Proof-of-Stake baseado em Tempo Discreto

Yoshitomi Eduardo Maehara Aliaga, 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

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

Abstract. *In this work, we present a new consensus mechanism for blockchains based on Prood-of-Stake, which makes easier the participation of nodes in the mining process. The new mechanism uses rounds based on a discrete time counter, where only participants who have passed the round challenge can generate a block. Therefore, it can guarantee a better chance of participation to each node because it does not require expensive hardware setup to mine new blocks.*

Resumo. *Neste trabalho apresentamos um novo mecanismo de consenso para blockchains baseadas em Proof-of-Stake, o qual facilita a participação de nós no processo de mineração. O novo mecanismo usa rodadas baseadas em um contador de tempo discreto, onde apenas participantes que tenham passado no desafio da rodada podem gerar um bloco. Portanto, ele pode garantir uma melhor chance de participação para cada nó, já que não exige caras instalações de hardware para a mineração de novos blocos.*

1. Introdução

As *blockchains* públicas têm atraído muita atenção principalmente depois da popularização das criptomoedas. A *blockchain* pode ser vista como um livro razão, onde as transações inseridas são imutáveis, após decorrido um tempo de confirmação. Aplicações que usam *blockchains* tornam-se cada dia mais populares e várias propostas inovadoras têm surgido para cartórios digitais, gestão de identidades, armazenamento confiável de informações críticas, criptomoedas, entre outras.

Uma das partes mais importantes de uma *blockchain* é o mecanismo de consenso, que permite que partes desconhecidas entre si e até mesmo concorrentes cheguem a um acordo sobre o estado atual e passado dos dados armazenados. Portanto, o consenso em *blockchain* viabiliza a criação de novos blocos de forma distribuída sem exigir validação por uma terceira parte confiável mesmo que existam participantes não confiáveis. Bashir [Bashir 2017] define o mecanismo de consenso como um conjunto de passos que são dados por todos ou pela maioria dos nós para entrar em consenso a respeito de um estado proposto ou valor. Em outras palavras este algoritmo permite que dois ou mais processos concordem sobre um determinado valor ou estado de uma rede. Um mecanismo de consenso pode ser visto como uma solução ao Problema dos Generais Bizantinos [Lamport et al. 1982] que consiste em resolver o dilema de atingir um consenso entre os participantes (que neste caso são os generais) com um objetivo comum. Embora existam generais traidores, os quais possuem objetivos opostos ao consenso e que tentarão

atrapalhar o processo, a ideia é que os generais leais (se forem maioria) atinjam o objetivo comum, sem que os traidores consigam impedí-los.

Dentre as propostas para mecanismos de consenso em *blockchains*, a mais conhecida é *Proof-of-Work* (PoW) [Nakamoto 2009]. Trata-se do mecanismo utilizado na mais popular das criptomoedas: *Bitcoin*. Porém, apesar de resolver o problema de consenso para uma criptomoeda de sucesso, ele traz problemas associados ao uso intensivo de poder computacional e ao alto consumo de energia. Este problema está diretamente ligado à segurança da *blockchain*, pois incentiva a formação de grupos de poderosos participantes (com acesso a instalações de alto poder computacional), que acabam dominando o processo de geração de novos blocos. Torna-se mais fácil então que se forme um grupo que supere os 50% de todo o poder computacional de mineração e comprometa a integridade da *blockchain*. Estes nós poderiam trabalhar em cadeias paralelas de blocos, chamadas de *forks*, revertendo transações já realizadas e colocando em risco a confiança no sistema. Esta exigência de alto poder computacional do PoW acaba dificultando a descentralização da rede e desmotivando a entrada de novos participantes.

Além disso o PoW é extremamente ineficiente quanto ao consumo de energia. Todos os participantes da rede utilizam alto poder computacional de suas máquinas mineradoras para vencer o desafio de geração do próximo bloco no menor tempo possível. Entretanto, a maior parte de toda essa energia é desperdiçada, pois apenas um participante de fato irá vencer a corrida e ganhar o direito de gerar um dado bloco n. Todos os demais participantes têm que abandonar suas tentativas de minerar o mesmo bloco e recomeçar do zero para tentar vencer a corrida pela mineração do bloco $n + 1$.

Por este motivo, foram propostos outros mecanismos de consenso tais como *Proof-of-Stake* [King 2013], *Proof-of-Activity* [Bentov et al. 2014], *Delegated-Proof-of-Stake* [Bitshare 2016], entre outros, como uma forma de contornar as desvantagens presentes no *Proof-of-Work*. Uma discussão mais detalhada e uma comparação entre estes mecanismos pode ser encontrada no trabalho de Maehara et al. [Maehara et al. 2018]

Neste texto apresentamos uma proposta de mecanismo de consenso baseado na ideia de sorteio presente no PoS. Trata-se de uma abordagem simplificada de um algoritmo *Proof-of-Stake*, onde novos blocos só podem ser gerados dentro de uma rodada esperada. Por meio do uso de um esquema síncrono, outros participantes podem calcular a rodada esperada de um bloco recebido da rede e compará-la com a rodada utilizada pelo participante criador do bloco. Assim é possível decidir se o bloco deve ser aceito, caso sua rodada seja a esperada, ou ignorado, caso contrário.

As seções seguintes irão destacar as principais características do mecanismo PoS, apresentar a proposta de um PoS baseado em tempo discreto, detalhar seus principais parâmetros e mostrar alguns resultados teóricos e práticos resultantes de uma implementação de prova de conceito.

2. Proof of Stake (PoS)

O *Proof-of-Stake* é um mecanismo de consenso em que o sistema faz uma escolha do nó que poderá criar um novo bloco por meio de um sorteio, cuja probabilidade de sucesso (de ser sorteado) pode ser influenciada pela sua quantidade de moedas.

Assim como no *Bitcoin*, o participante deve calcular o *hash de prova*, o qual é

resultante do cálculo de uma função $hash$ de boa qualidade sobre elementos do cabeçalho do bloco a ser criado. Uma vez calculado o *hash de prova*, este é comparado com um objetivo definido pelo sistema de modo a refletir um certo grau de dificuldade. Caso seja menor do que este objetivo, ocorre o sucesso do sorteio e o participante ganha o direito de criar o novo bloco e auferir os ganhos decorrentes de tal criação. A Eq. 1 define a condição que deve ser testada.

$$hash\_de\_prova < objetivo \tag{1}$$

Quanto maior o objetivo, menor é a dificuldade de satisfazer esta inequação.

Vasin [Vasin 2013] destaca que, para a geração de um bloco, é necessária uma prova de que o nó possua uma certa quantidade de moedas. Antes de se candidatar para gerar um bloco, um nó deve enviar moedas a si mesmo, para provar a propriedade das mesmas. A quantidade de moedas de um nó pode ser multiplicada pelo objetivo, aumentando assim a probabilidade de que tal nó venha a ter sucesso no sorteio. Daí advém o nome Proof-of-Stake.

Existem na literatura duas formas básicas de combinação do $stake$ com o objetivo: na criptomoeda *Peercoin* [King 2013] é usada a idade da moeda de um participante como modificador do *stake*, ou seja, o participante que possuir moedas por mais tempo possui maior probabilidade de gerar o próximo bloco.

Já no consenso utilizado em *Blackcoin* [Vasin 2013], apenas a quantidade de moedas é usada na modificação do objetivo. Este processo aumenta a possibilidade de criar grupos mais privilegiados na escolha para gerar novos blocos, o que pode ser um problema para atingir uma maior descentralização na rede, caso não sejam tomados cuidados para evitar uma grande influência do $stake$ de cada nó no processo.

Uma outra forma de se alterar o valor do objetivo é multiplicando-o pelo tempo decorrido desde a criação do último bloco. Desta forma, caso o nível de dificuldade usado no objetivo esteja alto e nenhum nó consiga sucesso no sorteio, em uma etapa seguinte o valor do tempo a ser multiplicado pelo objetivo irá aumentar, aumentando assim as chances de que algum nó seja sorteado.

Algumas características do mecanismo PoS:

- normalmente não possui recompensa por gerar um bloco, mas os mineradores recebem as taxas das transações contidas no mesmo [GreenfieldIV 2017];
- há um desafio grande em rastrear de forma confiável as mudanças de saldo de cada participante [Bano et al. 2017];
- este mecanismo diminuiu o gasto de energia em comparação com PoW, já que não incentiva e não depende de poder computacional elevado;
- 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 recolher as taxas de vários blocos gerados em ramos distintos;
- é mais caro realizar uma fraude em PoS que em PoW, já que para tentar possuir 51% das chances de ser sorteado (isto é, garantir o sorteio), é preciso possuir e usar muitas moedas; para isso o atacante que não as tenha poderá gerar uma demanda pelas mesmas, causando imediatamente sua valorização e dificultando a obtenção das que faltam;

- o PoS permite aumentar uma eventual concentração de riquezas, já que quem possui mais moedas acaba tendo uma chance maior de ser escolhido para criar novos blocos e ganhar as taxas por isso, ficando ainda mais rico.

As principais moedas que utilizam este mecanismo de consenso são NxT [Community 2014], BlackCoin [Vasin 2013] , PeerCoin [King 2013], Ethereum (Slasher [Buterin 2014] & Casper [Buterin and Griffith 2017]) e Cardano (Ouroboros)[Kiayias et al. 2017] .

3. Proposta de um novo mecanismo de consenso baseado em PoS

Antes de detalharmos a proposta, iremos definir alguns conceitos e parâmetros.

3.1. Definição de Parâmetros

Os mecanismos baseados em PoS, quando comparados àqueles baseados em PoW, trazem ganhos em relação ao consumo de energia e facilitam a participação de qualquer nó interessado em criar novos blocos. Entretanto, eles também trazem alguns desafios, tais como: como definir se um nó foi sorteado em um determinado instante? Como detetar e limitar o crescimento de cadeias secundárias geradas a partir de *forks*? Como aumentar a descentralização na geração de novos blocos? A seguir serão discutidos o sorteio do nó em uma rodada e o controle de *forks*, explicando como são tratados no novo mecanismo proposto.

3.1.1. Sorteio do nó em uma rodada

Como em todo mecanismo de sorteio, é necessário verificar se um determinado nó participante realmente foi sorteado. Em um ambiente distribuído é essencial que todos os outros participantes possam verificar quem de fato foi sorteado em uma determinada rodada. Nesta proposta, uma rodada é um intervalo de tempo representado por um número inteiro (número de intervalos de tempo transcorridos desde um dado instante). Cada nó participante deve tentar gerar o próximo bloco da cadeia blockchain sempre dentro de uma rodada. Se não tem sucesso no sorteio em uma dada rodada, precisa aguardar pela próxima para fazer uma nova tentativa.

A rodada para o próximo bloco da cadeia é encontrada por meio da quantidade de tempo, desde a chegada do último bloco aceito, dividida por um intervalo fixo chamado de $TIMEOUT$ (Eq. 2). O valor do $TIMEOUT$ é global e todos os participantes podem calcular a rodada esperada para um novo bloco a partir da rodada registrada em um bloco já criado. Observamos que o único valor variável na equação da rodada é o tempo atual. Ou seja, o valor da rodada só irá mudar após transcorrido um intervalo de tempo igual a $TIMEOUT$ desde o último cálculo da rodada pelo nó em questão.

$$\begin{aligned} rodada &= \textit{último\_bloco.rodada} \\ &+ \left\lfloor \frac{\textit{tempo\_atual}() - \textit{último\_bloco.tempo\_chegada}}{TIMEOUT} \right\rfloor \end{aligned} \tag{2}$$

onde:

- ***tempo_atual()*** representa a data e hora atual no padrão UTC,

- ***último_bloco.tempo_chegada*** representa o tempo de chegada do bloco anterior,
- ***último_bloco.rodada*** representa a rodada do bloco anterior,
- $TIMEOUT$ representa a duração de cada rodada.

Como já foi mostrado pela Eq. 1, cada participante deve calcular um *hash de prova* e o resultado do mesmo deve ser inferior a um dado objetivo. Supõe-se o uso de uma função *hash* de boa qualidade, na qual as saídas não demonstram correlação entre si, mesmo se as entradas forem correlacionadas. Nesse caso, tal função *hash* pode ser usada como gerador de números pseudoaleatórios ou como um mecanismo de sorteio. Se este *hash* de prova usa como entrada a identidade (fixa) do nó participante e outros dados que não podem ser alterados por ele, então conseguir vencer o objetivo é equivalente a vencer um sorteio.

O *hash de prova* usado nesse novo mecanismo está baseado na função de *hash* criptográfico SHA-256 e é calculado pela Eq. 3.

$$hash\_de\_prova = H(último\_bloco.hash \| rodada \| identificador\_do\_nó) \quad (3)$$

onde:

- $H()$: é a função *hash* SHA-256,
- ***último_bloco.hash***: é o *hash* do último bloco da cadeia,
- ***rodada***: é o valor da rodada atual,
- ***identificador_do_nó***: é a identidade do nó baseada em sua assinatura.

O cálculo do *hash* de prova funciona então como uma espécie de sorteio baseado nas três informações de entrada. Dependendo do valor definido para o objetivo (Equação 1), é provável que o nó não consiga um *hash* de prova pequeno o suficiente e tenha que fazer nova tentativa, o que só poderá ocorrer em uma próxima rodada, já que os demais elementos de entrada da função são fixos. Isto impede que nós com grande poder computacional possam fazer mais tentativas de sorteio que os nós mais simples, uma vez que todos os nós só poderão calcular um *hash* de prova a cada intervalo de tempo.

O valor do parâmetro $TIMEOUT$ deverá ser escolhido cuidadosamente, pois não poderá ser tão pequeno que privilegie os nós mais rápidos (e facilite a criação de *forks*), nem tão grande que prejudique o desempenho da rede.

3.1.2. Controle de forks

Diferentemente dos mecanismos baseados em prova de trabalho, onde minerar blocos em duas ou mais cadeias gera um alto custo, nos mecanismos PoS este custo é desprezível, pois limita-se ao cálculo de um *hash*. Por isso, mecanismos de consenso PoS permitem que um participante crie blocos em diferentes cadeias e não apenas na cadeia mais longa, o que resulta na geração de *forks* indesejáveis.

Considerando que o *stake* de um determinado participante seja igualmente distribuído entre as cadeias, ele possui, nesse caso, a mesma probabilidade de sucesso no sorteio para todas as cadeias consideradas. Como consequência disso, ele pode contribuir,

sem custos adicionais, para que uma cadeia secundária, em um determinado momento, fique mais longa que a cadeia principal, causando efeitos indesejáveis para a rede. Na literatura esse problema é conhecido como *Nothing at Stake*, ou seja, um participante não tem nada a perder em tentar minerar ao mesmo tempo em várias cadeias diferentes.

Segundo [Buterin and Griffith 2017] é necessário criar mecanismos para punir os participantes que desrespeitam a geração de blocos para a cadeia mais longa. Nesta linha, ele propõe um algoritmo chamado *Casper*, que será o substituto do atual algoritmo baseado em PoW no sistema distribuído *Ethereum*. No *Casper*, a cada 100 blocos são fixados *checkpoints*, nos quais os nós participantes têm que votar para definir qual é a cadeia principal entre cada par de *checkpoints* consecutivos, criando os *enlaces majoritários* entre *checkpoints*. Desde que dois terços dos participantes sejam honestos, é possível saber com precisão qual é a cadeia principal a seguir, pois neste caso o protocolo garante a finalização apenas de *checkpoints* não conflitantes. Além disso, *Casper* penaliza com perda de moedas os nós participantes que não sigam a regra.

Na nova proposta, a penalidade é mais branda e estará na menor probabilidade de sorteio para um participante que deseja criar um bloco em uma cadeia secundária. É permitido ao participante criar blocos em qualquer *fork* da rede, porém a probabilidade de geração de blocos no ramo principal é consideravelmente maior quando comparada com outras cadeias. Assim, é pouco provável que uma cadeia que perdeu a disputa em um determinado *fork* em um tempo passado assuma a condição de cadeia principal. A proposta para diferenciar a probabilidade de mineração por cadeia é definir o objetivo como na Eq. 4.

$$\begin{aligned} objetivo &= stake \times 2^{tamanho\_do\_hash - dificuldade - penalidade} \\ &= stake \times 2^{256-D-P} \end{aligned} \tag{4}$$

Este objetivo também deve ser compatível com o *hash* de prova, isto é, deve ser representável por, no máximo, o número de bits da função de *hash* utilizada. O parâmetro $dificuldade$ ($D < 256$) controla a dificuldade de sucesso no sorteio, com impactos na velocidade de convergência do algoritmo.

Já o parâmetro $penalidade$ ($P < 256 - D$) define o nível de aumento da dificuldade para se minerar nas cadeias não principais (menores). Caso seja criado um bloco na cadeia principal, fazemos $P = 0$. Com valores elevados para P ($P \simeq 256 - D$), esta proposta de controle de *forks* equivale na prática a proibir a geração de blocos em cadeias secundárias.

Este controle de *forks* pode dar uma maior agilidade ao mecanismo, uma vez que permite que os nós tentem minerar blocos em cadeias secundárias, ainda que com uma probabilidade menor de sucesso. A motivação para esta estratégia reside no fato de que é possível que nós em pontos distantes da rede tenham visões diferentes sobre quais cadeias são secundárias e qual é a principal, devido a atrasos de propagação de blocos. Assim, mesmo que um nó considere uma dada cadeia como secundária, ele tem permissão para tentar criar um novo bloco para a mesma (com menor probabilidade de sucesso) na expectativa de ser bem sucedido futuramente, caso esta cadeia secundária seja, na verdade, a principal, de acordo com a visão dos demais nós da rede. Se esta estratégia não for bem controlada, no entanto, ela pode contribuir para um aumento dos *forks* e uma maior dificuldade da rede em descobrir qual é a cadeia principal.

3.2. Proof-of-Stake baseado em Tempo Discreto (PoSBTD)

Esta seção detalha a proposta do novo mecanismo, chamada de PoS baseado em Tempo Discreto (PoSBTD) , pois a criação de blocos só ocorre em intervalos de tempo fixos (rodadas).

Apesar de estarmos tratando de um mecanismo baseado em PoS, no qual cada nó coloca seu $stake$ para influenciar sua chance de sucesso no sorteio, limitaremos nossas análises iniciais ao caso em que todos os nós têm um mesmo valor de *stake* $= 1$. Assim, eliminamos a diferenciação que nós mais ricos poderiam impor aos mais pobres e igualamos as chances de qualquer nó ter sucesso no sorteio (observe que a obrigatoriedade de se minerar somente um vez por rodada também iguala as chances entre nós poderosos e nós mais simples).

O mecanismo proposto pode ser dividido em três processos principais que trabalham de forma concorrente, como descrito a seguir.

3.2.1. Processo de mineração de novos blocos

O processo de mineração consiste em o nó participante verificar se obtém sucesso na comparação do *hash de prova* com o objetivo na rodada atual. Em caso de sucesso, ele calcula e divulga (por meio de *broadcast*) um novo bloco para cada uma das cadeias que conhece; caso contrário, ele aguarda a chegada de um novo bloco calculado por outro nó ou uma nova rodada para fazer uma nova tentativa (Fig. 1).

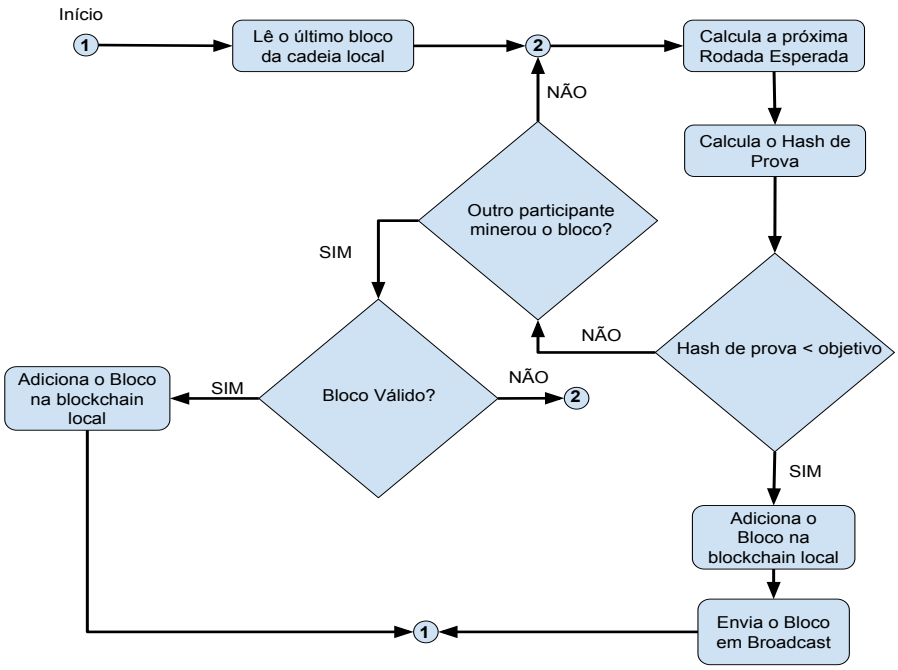

Figura 1. Processo de mineração de novos blocos

3.2.2. Processo de escuta de novos blocos

O processo de escuta é responsável por receber novos blocos criados por outros participantes e validá-los, buscando sua inserção na *blockchain* local (Fig. 2). Ele fica esperando uma mensagem com um novo bloco e, quando ela chega, é verificado se a cadeia local e este bloco estão sincronizados (isto é, se o bloco se encaixa no final da cadeia local ou se há alguma lacuna entre ele e a cadeia). Caso exista tal lacuna, o nó chama o processo de sincronização para completar sua cadeia local, o que pode também trazer novos *forks* para a mesma. Se não há lacuna, a consistência do bloco é verificada por meio do cabeçalho do bloco, da rodada esperada e da confirmação se o nó que criou e propagou o bloco de fato passou no teste do objetivo. Caso qualquer uma destas verificações falhe, o bloco é descartado e o processo volta ao início. Caso contrário, ele é aceito e a anexado à *blockchain* local.

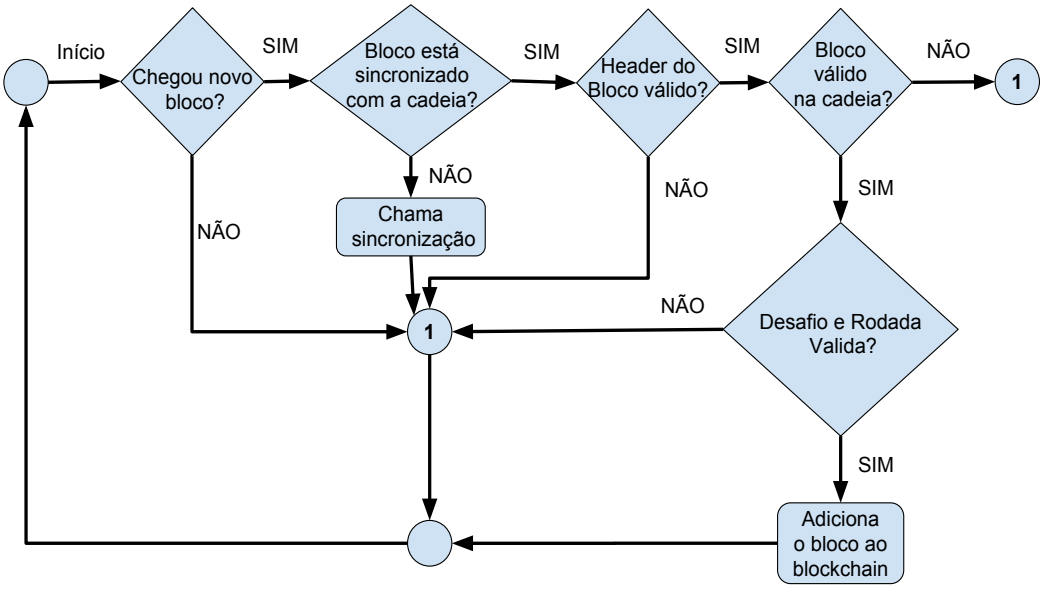

Figura 2. Processo de escuta de novos blocos

3.2.3. Processo de sincronização de blocos

O processo de sincronização consiste em buscar novos blocos de outros participantes para obter uma visão mais completa da *blockchain* (Fig. 3). Ao ser executado, o processo contata os *peers* do nó e solicita um a um os blocos que faltam, a partir do bloco com maior índice recebido, para preencher a lacuna que foi identificada no processo de escuta de novos blocos. Ao receber cada um dos blocos, o nó faz as verificações de consistência dos cabeçalhos e das rodadas antes de adicioná-los na *blockchain* local. Caso algum bloco não seja da cadeia principal, isso significa que existe um *fork* em algum ponto da cadeia. Esse *fork* é localizado e uma troca da cadeia principal pode ocorrer.

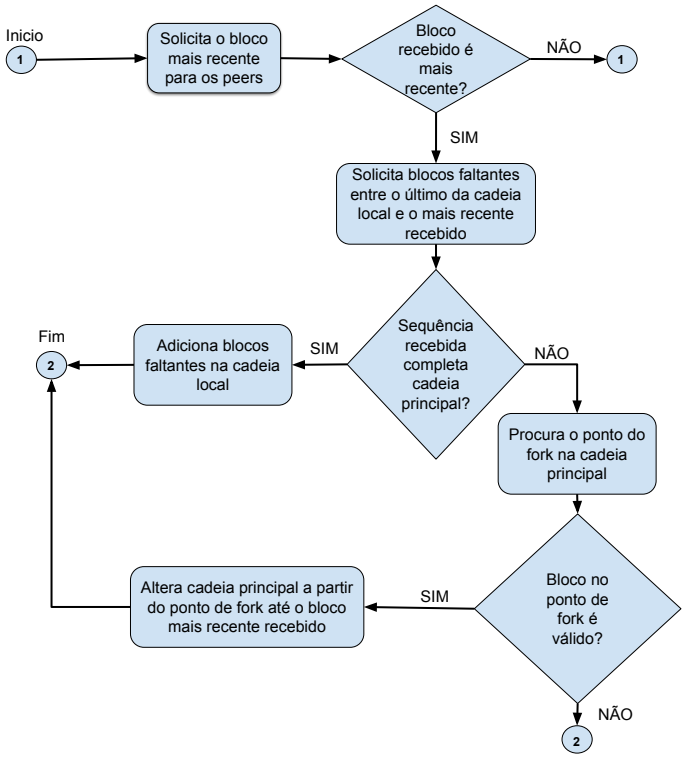

Figura 3. Processo de sincronização de novos blocos

4. Testes e Resultados

4.1. Resultados teóricos

Inicialmente faremos uma análise teórica do comportamento do mecanismo proposto.

Considerando que será utilizado um *stake* unitário para todos os nós, a Eq. 5 apresenta a probabilidade de ocorrência de um evento S_1 (sucesso no sorteio em um nó, isto é, sucesso em atingir o objetivo por meio do *hash de prova*).

$$p[S_1] = \frac{2^{256-D-P}}{2^{256}} \tag{5}$$

Usando o resultado da Eq. 5 podemos encontrar o número esperado de rodadas até ocorrer sucesso no sorteio em um nó (Eq. 6).

$$Nr_{S_1} = \left\lceil \frac{1}{p[S_1]} \right\rceil \tag{6}$$

Já o valor esperado para múltiplos nós depende da probabilidade de sucesso S_n, que é o sorteio bem sucedido em pelo menos um dos n nós participantes (Eq. 7).

$$p[S_n] = 1 - \prod_{i=1}^{n} (1 - p[S_1]) = 1 - (1 - p[S_1])^n \tag{7}$$

De forma similar pode-se encontrar o número esperado de rodadas para se ter algum sorteio (*hash*) bem sucedido quando se tem mais de um nó (Eq. 8).

$$Nr_{S_n} = \left\lceil \frac{1}{p[S_n]} \right\rceil \tag{8}$$

Com base nestas expressões foi avaliado, de forma teórica, o comportamento de uma *blockchain* controlada pelo novo mecanismo de consenso proposto.

A Fig. 4 mostra que a probabilidade de sucesso em uma dada rodada aumenta com o número de nós participantes na *blockchain*, mas diminui muito rapidamente quando cresce o nível de dificuldade. Foi adotada uma penalização P nula, já que é considerada a criação de bloco no ramo principal da *blockchain*.

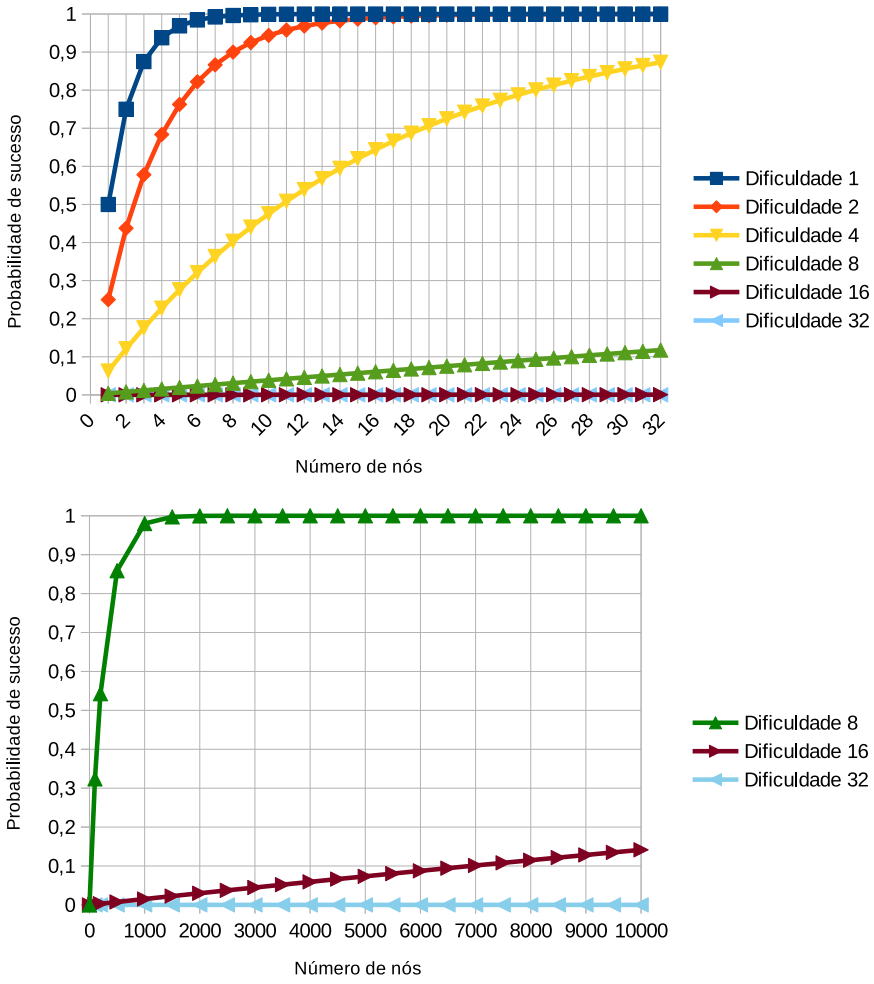

Figura 4. Probabilidade de sucesso no sorteio em função do número de nós

Na prática, os níveis de dificuldade devem ser ajustados de maneira a não permitir que seja muito provável qualquer nó ter sucesso no sorteio (aumentaria o número de *forks*), mas também não pode tornar esta probabilidade muito pequena, pois isso resultaria em atrasos muito grandes no processo de mineração. É possível notar que mesmo para uma rede similar em tamanho à rede do *Bitcoin* (cerca de 10.000 nós), os níveis de dificuldade 16 e 32 parecem ser muito fortes, não permitindo que a probabilidade de sucesso no sorteio de algum nó atinja os 15%. Pode parecer pouco, mas 15% de 10.000 são 1.500 nós tendo sucesso no sorteio, o que pode resultar em 1.500 *forks* a serem resolvidos. É por este motivo que se faz necessário um controle bem rigoroso do nível de dificuldade a fim de evitar que o número de *forks* aumente.

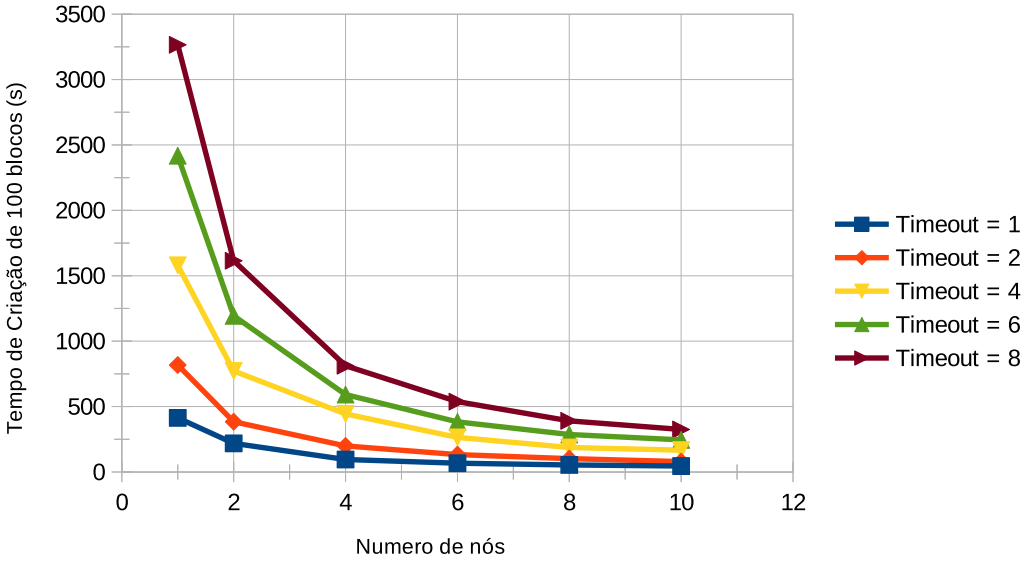

Figura 5. Tempo médio de criação de 100 blocos em função do número de nós para dificuldade 2 e P=0

4.2. Resultados práticos

Devido ao tempo necessário para cada teste e à dificuldade de gerenciar um número grande de nós que formam a *blockchain*, optamos por limitar o tamanho da rede em 10 nós e o $TIMEOUT$ em 8 segundos nesta avaliação inicial. Avaliações com outras configurações também foram efetuadas e os resultados foram coerentes com o esperado e com os aqui apresentados.

Os testes foram executados em um servidor de 16 *GB* de memória RAM e processador Intel *Xeon* de 2,4 *GHz* com um core. Os nós mineradores foram criados no simulador de redes *Mininet* em sua versão 2.3.0 usando a linguagem Python. Na simulação, todos os nós foram criados no mesmo servidor, não sendo utilizado link externo para comunicação com nós remotos. Nesses testes, foi adotado um nível de dificuldade $D = 2$ bits e penalidade $P = 0$.

A Fig. 5 mostra os tempos médios obtidos em cinco execuções do sistema para criar 100 blocos. A principal informação que se visualiza nas curvas para diferentes $TIMEOUTs$ é o tempo cada vez menor para criação de 100 blocos à medida que o tamanho da rede aumenta, como já era previsto teoricamente. A figura confirma também o crescimento esperado dos tempos de criação em função do aumento do $TIMEOUT$. Este crescimento do tempo é cada vez menor à medida que o sistema fica maior, dada a maior probabilidade de se obter sucesso no sorteio em poucas rodadas, o que diminui parcialmente o impacto do $TIMEOUT$. Resultados muito similares foram obtidos para $D = 4$ e $P = 0$, havendo apenas uma mudança de escala de tempo.

Na figura 6 temos uma comparação entre os tempos teóricos e práticos para a criação de 100 blocos para uma quantidade de nós variando de 1 até 10. É possível observar uma diferença maior entre os valores à medida que aumenta o número de nós. Essa diferença se deve principalmente ao fato de que o valor estimado pela Eq. 8 para

o número de rodadas está arrendondado para o primeiro inteiro acima. o que traz um erro maior quando menos rodadas são necessárias para se ter sucesso em um sorteio. Por outro lado, quanto mais nós estiverem participando dos sorteios, maior é a chance de algum deles conseguir obter sucesso antes do tempo estimado, já que se trata de uma estimativa baseada em probabilidades. Estes são os dois principais fatores responsáveis pela maior diferença entre os tempos para vários nós.

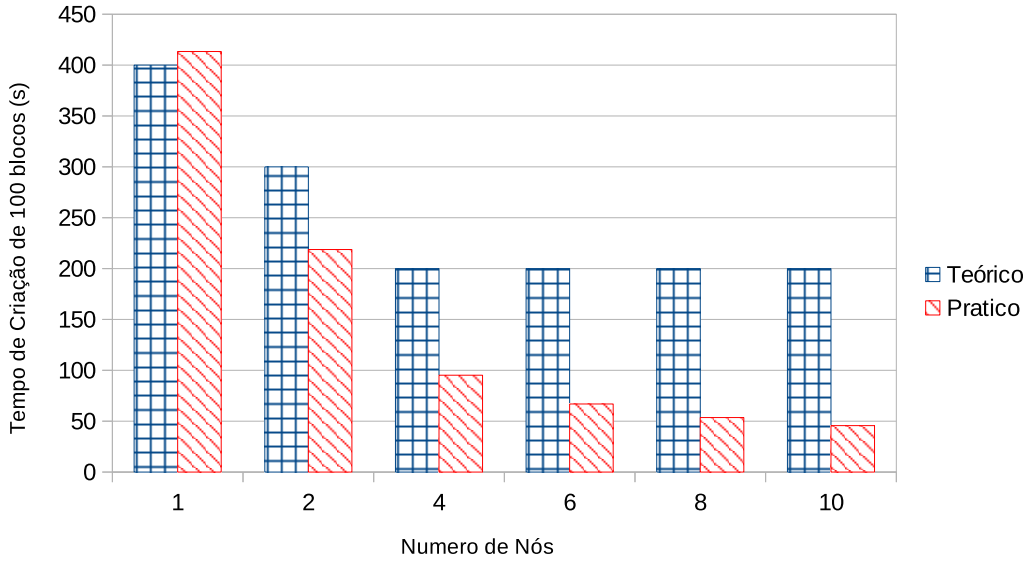

Figura 6. Comparação entre os tempos estimados e medidos para criação de 100 blocos com dificuldade 2 e $TIMEOUT = 1$

5. Segurança do PoSBTD

Ao contrário dos mecanismos baseados em PoW, nos quais há um custo muito alto para se produzir blocos fora da cadeia principal, naqueles baseados em PoS este custo é desprezível e um controle mais rigoroso sobre forks precisa ser exercido. Várias propostas de sistemas baseados em PoS fazem uso da ideia de se ter blocos especiais, chamados *checkpoints* que são pontos de controle sobre os quais os demais nós concordam que são definitivos e imutáveis. Entretanto, esta ideia de se usar *checkpoints* é bastante complexa e este é um dos motivos para não termos ainda um sistema de grande porte (de criptomoedas ou não) baseado nestas ideias do PoS.

No PoSBTD proposto, temos um controle de forks bem simples: os nós podem criar nós em cadeias secundárias, mas com uma probabilidade de sucesso bem menor do que a que têm na cadeia principal. Ainda são necessárias melhores avaliações desta proposta, mas acreditamos que seja uma forma simples e eficaz de evitar a proliferação de *forks*.

Outro problema de esquemas PoS é a Reversão de longo alcance, ou seja, um conjunto significativo de nós trabalhar em uma cadeia secundária e num dado momento propor a troca da cadeia principal pela secundária, revertendo muitas transações passadas. Para este caso também consideramos que o mecanismo proposto oferece uma boa robustez pela sua forma de construção, que exige a construção de blocos dentro de intervalos de tempo bem demarcados.

Assim como ocorre em outros mecanismos PoS ou até mesmo PoW, boa parte da segurança em PoSBTD advém do uso de funções $hash$ de boa qualidade, consideradas criptograficamente seguras para o intertravamento de blocos e para o sorteio dos nós autorizados a criar um novo bloco. Apesar de as transações que ocupam o corpo de um bloco não terem sido objeto das discussões neste trabalho, assume-se que elas estariam protegidas por assinaturas digitais tão seguras quanto possível e por uma Árvore de Merkle baseada em funções de $hash$ seguras, exatamente como é feito em outras blockchains, como a do *Bitcoin*, por exemplo.

6. Conclusões e Trabalhos Futuros

Os protocolos baseados em PoS são promissores para resolver problemas ligados ao consenso de *blockchain* públicos. Os grandes desafios estão ligados principalmente aos problemas de concorrência das cadeias geradas a partir de *forks* e também à dificuldade de rastreamento do *stake* possuído e utilizado por um determinado participante. Algumas propostas de sistemas baseados em PoS apareceram nos últimos anos na literatura, mas, devido principalmente à complexidade envolvida, nenhuma delas ainda está em produção em sistemas de maior porte.

O mecanismo proposto neste trabalho traz uma solução ao problema de grandes diferenças de oportunidades na definição do criador de um bloco. A obrigatoriedade de espera pela próxima rodada para se fazer uma nova tentativa de criação elimina as vantagens de poder computacional que um nó possa ter em relação a outros. Além disso, ele indiretamente resolve a questão do grande consumo de energia característico do PoW e propõe um controle simples de *forks*. Um protótipo do sistema com o mecanismo de consenso foi implementado e testado, apresentando resultados compatíveis com um modelo teórico previamente desenvolvido.

Um importante trabalho futuro é o desenvolvimento de um esquema automático de ajuste de nível de dificuldade para poder preservar o tempo de criação de blocos dentro de um intervalo previsível. Um outro ponto para desenvolvimento futuro seria um sistema de controle de *forks* com penalização adaptativa que dependesse basicamente do tamanho da diferença entre a cadeia principal e uma cadeia secundária específica.

A construção de um sistema para formação de grupos de validadores ou comitês para controlar a inserção de blocos na *blockchain* poderia trazer benefícios para a rede, como já proposto em vários outros trabalhos relacionados, já que liberaria os demais nós das tarefas de validação. Entretanto, alguns detalhes precisam ser melhor avaliados: a liberação dos nós comuns da tarefa de validação é realmente vantajosa? A escolha dos membros do grupo deve ser equiprovável e com certa rotatividade entre os mesmos? Os benefícios deste tipo de implementação compensariam os custos advindos do mesmo?

Acreditamos também que poderiam ser avaliadas possibilidades de utilização efetiva do $stake$ de cada nó em uma das suas possíveis variantes, mas de forma a não afetar de forma significativa as chances de sucesso no sorteio de um nó. É possível que o uso de $stake$ facilite o aumento da taxa de criação de blocos sem aumento dos *forks*, algo que precisa também de uma investigação mais aprofundada.

Agradecimento: à CAPES pelo financiamento parcial desta pesquisa (Processo núm. 88882.329365).

Referências

Bano, S., Sonnino, A., Al-Bassam, M., Azouvi, S., McCorry, P., Meiklejohn, S., and Danezis, G. (2017). Sok: Consensus in the age of blockchains. *ArXiv e-prints*.

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

Bentov, I., Lee, C., Mizrahi, A., and Rosenfeld, M. (2014). Proof of activity: Extending bitcoin's proof of work via proof of stake. *SIGMETRICS Perform. Eval. Rev.*, 42(3):34–37.

Bitshare (2016). Delegated proof-of-stake consensus a robust and flexible consensus protocol. `https://bitshares.org/technology/delegated-proof-of-stake-consensus/`. (acessado em 10/09/2017).

Buterin, V. (2014). Slasher: A punitive proof-of-stake algorithm. `https://blog.ethereum.org/2014/01/15/slasher-a-punitive-proof-of-stake-algorithm/`. (acessado em 27/09/2018).

Buterin, V. and Griffith, V. (2017). Casper the friendly finality gadget. *ArXiv e-prints*.

Community, N. (2014). Nxt whitepaper. [Online]. `https://bravenewcoin.com/assets/Whitepapers/NxtWhitepaper-v122-rev4.pdf`. Whitepaper.

GreenfieldIV, R. (2017). Explaining how proof of stake, proof of work, hashing and blockchain work together. `https://medium.com/@robertgreenfieldiv/explaining-proof-of-stake-f1eae6feb26f`. (acessado em 18/02/2018).

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.

Lamport, L., Shostak, R., and Pease, M. (1982). The byzantine generals problem. *ACM Transactions on Programming Languages and Systems (TOPLAS)*, 4(3):382–401.

Maehara, Y., Leal, V. C., de Lucena, A. U., and Henriques, M. A. A. (2018). Avaliação de mecanismos de consenso para blockchains em busca de nova estratégia mais eficiente e segura. In *XVIII Simpósio Brasileiro de Segurança da Informação e de Sistemas Computacionais.*, Natal - RN.

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

Vasin, P. (2013). Blackcoin's proof-of-stake protocol v2. [Online]. `https://blackcoin.co/blackcoin-pos-protocol-v2-whitepaper.pdf`. Whitepaper.