Uma proposta para confirmação de blocos sem comitês de validação em mecanismo de consenso Proof-of-Stake para blockchains públicas

Diego Fernandes Gonçalves Martins,
Marco Aurélio Amaral Henriques

[1]Departamento de Engenharia de Computação e Automação Industrial (DCA)
Faculdade de Engenharia Elétrica e de Computação (FEEC)
Universidade Estadual de Campinas (Unicamp)
13083-852 – Campinas, SP, Brasil

{diegofgm,marco}@dca.fee.unicamp.br

Abstract. *This paper presents a new method for public blockchain consensus mechanism based on Proof-of-Stake that does not need validation committees to accept a new block. First, the text explains the mechanism principles and how a node is selected to create a block in the chain. Then, it presents the probabilistic confirmation process as a way to make the acceptance of a block definitive without validation committees, along with practical results that evaluate the performance and security of the mechanism. The method security is demonstrated by a theoretical study of block reversion probability.*

Resumo. *Este artigo apresenta um novo método de consenso para blockchains públicas baseado em Proof-of-Stake que não requer comitês de validação para aceitar novos blocos. O texto mostra os princípios do mecanismo e como um nó é selecionado para criar um bloco na cadeia. Em seguida é apresentado o processo de confirmação probabilística como forma de tornar definitiva a aceitação de um bloco sem comitês de validação e resultados práticos que avaliam o desempenho e a segurança do mecanismo. A segurança do método é demonstrada por um estudo teórico da probabilidade de reversão de blocos.*

1. Introdução

Os algoritmos de consenso para blockchains permitem que uma determinada transação entre duas partes seja validada sem que exista uma terceira parte confiável [Christidis and Devetsikiotis 2016]. Ele pode ser definido como o conjunto de passos que são dados pelos nós que compõem a rede para entrar em consenso sobre um determinado valor [Bashir 2017]. As blockchains públicas são estruturas formadas por participantes que não possuem relação de confiança uns com os outros e que precisam tomar decisões conjuntas para que seja possível atingir o consenso sobre qual é o próximo bloco produzido que deve ser adicionado à cadeia [Tschorsch and Scheuermann 2016]. Além disso, este tipo de blockchain opera em um tipo de rede totalmente assíncrono, ou seja, não existe um tempo estabelecido para que os nós enviem suas mensagens.

Fischer et al. [Fischer et al. 1985], mostra não ser possível encontrar o consenso de forma determinística em uma rede totalmente assíncrona com a presença de nós maliciosos. Neste sentido, os mecanismos de consenso podem definir um grau de sincronismo para a rede e trabalhar com o conceito de consenso determinístico ou implementar

o conceito de consenso probabilístico [Greve et al. 2018]. Os mecanismos de consenso determinístico enfraquecem o problema, exigindo que a rede seja síncrona para que o consenso possa de fato ser alcançado. É comum a utilização de comitês de validação em consensos determinísticos, onde além de exigir que os votos sejam enviados à maioria dos nós, os membros do comitê estão sujeitos a tentativas de subornos, o que pode dificultar o estabelecimento do consenso. O Algorand [Gilad et al. 2017], Tendermint [Kwon 2014] e Ouroboros [Kiayias et al. 2017] são alguns dos principais representantes dos consensos determinísticos que estão baseados em comitês para confirmar seus blocos.

Já os consensos probabilísticos podem sofrer uma reversão de decisão passada, mas com uma probabilidade controlável e mantida em níveis desprezíveis. O *Proof-of-Work* (PoW) é o mecanismo probabilístico mais conhecido, onde os nós tentam criar o próximo bloco através de seus esforços computacionais [Nakamoto 2009]. Sua principal desvantagem está ligada a problemas de desperdício de energia elétrica em sua operação, já que os nós precisam trabalhar em alto desempenho e apenas o trabalho de um deles é aproveitado. Estes fatos despertaram o interesse para outros mecanismos, onde a prova não esteja alicerçada em esforços computacionais. Entre as abordagens, destaca-se o *Proof-of-Stake* (PoS), onde a prova de posse do *stake* é utilizada pelo nó para tentar criar novos blocos [King 2013]. A prova de posse consiste em utilizar recursos do próprio nó como, por exemplo, a quantidade de moedas associadas a um endereço em seu controle para garantir que o nó pode propor e validar blocos.

Este trabalho apresenta um mecanismo de consenso probabilístico baseado em Proof-of-Stake, que permite a confirmação de um bloco sem a utilização de comitês de validação. O mecanismo propõe uma versão modificada do sorteio criptográfico proposto no mecanismo Algorand [Gilad et al. 2017]. Além disso, ele traz uma estratégia de confirmação de blocos totalmente nova que é baseada em um cálculo probabilístico que todo nó participante realiza a partir dos blocos recebidos em cada rodada de produção. Assim, primeiramente é explicado como o mecanismo Algorand é capaz de confirmar blocos por meio de um consenso imediato que utiliza comitês de validação, com o objetivo de esclarecer suas principais vantagens e desvantagens. Em seguida, o novo mecanismo de consenso para blockchains públicas é apresentado, mostrando quais são os critérios de seleção para que um novo bloco seja indexado à cadeia e em como é realizada a confirmação probabilística. Por fim, são apresentadas as diferenças para o mecanismo Algorand e resultados que mostram o desempenho do mecanismo para diferentes níveis de confiabilidade contra reversão do consenso obtido.

2. O mecanismo Algorand

Algorand proposto por Gilad et al. [Gilad et al. 2017] é um mecanismo que implementa o conceito de comitê de validação de blocos usando funções aleatórias verificáveis. Essas funções permitem que qualquer participante da rede possa verificar se foi sorteado para criar o próximo bloco, 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 é o vencedor a partir da formação de um comitê de validação dinâmico e aleatoriamente definido. Assim, o consenso do Algorand ocorre em duas fases distintas a cada nova rodada de produção.

Na primeira fase os nós realizam o sorteio com o objetivo de verificar se podem

criar e divulgar um novo bloco na rodada atual. Já na segunda fase, os nós realizam o sorteio criptográfico para verificar se estão aptos a participar do comitê de validação, ou seja, se são membros do subconjunto de nós que irão votar no bloco que deve ser mantido (confirmado) na blockchain. O sorteio criptográfico garante 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. Na fase 2, apenas um bloco entre todos os que foram propostos durante a fase 1 deve ser escolhido para ser inserido na blockchain, por meio dos votos emitidos pelo comitê de validação. Assim, os nós participam novamente do sorteio criptográfico, porém agora com o objetivo de verificar se são considerados sorteados para compor tal comitê, o qual deve eleger o bloco a ser indexado à blockchain a partir do envio de votos para a rede. As chances de um nó ser selecionado para compor o comitê de validação são proporcionais à quantidade de *stake* que eles controlam através de suas chaves privadas, que estão associadas a suas identidades (endereços). Por fim, quando $2/3$ dos votos emitidos pelo comitê são na direção de um mesmo bloco, ele é confirmado e inserido na blockchain.

O Algorand é parte de um conjunto de propostas que buscam alcançar o consenso a partir de provas baseadas em PoS com a utilização de comitês de validação. Assim, suas principais vantagens estão na sua capacidade de aumentar consideravelmente a taxa de confirmação de blocos quando comparado ao PoW do Bitcoin, e também na eliminação do esforço computacional como estratégia de prova. Por outro lado, o Algorand é um mecanismo complexo do ponto de vista de implementação, já que utiliza um modelo de comitê de validação descentralizados que deve convergir a cada nova rodada para que um novo bloco seja confirmado na blockchain. Essa convergência pode ser dificultada na presença de nós desonestos que empregam seus esforços (*stake*) com a intenção de atrasar a decisão do comitê. Esses nós desonestos atingem esse objetivo quando votam em blocos diferentes daquele que foi escolhido pelos nós honestos, ou quando eles dificultam a propagação dos votos na rede, contribuindo para que alguns nós honestos não recebam todos os blocos produzidos, o que é conhecido na literatura como *Eclipse attack* [Heilman et al. 2015]. Devido a essas desvantagens associadas ao uso de comitês de validação distribuídos, o novo mecanismo apresentado neste trabalho, torna-se uma alternativa, já que não requer o de uso comitês de validação em nenhuma de suas fases.

3. Um novo mecanismo de consenso PoS sem comitês de validação

Esta seção apresenta as ideias que estruturam o novo mecanismo de consenso proposto e, por questões de espaço, mostra de maneira intuitiva as principais ideias de como o consenso PoS baseado em rodadas de execução com período fixo foi adaptado para este novo esquema. Um artigo científico estendido está em preparação, onde o funcionamento do mecanismo é descrito mais detalhadamente. A seção 3.2 explica como funciona um sorteio criptográfico e como ele foi modificado para trazer as características que nosso novo mecanismo de consenso exigia. A seção 3.3 destaca o importante conceito de prioridade entre *hashes*, fator determinante para se alcançar o consenso no mecanismo. Na seção 3.4 são definidos os critérios para aceitação de blocos propostos pelos nós e a se-

ção 3.5 completa toda a descrição do novo mecanismo, mostrando como um esquema de confirmação probabilística de blocos pode ser estruturado de forma a dispensar o uso de comitês de validação.

3.1. Características básicas

O mecanismo de consenso Proof-of-Stake sem comitês de validação é um algoritmo baseado em sorteios que ocorrem em rodadas de tempo finito com duração T (neste trabalho T é considerado maior que o tempo gasto por qualquer mensagem para trafegar entre dois nós quaisquer da rede). Neste sentido, o mecanismo é síncrono; a cada intervalo de tempo T um novo ciclo e uma nova rodada se iniciam. Os nós participantes devem tentar gerar o próximo bloco sempre dentro da rodada atual de acordo com sua referência de tempo local. Caso o participante não vença o desafio, ele aguarda o início de uma próxima rodada para tentar criar o bloco novamente. Qualquer nó pode verificar se a rodada r presente no cabeçalho de um novo bloco recebido é igual à rodada esperada, calculada, utilizando a Eq. (1). O quociente desta equação fornece o número de rodadas inteiras transcorridas entre o tempo de criação do primeiro bloco produzido (gênesis) e o bloco proposto (corrente) que deve ser verificado.

$$r_{exp} = \left\lfloor \frac{t_{cb} - T_0}{T} \right\rfloor \tag{1}$$

Os parâmetros utilizados na Eq. (1) são definidos abaixo:

- 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 uso do tempo de chegada é uma forma de permitir que um nó faça os cálculos usando somente suas referências de tempo e evita que um nó destinatário tenha que confiar no tempo de criação do bloco de acordo com outro relógio (o bloco pode ter sido criado em um nó com relógio desajustado).

3.2. Sorteio criptográfico

O algoritmo de sorteio utilizado é uma versão modificada daquele presente no mecanismo Algorand [Gilad et al. 2017]. Desta forma, são utilizados os conceitos de sorteio aleatório baseados em hashes criptográficos, onde em cada rodada do sorteio novos nós são selecionados para produzir o próximo bloco da blockchain. Em uma rodada r qualquer, o nó i calcula o hash do nó $(U_i^{(r)})$. Como este valor é a saída de uma função de hash criptográfico (H), ele pode ser considerado praticamente aleatório e representar o nó i dentro de uma rodada. O cálculo deste hash utiliza como entradas a rodada atual (r), o ID exclusivo do nó (ID_i) e o resultado do hash do bloco confirmado no período mais recente (H_{fc}). No mecanismo proposto, um período é formado por um intervalo de rodadas, onde um número fixo de blocos são produzidos e confirmados. O cálculo de $(U_i^{(r)})$ é apresentado na Eq. (2) por meio da saída de um hash de 256 bits. Os três parâmetros e a saída da função hash são colocados no cabeçalho do bloco, caso o nó i seja sorteado para criá-lo na rodada r.

$$U_i^{(r)} = H(ID_i||r||H_{fc}) \tag{2}$$

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 $1 \leq j \leq w_i$. Cada valor de j representa uma oportunidade do nó i ser sorteado dentro da rodada r, como será mostrado mais adiante. A soma W de todos os *stakes* 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 (Eq. 3) que representa o valor esperado de sorteios bem sucedidos por rodada para toda a rede.

$$\tau = p \times W \tag{3}$$

Portanto, a probabilidade de haver exatamente k sucessos na rodada r em algum ponto da rede segue uma distribuição binomial que pode ser calculada por $\binom{W}{k} p^k (1-p)^{W-k}$, onde p é a probabilidade de sucesso.

A partir do valor $U_i^{(r)}$ e do *stake* w_i, cada nó i realiza localmente o sorteio criptográfico na rodada r por meio do Algoritmo 1, adaptado a partir da versão original apresentada por Gilad et al. [Gilad et al. 2017]. Essa nova versão do algoritmo deixa de utilizar funções verificáveis (VRFs), pois é possível definir um parâmetro secreto para o sorteio, a partir da própria assinatura do *hash* do nó ($\sigma(U_i^{(r)}, Pr)$), utilizando a chave privada Pr do nó i. Assim, apenas o nó i é capaz de produzir todos os parâmetros necessários para a realização do sorteio, já que ele é o único nó capaz de realizar tal assinatura. O objetivo do Algoritmo 1 é determinar o número de sorteios bem sucedidos para o nó i. No início é calculado um número pseudo-aleatório q entre 0 e 1, que depende exclusivamente de $\sigma(U_i^{(r)}, Pr)$ 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

Algorithm 1: Sorteio $(\sigma(U_i^{(r)}, Pr), w_i)$

$p \leftarrow \frac{\tau}{W}$
$q \leftarrow \frac{\sigma(U_i^{(r)}, Pr)}{2^{256}-1}$
$j \leftarrow 0$
while $q \geq \sum_{k=0}^{j} B(k, w_i, p)$ **do**
  $j++$
end while
return j

cada subintervalo é determinada pela probabilidade de até j sorteios bem sucedidos ocorrerem em w_i tentativas. No Algoritmo, $B(k, w_i, p) = \binom{w_i}{k} p^k (1-p)^{w_i - k}$ é a probabilidade de ocorrerem exatamente k sorteios bem sucedidos em w_i tentativas.

Para cada um dos j sorteios bem sucedidos é necessário calcular $H_t = H(U_i^{(r)} || t)$, ou seja, um segundo hash chamado de hash de prova. O parâmetro t é um número inteiro definido no intervalo $1 \leq t \leq j$. O hash de prova vincula o nó $(U_i^{(r)})$ a cada um dos j sorteios bem sucedidos. Qualquer nó da rede pode confirmar a validade de um hash de prova. Para isso, os nós devem executar o Algoritmo 1 com os parâmetros de entrada iguais aos utilizados pelo nó i (nó que criou o bloco). O *stake* w_i deve ser obtido por meio de uma consulta ao saldo atual do endereço associado a identidade ID_i indicada pelo nó i. Para definir seu *stake*, é necessário que o nó i transfira uma quantidade de moedas

para o endereço ID_i em cada período que ele tenha interesse de participar da produção de novos blocos. Desta forma, é possível confirmar se o nó i realmente possui a quantidade de moedas w_i utilizadas como *stake* no momento da criação do bloco.

Portanto, a partir da execução do Algoritmo 1 qualquer nó da rede é capaz de produzir j hashes de prova e calcular o hash $H_t^{'(r)}$. Assim, caso $H_t^{'(r)} = H_t^{(r)}$, para algum t, o hash de prova $H_t^{(r)}$ do bloco produzido pelo nó i é considerado válido. Caso o nó seja sorteado, ele deve transmitir o bloco juntamente com os parâmetros utilizados para a criação do hash de prova ($H_t^{(r)}$) e do hash do nó ($U_i^{(r)}$). Além disso, é necessário calcular um terceiro hash chamado de hash do bloco (H_{block}) (Eq. (4)).

$$H_{block} = H(H_t^{(r)}||H_{prev}||M_{tree}) \tag{4}$$

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].

3.3. Prioridade do hash de prova

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 de prova são válidos, já que foram produzidos a partir de algum elemento t no intervalo definido. A possibilidade de o nó ser sorteado mais de uma vez, aumenta sua probabilidade de produzir o próximo bloco; entretanto, o nó deve escolher apenas um hash de prova entre os possíveis. 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. 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 nó 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, o nó pode divulgar na rede apenas o seu hash de prioridade, descartando todos os outros que poderiam ser criados utilizando outro elemento t.

É possível obter, a partir de um modelo teórico e experimentos práticos, a probabilidade de produção de *forks* no novo mecanismo proposto aqui. Um *fork* ocorre quando dois blocos são produzidos na mesma rodada e são sucessores de um mesmo bloco da blockchain. Portanto, os *forks* produzem visões momentâneas de duas ou mais cadeias produzidas pela chegada de dois ou mais blocos com esta característica. Os *forks* produzem períodos de indeterminação no mecanismo e, durante estes períodos, nenhuma nova transação pode ser confirmada. Como consequência, o desempenho do algoritmo em relação ao número de transações confirmadas por segundo é prejudicado. Assim, a partir do conceito de hash de prioridade, os *forks* podem ser resolvidos de maneira quase instantânea, como será mostrado mais adiante. Neste trabalho, um bloco será representado pela notação ${}_vB_r$, onde os subíndices indicam:

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

Quando dois blocos (${}_uB_r$ e ${}_vB_r$) produzidos em uma rodada r caracterizam um *fork*, ele pode ser resolvido utilizando a rodada de criação e o hash de prova dos blocos. Como as rodadas dos blocos recebidos na rodada r são iguais, o bloco que possuir o hash de prova com menor valor vence a disputa e o outro bloco pode ser eliminado ainda durante a rodada r, adotando um critério similar ao de hash de prioridade. Sendo assim, na próxima seção será apresentado como aplicar os critérios de aceitação aos novos blocos recebidos em cada rodada, para decidir se eles devem ser aceitos ou rejeitados.

3.4. Gerenciamento de blocos

Nesta seção os critérios de aceitação de um bloco são definidos com base na definição do hash de prioridade. A premissa fundamental para aceitação de um bloco B é de garantir que sua rodada seja compatível com a rodada esperada pelo nó e que seu hash de prova tenha menor valor em relação ao hash de prova do bloco de mesma posição na blockchain, caso ele exista. A posição k de um bloco é um valor inteiro que representa a sua altura na blockchain. Também é possível definir um intervalo de tolerância para aceitação de blocos, ou seja, um nó pode aceitar rodadas que estejam deslocadas da sua rodada atual esperada calculada por meio da Eq. 1, para que sejam aceitos blocos a partir de nós que estão com diferenças de tempo em relação ao tempo real. O intervalo de tolerância é do tipo $[x; x + tol]$, onde tol é um inteiro que representa o número máximo de rodadas deslocadas para que o bloco ainda seja aceito quando a rodada atual esperada é x. Sendo assim, um bloco B com rodada de criação r, hash de prova v e posição k, será aceito na rodada atual esperada x se qualquer uma das três condições a seguir for atendida:

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

Caso um bloco de índice k atenda qualquer uma das condições acima ele é aceito e caso exista um bloco aceito anteriormente durante a rodada r ele é removido da visão da blockchain do nó. No diagrama de estados da Fig. 1 é mostrada a funcionalidade *listen* para obter novos blocos, desde o recebimento do bloco através do *socket* conectado ao *peer* até sua aceitação ou 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 as informações do bloco recebido 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 está sincronizado com a blockchain, ou seja, se ele conhece todos os blocos até este novo bloco recebido. Caso ele não conheça o antecessor do novo bloco ele é rejeitado no estado *Rejecting block*. No estado *Validating block round* é calculada a rodada esperada do nó utilizando a função *calcExpectedRound()*. Esta função calcula a rodada esperada (atual) a partir da Eq. (1) utilizando o relógio local do nó. Caso a rodada não seja válida, o diagrama segue para o estado *Rejecting block* e o bloco não é aceito. Para uma rodada válida é verificado se alguma das três condições de aceitação definidas acima é satisfeita por meio da execução da função *checkAcceptanceBlock* no estado *Validating block acceptance*. Por fim, o bloco é inserido na blockchain no estado *Inserting block* caso a função retorne que esta premissa foi satisfeita.

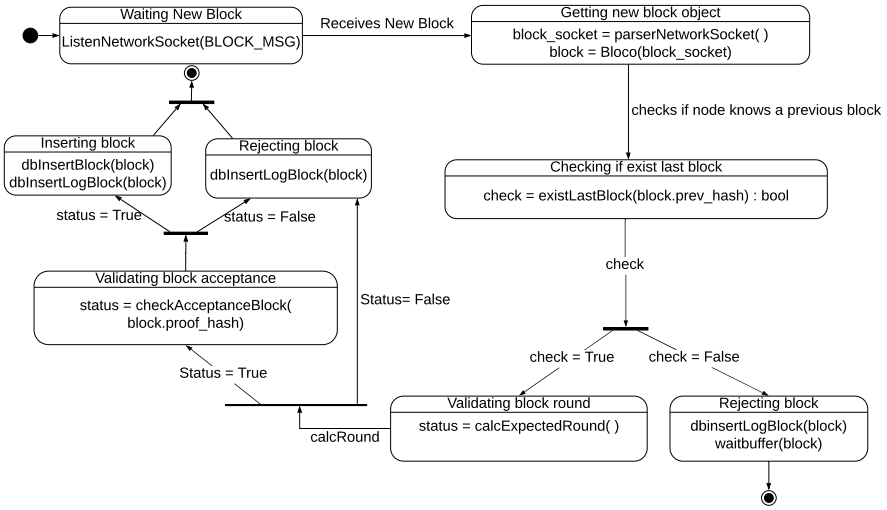

Figura 1. Diagrama do processo de recebimento de um bloco

Os critérios definidos nesta seção colaboram para que o mecanismo proposto alcance uma convergência, na qual a maioria dos nós possam concordar sobre a visão da blockchain. Porém, os critérios desta seção são aplicados pelos nós apenas aos blocos recebidos 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 instantânea 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 próxima seção.

3.5. Confirmação probabilística de um bloco no novo mecanismo

As condições de aceitação descritas na seção 3.4 são baseadas apenas na visão local do nó, ou seja, cada nó da rede toma suas decisões de aceitação apenas considerando os blocos que são recebidos por eles. Em um cenário ideal, onde todos os blocos são recebidos por todos os nós, estas condições de aceitação fornecem uma confirmação determinística para o bloco, haja vista que as decisões são tomadas considerando todos os blocos que foram produzidos. Todavia, em uma situação real de operação, as redes possuem características bizantinas e, por isso, alguns blocos podem não ser recebidos por todos os nós da rede. Desta forma, não há garantia de que os blocos que prevaleceram sobre os outros sejam de fato os melhores e não serão revertidos. Nesta seção será definido o conceito de confirmação probabilística do mecanismo proposto, que associa uma probabilidade de reversão a um bloco da blockchain, assumindo que podem existir blocos produzidos em uma rodada qualquer que sejam desconhecidos pelo nó i. Esse conceito probabilístico é capaz de produzir o consenso em relação ao bloco ${}_vB_r$ quando a probabilidade do acordo ser revertido diminui com a produção de novos blocos, o que ocorre em rodadas subsequentes àquela onde o bloco em questão foi produzido. É esperado que a probabilidade de reversão em questão diminua sempre que o bloco ${}_vB_r$ compor a blockchain que é seguida pela maioria dos nós da rede.

De maneira mais formal dizemos que o bloco ${}_vB_r$ pode ser revertido em uma rodada atual x ($x > r$), quando existe uma cadeia paralela em que cada um de seus blocos atende qualquer um dos critérios de aceitação descritos na seção anterior. Quando um nó i

recebe (ou cria) um bloco qualquer ele ganha informações a respeito do nó que o produziu e, assim, é possível para o nó i saber quantas vezes o nó criador do bloco foi sorteado por meio da execução do Algoritmo 1, como explicado na seção 3.2. Ampliando este conceito para todos os blocos recebidos pelo nó i dentro da rodada r, o nó pode calcular a partir da Eq. (5) a probabilidade Q_r de existirem sorteios bem sucedidos desconhecidos por ele na rodada r.

$$Q_r = 1 - \sum_{k=0}^{s_r} \binom{W}{k} p^k (1-p)^{W-k} \tag{5}$$

O termo s_r da Eq. (5) é o número total de sucessos conhecidos pelo nó i na rodada r através dos blocos que foram recebidos ou produzidos durante esta rodada. Quanto mais sucessos são conhecidos pelo nó i maior será o valor do somatório e, consequentemente, menor será a probabilidade Q_r de existência de blocos desconhecidos. Isto garante que o mecanismo seja capaz de alcançar o consenso probabilístico, pois o número de sorteios bem sucedidos (sucessos) alcançado pelos nós em cada rodada é calibrado pelo número esperado de sucessos (τ). Assim, não é esperado que essa quantidade de sucessos se afaste de τ e, portanto, é possível afirmar que um nó pode medir a confiabilidade do bloco mantido na sua visão local da blockchain, utilizando a Eq. 5 e o número de sucessos recebidos (s_r) em cada rodada.

A Fig. 2 mostra a visão do nó i quando ele está no início de três diferentes rodadas: (a) $r+1$, (b) $r+2$ e (c) $r+4$. Na visão (a) da Fig. 2 o nó i está no início da rodada $r+1$ e, durante a rodada r, ele recebeu o bloco ${}_vB_r$. A partir deste bloco, ele tem a informação que o nó que o criou foi sorteado uma vez ($s_r = 1$). Assim, o nó i sabe que na rodada r foi produzido ao menos um bloco e, para que existam blocos que ele não conheça, é necessário que pelo menos dois sorteios bem sucedidos tenham ocorrido durante a rodada r. Neste caso Q_r é, portanto, equivalente à probabilidade do número de sucessos na rodada r ser pelo menos dois, onde a probabilidade de alguém ser sorteado é p. Assim, na rodada r temos $Q_r = 1 - (1-p)^W + Wp(1-p)^{W-1}$. A visão (b) da Fig. 2, mostra o nó i no início da rodada $r+2$ e também que durante a rodada $r+1$ ele recebeu mais um bloco produzido por algum nó que foi sorteado apenas uma vez ($s_{r+1} = 1$). Portanto, calculando Q_{r+1} da mesma forma, temos $Q_r = Q_{r+1}$. Na rodada $r+2$, mostrada na visão (c) da Fig. 2 o nó não recebeu nenhum novo bloco e, para que exista pelo menos um bloco que ele não tenha recebido nesta rodada, basta que tenha ocorrido pelo menos um sorteio bem sucedido, o que resulta em $Q_{r+2} = 1 - (1-p)^W$. Já na rodada $r+3$ o nó sabe da existência de dois sorteios bem sucedidos ($s_{r+3} = 2$).

Este número pode ser resultado de dois sorteios bem sucedidos pelo nó criador do bloco ${}_cB_{r+3}$ e, neste caso, apenas este bloco foi recebido pelo nó i, porém existe ainda a possibilidade de que estes sucessos sejam de dois nós diferentes sorteados na rodada $r+3$ apenas uma vez cada. Neste segundo caso, são produzidos dois blocos e um deles é excluído pelo nó i, pois tem um hash de prova maior. Entretanto, antes da exclusão, ele soma o sorteio que originou este bloco no seu número de sorteios bem sucedidos conhecido, resultando em $s_{r+3} = 2$ e $Q_{r+3} = 1 - ((1-p)^W + Wp(1-p)^{W-1} + \binom{W}{2} p^2 (1-p)^{W-2})$. Em linhas gerais, em uma rodada r qualquer a probabilidade de que exista mais sucessos além dos conhecidos e, portanto, mais blocos desconhecidos pelo

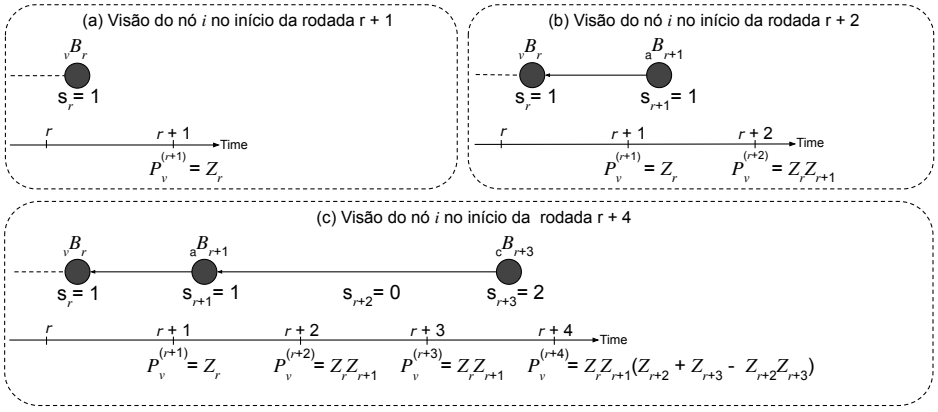

Figura 2. Número de sucessos recebidos por um nó em um intervalo de rodadas

nó i, determina a probabilidade de reversão de um bloco. A probabilidade Q_r é apenas uma medida da possibilidade de existir mais nós sorteados na rodada r que o nó i não conheça. Assim, ela não é a probabilidade de existir um bloco desconhecido na rodada r cujo hash de prova seja menor que o hash de prova conhecido pelo nó i na rodada r. Para determinar esta probabilidade, é necessário considerar a razão entre o valor do hash de prova conhecido e o número total de hashes de provas que podem ser produzidos (2^{256}). Considerando a Fig. 2, na rodada r um bloco desconhecido pode existir com probabilidade Q_r e a probabilidade de seu hash de prova ser menor é $(v-1)/2^{256}$. Neste sentido, a Eq. (6) generaliza a probabilidade de um bloco desconhecido existir e seu hash de prova ser menor que o conhecido pelo nó i. Na Eq. (6), v é o hash de prova de menor valor conhecido pelo nó i na rodada r. Caso o nó não receba ou não produza algum bloco em uma rodada, qualquer hash de prova de qualquer bloco produzido será aceito.

$$\begin{aligned} Z_r &= Q_r \cdot (v-1)/2^{256}, \quad && se\ s_r > 0 \\ Z_r &= Q_r, && se\ s_r = 0 \end{aligned} \tag{6}$$

Desta forma, Z_r é a razão entre o número de hashes de prova menores que v e o número total de hashes que podem ser produzidos a partir da saída de uma função de 256 bits ponderada pela probabilidade Q_r.

Seja $P_v^{(x)}$ a probabilidade de que aconteça uma reversão de um bloco com hash de prova v criado na rodada r em uma rodada x qualquer ($x > r$). Esta probabilidade depende do número de sucessos que o nó conhece desde sua rodada de criação (r) até a rodada x. Assim, quanto mais informações a respeito de blocos produzidos depois da rodada r, menor será a probabilidade de reversão do bloco produzido nesta rodada. A probabilidade de reversão é sempre calculada no início de uma nova rodada, ou seja, ela representa a probabilidade de reversão considerando os sorteios bem sucedidos até o final da rodada anterior. Na Fig. 2 é analisada a probabilidade de reversão do bloco ${}_vB_r$ ($P_v^{(x)}$) entre as rodadas $r+1$ e $r+4$. Na visão (a) da Fig. 2, é calculada primeiramente a probabilidade $P_v^{(r+1)}$ no início da rodada $r+1$. Como é conhecida apenas a rodada de criação do bloco de interesse, essa probabilidade é $P_v^{(r+1)} = Z_r$. Na visão (b) o nó i recebe mais um bloco (${}_aB_{r+1}$) e descobre mais um sorteio bem sucedido. Para reverter o bloco ${}_vB_r$ neste momento, é necessário que sejam desconhecidos pelo menos dois blocos em duas rodadas, já que o bloco ${}_aB_{r+1}$ é seu sucessor e é preciso que surja um fork com

comprimento pelo menos igual ao atual (2 blocos) para viabilizar a reversão. Deste modo, a reversão de ${}_vB_r$ torna-se menos provável quando calculada no início da rodada $r+2$, ou seja, $P_v^{(r+2)} = Z_r Z_{r+1}$.

Na visão (c) da Fig. 2, nenhum novo bloco é recebido na rodada $r+2$, o que faz com que a probabilidade de reversão do bloco em análise, calculada no início da rodada $r+3$, seja a mesma que foi calculada no início da rodada $r+2$. Isso mostra que o número de blocos que precisam ser revertidos na visão do nó i permanece o mesmo ao final da rodada $r+2$. Quando nenhum bloco é recebido ou produzido dentro de uma rodada, temos uma lacuna. As lacunas aumentam a probabilidade de reversão de um bloco, já que aumentam a possibilidade de reversão de blocos com rodadas superiores a elas. Neste sentido, o bloco recebido durante a rodada $r+3$ (${}_cB_{r+3}$) pode ser revertido por um bloco produzido na rodada $r+2$ e também por um bloco com rodada $r+3$ e hash de prova de menor valor, já que em ambos os casos os critérios de aceitação estabelecidos na seção 3.4 são atendidos. A probabilidade de reversão do bloco ${}_vB_r$ no início de $r+4$ é mostrada por meio da árvore de reversão apresentada na Fig. 3. De acordo com a Fig. 3 nota-se que até o surgimento da lacuna na rodada $r+2$, existe apenas uma possibilidade possível para reversão do bloco ${}_vB_r$ ($P_v^{(r+2)} = Z_r Z_{r+1}$). Com o surgimento da lacuna, o bloco ${}_cB_{r+3}$ pode ser revertido por qualquer bloco produzido na rodada $r+2$, o que ocorre com probabilidade Z_{r+2}, fato que é representado pelo ramo inferior da árvore. Caso não exista bloco desconhecido na rodada $r+2$, o que ocorre com probabilidade $1 - Z_{r+2}$, o bloco ${}_cB_{r+3}$ pode ser revertido na rodada $r+3$ com probabilidade Z_{r+3}. Realizando a multiplicação das probabilidades de cada nó de um caminho e somando todos os caminhos possíveis, temos a probabilidade de reversão do bloco ${}_vB_r$ no início da rodada $r+4$ dada por $P_v^{(r+4)} = Z_r Z_{r+1}(Z_{r+2} + Z_{r+3} - Z_{r+2}Z_{r+3})$.

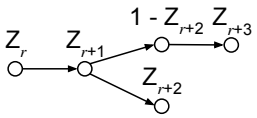

Figura 3. Árvore de reversão do bloco ${}_vB_r$ no início da rodada $r+4$

O bloco ${}_vB_r$ é considerado confirmado de maneira probabilística em uma rodada x qualquer, quando sua probabilidade de reversão é menor que um limiar ϵ estabelecido ($P_v^{(x)} < \epsilon$). Desta forma, o valor de ϵ determina um compromisso entre a segurança e o desempenho. Quando são utilizados valores muito pequenos para o mesmo, o bloco torna-se mais seguro contra reversões, já que esta possibilidade torna-se muito pequena. Entretanto, são necessárias mais rodadas para que a probabilidade de reversão atinja o limiar estabelecido por ϵ, o que aumenta o tempo necessário para a confirmação de blocos e reduz o desempenho do mecanismo. Assim, a partir da confirmação probabilística é possível definir qual será o limiar de confirmação utilizado pela rede. Isso torna o mecanismo proposto mais flexível na confirmação de blocos quando comparado com os mecanismos determinísticos, já que nesta segunda categoria o consenso de fato deve ser obtido por meio dos votos emitidos pelos membros de um comitê de validação. Portanto, para obter a probabilidade de reversão do bloco ${}_vB_r$ em uma rodada x ($x > r$), o mecanismo proposto produz a árvore de probabilidade de reversão, realizando, na sequência, a multi-

plicação de todas as probabilidades encontradas em todos os caminhos produzidos entre a raiz e as folhas da árvore de reversão e somando o resultado obtido de todos os caminhos.

4. Resultados práticos

As primeiras análises relacionadas ao novo mecanismo de confirmação probabilística foram realizadas por meio da implementação do protocolo e posterior execução em um ambiente virtualizado baseado em Mininet. O Mininet foi executando em um computador com 16 GB de memória RAM, processador Intel i7 de 1.80 GHz e sistema operacional Linux Ubuntu 18.04. No cenário apresentado nesta seção as execuções foram feitas para uma rede *peer-to-peer* com 10 nós e 10000 moedas ($W = 10000$) igualmente distribuídas entre os nós para diferentes valores de probabilidade de sucesso (p) e limiares para confirmação probabilística (ϵ). Essas execuções são preliminares e apresentam resultados ainda limitados, haja vista que na prática não serão encontradas redes com distribuição totalmente equalitária do *stake*. Essa simplificação foi feita como preparação para testes mais genéricos.

O gráfico da Fig. 4 mostra o desempenho do mecanismo em número de transações confirmadas por rodada para cada valor de ϵ e p avaliados nas execuções. É considerado que um bloco tem em média 2000 transações, como ocorre com o número médio de transações contidas em um bloco do Bitcoin [blockchain.com 2021]. O primeiro resultado

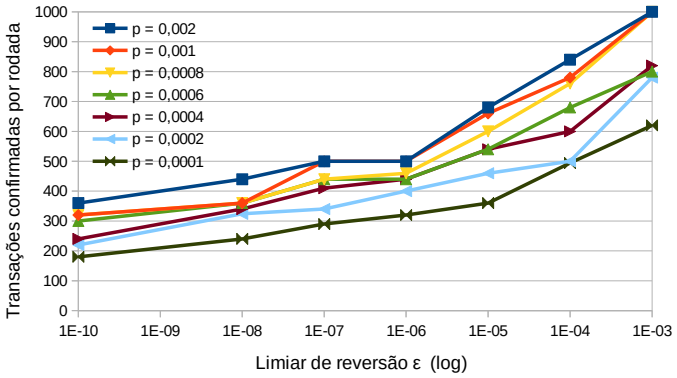

Figura 4. O limiar de probabilidade de reversão tem impacto maior no desempenho do mecanismo que a variação da probabilidade de um nó ser sorteado

observado é que, com o aumento do parâmetro ϵ, o mecanismo passa a confirmar transações mais rapidamente, já que o limiar torna-se mais fácil de ser atingido. Assim, a partir do aumento do parâmetro ϵ, o mecanismo é capaz de confirmar seus blocos mais rapidamente o que, consequentemente, aumenta o número de transações confirmadas. Neste sentido, como exemplo, a partir do gráfico da Fig. 4, é possível notar que o mecanismo pode oferecer um nível de segurança contra reversão de 10^{-6} obtendo 500 transações confirmados por rodada para uma probabilidade de sucesso $p = 0,001$. Para efeito de comparação, destacamos que este nível de segurança contra reversões de 10^{-6} é similar ao oferecido pela blockchain do Bitcoin após seis confirmações de uma transação (o que leva pelo menos uma hora) [Tschorsch and Scheuermann 2016]. Nestas execuções foram utilizadas rodadas com duração $T = 5$ segundos, o que garatiu 100 transações confirmadas por segundo. Nota-se também que a taxa de transações confirmadas por rodada

melhora quando a probabilidade de sucesso p é aumentada para um valor de ϵ constante. Isso acontece porque em geral existem mais blocos por rodada produzidos na rede a partir do aumento de p.

O comportamento do número de blocos produzidos por rodada em função do aumento do número esperado de sorteios bem sucedidos (τ) é mostrado no gráfico da Fig. 5, construído a partir de execuções em uma rede com 10 e 20 nós. De acordo com o gráfico,

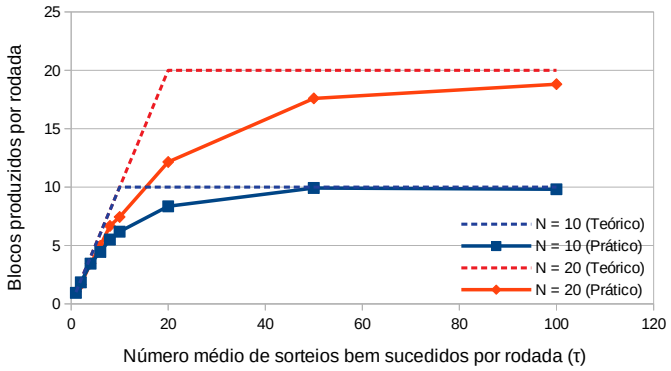

Figura 5. Blocos produzidos por rodada são limitados pelo valor de τ

quando τ é aumentado o número de blocos produzidos por rodada também aumenta. Mas este aumento é 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 mecanismo. Por outro lado, se τ é fixado e o tamanho da rede aumenta, algo que é esperado na prática, o número total de blocos produzidos por rodada também será limitado, só que agora pelo valor de τ. Isso significa que a rede pode crescer indefinidamente, mas o tráfego causado pela produção de novos blocos não irá acompanhar este crescimento, ficando limitado pelo parâmetro τ e tornando o mecanismo de consenso escalável.

5. Conclusões e Trabalhos Futuros

O mecanismo de consenso proposto neste artigo é uma alternativa aos mecanismos baseados em Proof-of-Stake que utilizam confirmação determinística a partir de comitês de validação. Ao realizar a confirmação de blocos de forma probabilística, os nós da rede *peer-to-peer* proposta dependem apenas da quantidade de sorteios bem sucedidos que é informada pelos blocos recebidos na rodada e, assim, decidem qual bloco deve ser confirmado. Além disso, o mecanismo proposto é mais flexível na confirmação de blocos quando comparado com mecanismos determinísticos, já que a confirmação é baseada em um limiar ajustável. Como trabalhos futuros, é necessário considerar cenários mais genéricos para o mecanismo, onde seja considerada a possibilidade dos nós não conseguirem seguir o melhor bloco produzido segundo os critérios estabelecidos na seção 3.4. Além disso, deve ser realizada uma análise de convergência e de correção do mecanismo além de comparações de desempenho e segurança com consensos determinísticos.

Referências

Bashir, I. (2017). *Mastering Blockchain*. Packt Publishing Ltd., 1 edition.

blockchain.com (2021). Average transactions per block. [Online]. `https://www.blockchain.com/charts/n-transactions-per-block`.

Christidis, K. and Devetsikiotis, M. (2016). Blockchains and smart contracts for the internet of things. *IEEE Access- Internet of Things Journal. https://ieeexplore-ieee-org.ez338.periodicos.capes.gov.br/document/7467408/*, 4:2292–2303.

Fischer, M. J., Lynch, N. A., and Paterson, M. D. (1985). Impossibility of distributed consensus with one faulty process. *Journal of ACM*, 32(2):374–382.

Gilad, Y., Rotem Hemo, S. M., Vlachos, G., and Zeldovich, N. (2017). Algorand: Scaling byzantine agreements for cryptocurrencies. *SOSP '17 Proceedings of the 26th Symposium on Operating Systems Principles*, pages 51–68.

Greve, F., Sampaio, L., Abijaude, J., Coutinho, A., Ítalo Valcy, and Queiroz, S. (2018). Blockchain e a revolução do consenso sob demanda. *XXXVI Simpósio Brasileiro de Redes de Computadores e Sistemas Distribuídos*, Minicurso(1).

Heilman, E., Kendler, A., Zohar, A., and Goldberg, S. (2015). Eclipse Attack on Bitcoin's Peer-to-Peer Network. In *Proceedings of the 24th USENIX Conference on Security Symposium*, pages 129–144, Washington,DC, United States. USENIX Association.

Kiayias, A., Russell, A., David, B., and Oliynykov, R. (2017). Ouroboros: A provably secure proof-of-stake blockchain protocol. *Advances in Cryptology – CRYPTO 2017*.

King, S. (2013). Primecoin: Cryptocurrency with prime number proof-of-work. [Online]. `http://primecoin.io/bin/primecoin-paper.pdf`. Whitepaper.

Kwon, J. (2014). Tendermint: Consensus without mining. *whitepaper. https://tendermint.com/static/docs/tendermint.pdf*.

Merkle, R. C. (1990). A certified digital signature. In *Advances in Cryptology — CRYPTO' 89 Proceedings*, pages 218–238, New York, NY, United States. Springer New York.

Nakamoto, S. (2009). Bitcoin: A peer-to-peer electronic cash system. [Online]. `https://bitcoin.org/bitcoin.pdf`. Whitepaper.

Tschorsch, F. and Scheuermann, B. (2016). Bitcoin and beyond: A technical survey on decentralized digital currencies. *IEEE Communications Surveys Tutorials*, 18(3):2084–2123.