarem zero sucessos. O termo $X_c^{(r)}$ (Eq. F.3) representa a contribuição total da probabilidade de c sucessos em uma rodada $r > 0$ quando nenhum sucesso ocorre na rodada $r-1$.

$$X_c^{(r)} = \sum_{t=1}^{r-(R_{min}-1)} \left(P_c^{(r-t)} \prod_{j=1}^{t} P_0^{(r-j)} \right) \tag{F.3}$$

Deste modo, a Eq. F.4 apresenta a probabilidade de c sucessos na rodada r com N nós para $r > 0$ e $0 \leq c \leq N^{r+1}$.

$$P_c^{(r)} = X_c^{(r)} + \sum_{i=S_{min}}^{N^r} \binom{iN}{c} (1-p)^{iN-c} p^c P_i^{(r-1)} \tag{F.4}$$

F.2 Probabilidade de *fork*

A Eq. F.5 mostra o cálculo da probabilidade $\overline{p_f}$ de não ocorrência de *fork* em uma cadeia qualquer. Assim, não existe *fork* quando nenhum nó é sorteado e nenhum novo

bloco é produzido na rodada, ou quando apenas um nó é sorteado e cria o próximo bloco.

$$\overline{p_f} = (1-p)^N + \binom{N}{1}(1-p)^{N-1}p = (1-p)^N + N(1-p)^{N-1}p \tag{F.5}$$

A probabilidade de *fork* p_f em uma cadeia qualquer é dada pelo complementar de $\overline{p_f}$ ($p_f = 1 - \overline{p_f}$) e é utilizada quando existe apenas uma cadeia. Isso ocorre na rodada inicial da análise ($r = 0$).

Já a probabilidade de ocorrer pelo menos um *fork* em uma rodada r qualquer, depende do número de cadeias (c) disponíveis no início desta rodada (geradas a partir de sucessos em sorteios na rodada $r-1$). Para cada uma das c cadeias disponíveis no início da rodada r, existe uma probabilidade de *fork* associada. Portanto, a probabilidade de ocorrência de pelo menos um *fork*, considerando cada quantidade de i cadeias ($i \leq c$) é dada por $P_c^{(r-1)} \sum_{i=0}^{c-1} \binom{c}{i} (p_f)^{c-i} (\overline{p_f})^i$. O termo $P_c^{(r-1)}$ é necessário, pois o número de cadeias disponíveis (c) no início da rodada r depende da probabilidade de que na rodada $r-1$ tenham ocorrido c sorteios bem sucedidos. Deste modo, para se obter a probabilidade de *fork* na rodada r, quando pelo menos um sucesso ocorre na rodada $r-1$, é necessário variar o parâmetro c em seu intervalo possível ($1 \leq c \leq N^r$) para a rodada $r-1$, resultando em $\sum_{c=1}^{N^r} P_c^{(r-1)} \sum_{i=0}^{c-1} \binom{c}{i} (p_f)^{c-i} (\overline{p_f})^i$.

A probabilidade de *forks* total na rodada r é obtida por meio deste resultado acréscido da contribuição da probabilidade de *forks* quando nenhum sucesso é observado na rodada $r-1$, a qual é semelhante ao parâmetro $X_c^{(r)}$ já explicado anteriormente. Porém para cada rodada $t \leq r-1$, deve-se calcular a probabilidade de *fork* utilizando a probabilidade de que todas as rodadas maiores que t não obtiveram sucessos nos sorteios. O parâmetro $Y^{(r)}$ (Eq. F.6) mostra a parcela da probabilidade de *forks* na ausência de sucessos na rodada anterior.

$$Y^{(r)} = p_f \prod_{j=1}^{r} P_0^{(r-j)} + \sum_{t=1}^{r-1} \left(P_f^{(r-t)} \prod_{j=1}^{t} P_0^{(r-j)} \right) \tag{F.6}$$

O primeiro termo da Eq. F.6 é a parcela limite, ou seja, é a condição onde nenhuma rodada no intervalo $0 \leq t < r$ produziu bloco e, portanto, existe apenas a cadeia do início da análise. O segundo termo corresponde a soma das probabilidade de *fork* de cada rodada intermediária t, ponderada pela probabilidade de todas as rodadas maiores que t não produzirem blocos. Deste modo, a probabilidade total de *forks* na rodada r é calculada na Eq. F.7.

$$P_f^{(r)} = Y^{(r)} + \sum_{c=1}^{N^r} P_c^{(r-1)} \sum_{i=0}^{c-1} \binom{c}{i} (p_f)^{c-i} (\overline{p_f})^i \tag{F.7}$$

F.3 Aplicação do modelo na primeira versão do CPoS

Os resultados apresentados nessa seção foram obtidos através de uma implementação do *CPoS* sobre a plataforma *Mininet*. O *Mininet* é um emulador de redes criado

na linguagem Python com o objetivo principal de ser um ambiente para validar redes definidas por software, permitindo a utilização de switches que implementam o protocolo OpenFlow e que podem ser gerenciados por controladores externos. Os controladores são responsáveis pela implementação dos fluxos nas interfaces dos switches que participam da comunicação. Nessa emulação, cada nó da rede do mecanismo *CPoS* é um host conectado a uma interface de um switch OpenFlow, como mostrado na Fig. F.2.

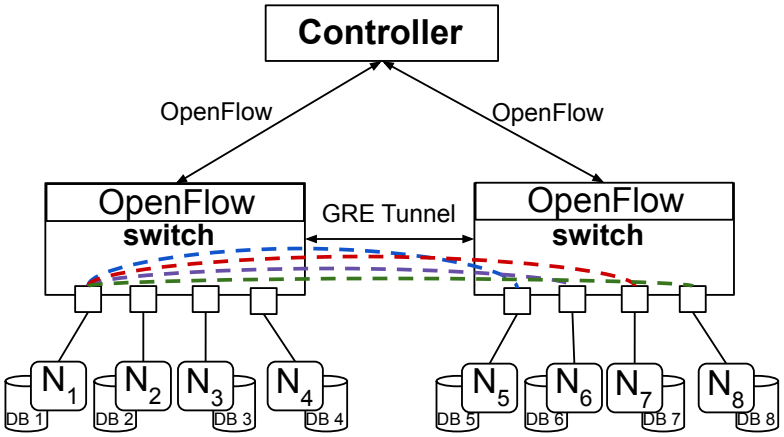

Figura F.2 – Arquitetura de rede do experimento prático

Nessa topologia, cada nó está conectado com todos os outros, o que proporciona uma comunicação direta entre o nó que produz o bloco e o restante da rede. Cada nó executa uma instância do *CPoS* e armazena os blocos no seu banco de dados local. Como o *open vswitch* utilizado não se mostrou adequado para muitos nós conectados (acima de 200), foi necessário utilizar mais switches para conexões de mais nós. Estes switches foram alocados em computadores diferentes e interconectados utilizando o protocolo GRE (*Generic Routing Encapsulation*) e, com isso, a quantidade de nós pôde ser aumentada, como mostrado na figura. F.2.

Os nós sempre tentam criar o próximo bloco em todas as cadeias que eles enxergam como disponíveis e válidas nas suas visões locais. Assim, estes primeiros resultados permitem uma análise de pior caso em relação ao número de *forks*, já que não existe nenhum custo associado com as criações em múltiplas cadeias. Isto permite que um determinado nó possa produzir um bloco para cada cadeia em uma mesma rodada, dificultando a convergência em torno de uma única cadeia, pois todas elas podem crescer com igual probabilidade.

Esse problema é conhecido na literatura como *nothing-at-stake* e ocorre quando não existe um custo associado às ações tomadas por um participante no mecanismo, como por exemplo, escolher uma cadeia entre todas as possíveis para propor o próximo bloco. Quando o *nothing-at-stake* é controlado, não se permite que o nó faça uso irrestrito do seu recurso (*stake*), definindo punições ou reduzindo o poder de mineração dos nós que não estiverem de acordo com o mecanismo proposto.

F.4 Criação de *forks*

A Fig. F.3 apresenta os resultados práticos e teóricos utilizando a topologia de teste mostrada na Fig. F.2 e o modelo da seção F.1.

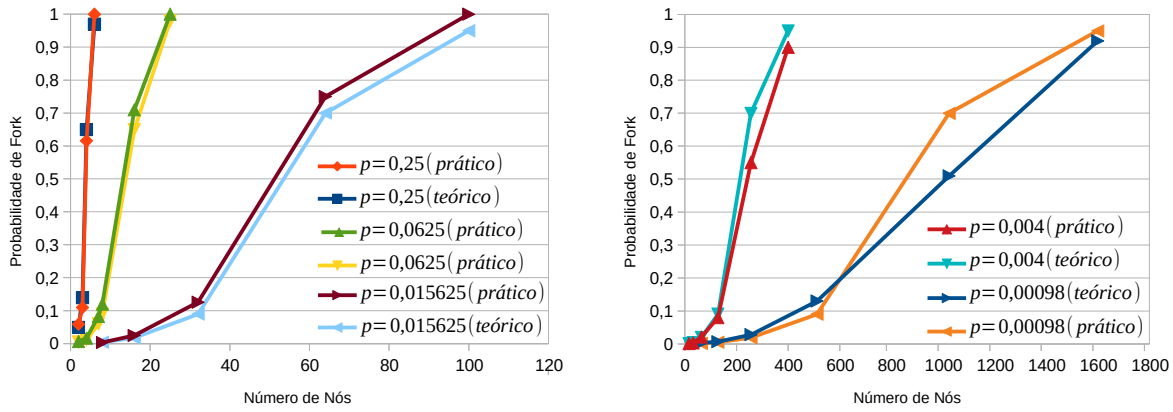

Figura F.3 – Probabilidade de forks para diferentes valores de p

Os testes foram executados até que 100 blocos da blockchain local de cada um dos nós fossem confirmados sem a utilização de intervalo de tolerância ($tol = 0$), conforme definido na seção F.1. Os resultados permaneceram próximos do valor teórico esperado para todos os desafios analisados. Além disso, é possível ver que são necessários aproximadamente quatro vezes mais nós para produzir a mesma probabilidade de *forks* a partir de um redução de quatro vezes na probabilidade de sorteio p.

Neste sentido, é importante conhecer a probabilidade de *forks* em função do número de nós para que a probabilidade de sucessos p seja calibrada, em função dos parâmetros do mecanismo, com o objetivo de minimir os *forks*, sem contudo deixar a blockchain ociosa. Este modelo pode ser empregado em mecanismos de consenso que utilizam o Proof-of-Stake e o consenso alcançado é baseado no estado da cadeia.

ANEXO G – Artigos Publicados

G.1 Artigos derivados desta pesquisa

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, Workshop WBlockchain.* Gramado - RS, 2019.

MARTINS, D. F. G.; HENRIQUES, M. A. A. Uma análise preliminar do controle de *forks* no mecanismo de consenso Proof-of-Stake com Tempo Discreto. In: *XIX Simpósio Brasileiro de Segurança da Informação e de Sistemas Computacionais* São Paulo - SP, 2019. **Artigo premiado com Menção Honrosa na trilha principal do evento.**

MARTINS, D. F. G; HENRIQUES, M. A. A. Avaliação da incidência de *forks* no algoritmo de consenso Probabilistic Proof-of-Stake (PPoS). In: *XXXVIII Simpósio Brasileiro de Redes de Computadores e Sistemas Distribuídos, Workshop WBlockchain.* Rio de Janeiro - RJ, 2020. **Artigo premiado como o melhor trabalho apresentado no evento**.