Avaliação da incidência de *forks* no algoritmo de consenso *Probabilistic Proof-of-Stake* (*PPoS*)

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 the Probabilistic Proof-of-Stake (PPoS) consensus protocol and analyses it theoretically and practically, focusing on its probability of producing forks. The text explains first the operation of the algorithm, where it is possible to understand how a node participates in a lottery every round, in order to gain the right to produce a new block in a chain. Next, it shows the rules for block acceptance and commitment, followed by an analysis of the fork probability and the expected number of rounds between two blocks. A blockchain based on this protocol was implemented and its practical results showed a strong agreement with the theoretical analysis, validating it.*

Resumo. *Este artigo apresenta o protocolo de consenso Probabilistic Proof-of-Stake (PPoS) e analisa o mesmo dos pontos de vista teórico e prático, focando na sua probabilidade de produzir forks. O texto primeiramente explica o funcionamento do algoritmo, onde é possível entender como um nó participa de um sorteio em rodadas a fim de ganhar o direito de criar um novo bloco para uma cadeia. Em seguida ele apresenta os critérios para aceitação e para confirmação de blocos, seguidos de uma análise da probabilidade de forks e do número esperado de rodadas entre dois blocos consecutivos. Uma blockchain baseada neste protocolo foi implementada e seus resultados práticos mostraram uma boa concordância com a análise teórica, validando a mesma.*

1. Introdução

Os algoritmos de consenso em blockchains permitem que uma determinada transação entre duas partes seja validada sem que exista uma terceira parte confiável [Christidis and Devetsikiotis 2016]. O mecanismo de consenso é definido por Bashir [Bashir 2017] como um conjunto de passos que são dados pelos nós que compõem a rede para entrar em consenso a respeito de um valor. As blockchains públicas necessitam de mecanismos de consenso eficientes já que os desafios para este tipo de estrutura são bastante elevados, principalmente os relacionados a presença de nós desonestos [Nguyen et al. 2019]. Este tipo de rede é totalmente assíncrono, ou seja, não existe um tempo estabelecido para que os nós enviem suas mensagens. Neste sentido, os mecanismos de consenso para blockchain públicas buscam resolver o problema dos Generais Byzantinos em redes assíncronas e com a presença de participantes maliciosos. Segundo

Lamport [Lamport et al. 1982], só é possível atingir o consenso distribuído se, no máximo, um terço dos nós presentes na rede forem desonestos.

Neste cenário, onde existem nós desonestos e a rede é assíncrona, destaca-se o trabalho de Fischer et al. [Fischer et al. 1985], que mostra não ser possível encontrar o consenso de forma determinística em uma rede totalmente assíncrona com a presença de nós maliciosos. Assim, os mecanismos de consenso para blockchains públicas podem definir um grau de sincronismo na rede para que seja possível criar mecanismos de consenso determinístico ou trabalhar com o conceito de consenso probabilístico [Greve et al. 2018]. Entre os mecanismos que exploram o sincronismo como forma de garantir o consenso determinístico, destaca-se o protocolo Algorand [Gilad et al. 2017]. Esse protocolo define um comitê de votação, onde o objetivo é validar todos os novos blocos que serão incluídos na blockchain.

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 todas as decisões dos nós honestos sejam em direção de um único valor [Greve et al. 2018]. Estes mecanismos são amplamente utilizados em blockchains públicas, onde é necessário concordar sobre qual é o próximo bloco da cadeia. Quando a opção é por este tipo de consenso, é possível que ocorram *forks*, se os nós não concordarem sobre quem é o sucessor de um determinado bloco. Nesse caso a blockchain permanece bifurcada até que os critérios estabelecidos pelo mecanismo de consenso sejam atendidos e os nós possam convergir para uma única cadeia [Deirmentzoglou et al. 2019].

O mecanismo de consenso probabilístico mais conhecido é o Proof-of-Work (PoW), onde os nós buscam criar o próximo bloco através de seus esforços computacionais para encontrar um valor de hash que atenda um desafio [Nakamoto 2009]. A principal 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 utilizado (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. Estes nós dispõem de recursos financeiros para investir em melhores hardwares, com desempenho muito maior quando comparados com processadores de propósito geral. Isso causa uma centralização em poucos nós que é indesejada em uma rede *peer-to-peer*, pois estes poucos nós poderiam controlar a rede, e executar um ataque conhecido como ataque dos 51%, onde os nós desonestos formam um grupo que detem a maioria dos nós e assim conseguem manipular alguns consensos de forma a tirar proveito de algum resultado de interesse do grupo [Nguyen et al. 2019].

Estes fatos despertaram o interesse por outros mecanismos de consenso probabilísticos e, entre as abordagens destaca-se o Proof-of-Stake (PoS), onde a prova de posse do *stake* é utilizada pelo nó para ganhar o direito de tentar criar novos blocos e receber recompensas pelo seu trabalho [King 2013]. A prova de posse do *stake* consiste em utilizar recursos do próprio nó, como por exemplo, sua 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. Existem algumas implementações que possuem variações principalmente relacionadas ao modelo de remuneração dos nós criadores de blocos. No

algoritmo Casper [Buterin and Griffith 2017] a remuneração está associada ao grupo de validadores que vota em blocos especiais chamados de *checkpoints*. Neste protocolo todos os nós que participam da votação são remunerados de maneira proporcional à quantidade de *stake* que eles depositaram no momento que entraram no grupo de validação.

O Algorand [Gilad et al. 2017] é um protocolo de consenso que implementa o conceito de comitê de validação de blocos através do uso de funções aleatórias verificáveis (*verifiable random function*). Essas funções permitem que qualquer nó da rede possa verificar se será sorteado para criar o próximo bloco, além de gerar uma prova do sorteio para que os outros nós possam atestar a veracidade do mesmo. Durante a fase anterior, um ou mais blocos podem ser gerados e o protocolo define qual será o bloco vencedor a partir da formação de um comitê de validação. As funções verificáveis são utilizadas novamente para formar o comitê de validação, onde qualquer nó da rede tem chances proporcionais à sua quantidade de moedas de ser sorteado e compor o comitê. Como apenas um único bloco terá maioria absoluta dos votos, o protocolo é livre de *forks*. Para garantir que a blockchain do Algorand tenha a inserção de novos blocos e com isso possa crescer, é necessário que os nós recebam os votos do comitê dentro de um tempo máximo determinado. Isso implica em requisitos rigorosos de atraso de comunicação entre os nós que compõem a rede para que este tempo possa ser atendido.

Esse trabalho analisa de forma aprofundada o mecanismo de consenso baseado em PoS chamado *Probabilistic Proof-of-Stake* (*PPoS*). Este é um novo mecanismo que evoluiu a partir do proposto por Maehara et. al [Maehara et al. 2019], por meio de uma revisão profunda de seu funcionamento e alterações em sua estrutura, o que trouxe novas características e propriedades. O *PPoS* não possui comitê de validação e seus blocos são propostos pelos nós em rodadas de tempo constante. A rodada de produção de um bloco pode ser validada por qualquer um dos nós que compõem a rede, o que impede a aceitação de um bloco com rodada incorreta, ou seja, rodada adiantada ou atrasada além de uma tolerância especificada.

De forma mais concreta, o objetivo deste trabalho é detalhar o *PPoS* e apresentar um modelo teórico sobre a probabilidade de sucessos em uma rodada qualquer do mesmo. Deste modelo são derivados a probabilidade de *forks* e o número esperado de rodadas entre blocos. Os resultados são comprovados por meio de uma implementação do protocolo *PPoS* escrita em Python e executada no ambiente de virtualização de redes *Mininet*.

2. O protocolo *Probabilistic Proof-of-Stake* (*PPoS*)

2.1. Características básicas

O protocolo de consenso *Probabilistic Proof-of-Stake* (*PPoS*) é um algoritmo baseado em sorteios que ocorrem em rodadas de tempo finito com duração T. A cada intervalo de tempo T um novo ciclo e uma nova rodada se iniciam. Os 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, isto é, não seja sorteado, ele deve aguardar uma próxima rodada para tentar criar o bloco novamente.

Para que um nó tenha a oportunidade de criar um bloco em uma dada rodada, ele deve calcular uma função hash usando como entradas a rodada atual, o ID exclusivo do nó e o resultado do hash do bloco imediatamente anterior. Ao utilizar a rodada como um

dos parâmetros, qualquer nó na rede pode compará-la com sua rodada atual para verificar se a rodada proposta no bloco está dentro de uma tolerância conhecida pela rede. O hash do bloco anterior é um parâmetro que vincula o novo bloco proposto com o último aceito na cadeia. Já o ID, associa um nó ao seu endereço na rede, capaz de receber possíveis recompensas em um sorteio bem sucedido.

- ID: identificador do nó.
- r: rodada calculada pelo nó e que será transmitida no cabeçalho do bloco.
- $hash\_anterior$: hash do último bloco aceito na *blockchain*.

Seja $H_{256}(ID||r||hash\_anterior)$ uma função hash com 256 bits de saída, onde a entrada é a concatenação dos bits dos três parâmetros. A Eq.1 deve ser satisfeita para que o nó ID seja autorizado a construir o próximo bloco. Dadas as características de uma boa função de hash criptográfico como SHA-2 ou SHA-3, este teste pode ser considerado um sorteio.

$$H_{256}(ID||r||hash\_anterior) < desafio \tag{1}$$

O desafio é qualquer número inteiro de 256 bits com uma certa quantidade de bits mais significativos iguais a zero. Ele é conhecido por todos os nós que compõem a rede *peer-to-peer*. Os três parâmetros (ID, rodada e hash do bloco anterior) e a própria saída da função hash são colocados no cabeçalho deste bloco. O desafio proposto na Eq. 1 possui cálculos semelhantes aos utilizados no consenso baseado em PoW [Nakamoto 2009], porém não está alicerçado na busca contínua de um *Nonce* capaz de atender o desafio, como ocorre no Bitcoin, por exemplo, já que utiliza parâmetros estáveis em cada rodada. Como o nó está limitado a fazer apenas um cálculo de hash por rodada, o consumo de recursos computacionais por nó é extremamente baixo comparado a outras alternativas como a blockchain do Bitcoin, por exemplo.

Qualquer nó pode verificar se a rodada r presente no cabeçalho de um novo bloco recebido é compatível com a rodada esperada calculada, utilizando a Eq. 2. O quociente da Eq. 2 fornece a quantidade de rodadas inteiras transcorridas entre as chegadas do último bloco aceito na cadeia e o último proposto por algum nó e que deve ser verificado.

$$round_{exp} = round_{lb} + \left\lfloor \frac{t_{arr-cb} - t_{arr-lb}}{T} \right\rfloor \tag{2}$$

Os parâmetros utilizados na Eq. 2 são definidos abaixo:

- $round_{exp}$: rodada esperada calculada pelo nó com seus tempos locais;
- $round_{lb}$: rodada do último bloco aceito;
- t_{arr-cb} : tempo de chegada do bloco atual, ou seja, do bloco em verificação;
- t_{arr-lb} : tempo de chegada do último bloco aceito.

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). Isto é possível, desde que o atraso na rede não seja superior a um máximo tolerado pelo protocolo, ou seja, é necessário que os nós possam calcular a rodada esperada dentro do mesmo intervalo de tempo utilizado pelo nó que criou o bloco. Este limite é o próprio valor definido para T.

Neste sentido, o valor de T deve ser superior ao maior tempo de transmissão de uma mensagem na rede, considerando que a mesma esteja em operação normal. Define-se como operação normal aquela que permite que os nós se comuniquem dentro de um limite de tempo pré-definido. Dessa forma, não deve considerar possíveis desconexões de um ou múltiplos nós, porque isso seria uma condição anormal de operação.

2.2. Critérios para aceitação e confirmação de blocos no *PPoS*

As primeiras motivações para os critérios dessa seção foram apresentadas no trabalho de Martins et al. [Martins and Henriques 2019], que estudou a evolução de cadeias geradas a partir de *forks* em alguns cenários relevantes . No *PPoS* um bloco B é aceito quando sua rodada não está muito atrasada e é, no máximo, igual à rodada de qualquer bloco de mesma posição da blockchain (bloco de mesmo índice) que tenha sido recebido e aceito em uma rodada anterior, caso ele exista. Portanto, um bloco B com rodada de criação x e índice k, será aceito durante a rodada r se atender duas condições:

- **Condição 1:** a rodada x deve pertencer ao intervalo de tolerância definido.
- **Condição 2:** caso exista(m) bloco(s) com índice k recebido e aceito em uma rodada j qualquer, a rodada x do bloco B deve ser no máximo igual a j.

O intervalo de tolerância é variável e depende dos atrasos de relógio e rede que o protocolo deseja tolerar. Este intervalo é definido em função da diferença de rodadas (múltiplo de T_0 segundos) em relação ao tempo real que o protocolo aceita. Caso seja utilizada uma variação máxima de T em relação ao tempo real, dois nós podem estar separados em até $2T$ (um nó atrasado em T e outro adiantado em T em relação ao tempo real), o que justifica um intervalo de tolerância de duas rodadas ($[r-2, r+2]$). Por outro lado, se a tolerância aceitável para erros de relógio em relação a uma referência qualquer for de, no máximo, $\frac{T}{2}$, então a diferença máxima entre relógios poderá chegar a $\frac{T}{2} - (-\frac{T}{2}) = T$. Ou seja, a tolerância neste caso será de apenas uma rodada para mais ou para menos. Em termos gerais podemos escrever o intervalo de tolerância de rodadas como $[r - tol, r + tol]$, onde tol indica a tolerância em número de rodadas.

Após a aceitação do bloco B, ele pode em algum momento ser considerado confirmado, o que ocorre quando não é mais possível criar um novo *fork* a partir de seus antecessores, ou seja, quando um novo bloco não pode ser sucessor de algum bloco que vem antes de B. Além disso, o bloco B não pode fazer parte de algum *fork* que ocorreu antes dele e que ainda não foi resolvido. Desta forma, duas novas condições devem ser atendidas para que um bloco B com rodada x possa ser confirmado no início de uma rodada r qualquer:

- **Condição 3:** no início da rodada r, x deve ser inferior à menor rodada dentro do intervalo de tolerância ($x < r - tol$).
- **Condição 4:** o bloco B não pode fazer parte de nenhum *fork* que ainda não tenha sido resolvido.

3. Análise teórica do *PPoS*

Nessa seção é definido um modelo téorico para análise da probabilidade de *forks* e número de rodadas esperadas entre blocos no *PPoS*. Esse modelo é posteriormente validado através de execuções do *PPoS* que seguem os critérios definidos na Seção 2.2. De

posse da probabilidade de *forks*, é possível ajustar o desafio para que a quantidade esperada de *forks* em função do número de nós permaneça dentro do limite estabelecido pelos usuários, de acordo com a necessidade da aplicação que utiliza o mecanismo.

Além disso, o número esperado de rodadas entre blocos deve permanecer próximo de um, o que garante que o mecanismo não permaneça muitas rodadas sem produzir blocos (fato que ocorre quando o número médio de rodadas entre blocos é maior que um) e que também não produza muitos blocos em uma mesma rodada, gerando muitos *forks* (número de rodadas entre blocos menor que um).

3.1. Probabilidade de sorteio de um nó em cada rodada

Nesta seção é definido um modelo teórico para a probabilidade de um nó ser sorteado em uma rodada. 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*).

A Eq. 3 define a probabilidade de um nó passar no desafio e produzir um novo bloco. O parâmetro D é o desafio utilizado por todos os nós da rede e quanto maior seu valor, menor será a probabilidade de o nó ser sorteado para produzir o próximo bloco.

$$p = \frac{2^{256-D}}{2^{256}} \tag{3}$$

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. Assim, é necessário definir qual é a probabilidade de existirem k cadeias disponíveis no início de uma rodada r qualquer e, para cada valor possível de k, definir a probabilidade de *fork*. São definidas duas novas notações que suportam o cálculo da probabilidade de sorteios bem sucedidos e de *forks*:

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

A Fig.1 mostra a variação possível do parâmetro k na rodada $r = 0$ para $N = 3$.

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

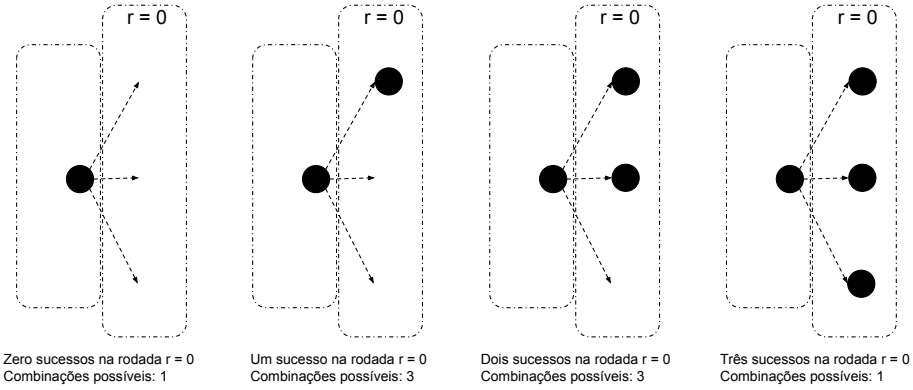

Figura 1. Número de sucessos possíveis na rodada inicial

de cada configuração da Fig. 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 k 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 k novas cadeias a partir de N sorteios é dada pela Eq. 4.

$$P_k^{(0)} = \binom{N}{k}(1-p)^{N-k}p^k \tag{4}$$

Em uma rodada r qualquer, a probabilidade de k sucessos muda em função do número de sucessos que foram obtidos na rodada anterior ($r - 1$). Assim, a probabilidade de ocorrência de k 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 k 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 k sucessos na rodada atual. Como exemplo, na Fig. 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 k cadeias na rodada atual é dado por $S_{min} = max\left(1, \left\lceil \frac{k}{N} \right\rceil\right)$.

Seja i o número de sorteios bem sucedido(cadeias produzidas) em uma rodada $r-1$. É possível que k cadeias sejam geradas na rodada r se $i \geqslant S_{min}$ 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 k sucessos bem sucedidos quando i cadeias estão disponíveis é dada por $\binom{iN}{k}(1-p)^{iN-k}p^k$. 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 k sorteios bem sucedidos com i cadeias disponíveis é dada por $\binom{iN}{k}(1-p)^{iN-k}p^k P_i^{(r-1)}$.

Na rodada $r-1$, é possível que diferentes quantidades de sorteios bem sucedidos sejam capazes de produzir k cadeias na rodada r (qualquer quantidade que atenda a condição $i \geqslant S_{min}$). 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 k 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 $\sum_{i=S_{min}}^{N^r} \binom{iN}{k}(1-p)^{iN-k}p^k P_i^{(r-1)}$.

Todo este desenvolvimento foi para valores não nulos de i. Mesmo para o caso de $k = 0$, 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 k 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 k sucessos na rodada r utilizando o número de sucessos da rodada $r-2$ é dado por $P_k^{(r)}P_0^{(r-1)}$ e, $P_k^{(r)}$ com zero sorteios bem sucedidos na rodada $r-1$, é o mesmo que calcular a probabilidade $P_k^{(r-1)}$ com a rodada atual sendo r, porém com a premissa de que durante a rodada $r-1$ zero sucessos ocorreram ($P_k^{(r)}P_0^{(r-1)} = P_k^{(r-1)}P_0^{(r-1)}$). Além disso, existe a probabilidade da rodada $r-2$ também não ter nenhum sucesso, o que corresponde a calcular $P_k^{(r-2)}$ na rodada atual r, sabendo que durante as rodadas $r-1$ e $r-2$ nenhum sucesso ocorreu ($P_k^{(r-2)}P_0^{(r-2)}P_0^{(r-1)}$). 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 k sucessos na rodada r quando nenhum sorteio bem sucedido ocorre na rodada $r-1$. É necessário, portanto, definir esta rodada mínima possível. A Eq. 5 define o termo R_{min} como sendo a rodada minima capaz de produzir os S_{min} sucessos.

$$\begin{aligned} R_{min} &= 1, \;\; se\ k = 0\ ou\ k = 1 \\ R_{min} &= \lceil \log_N k \rceil, \;\; se\ k > 1 \end{aligned} \tag{5}$$

Assim, para cada rodada $t \geqslant R_{min}$ é calculada a probabilidade de se obter k sucessos a partir dos sucessos desta rodada ($P_k^{(t)}$) com a respectiva probabilidade de todas as rodadas maiores que t apresentarem zero sucessos. O termo $X_k^{(r)}$ (Eq. 6) representa a contribuição total da probabilidade de k sucessos em uma rodada $r > 0$ quando nenhum sucesso ocorre na rodada $r-1$.

$$X_k^{(r)} = \sum_{t=1}^{r-(R_{min}-1)} \left(P_k^{(r-t)} \prod_{j=1}^{t} P_0^{(r-j)} \right) \tag{6}$$

Deste modo, a Eq. 7 apresenta a probabilidade de k sucessos na rodada r com N nós para $r > 0$ e $0 \leq k \leq N^{r+1}$.

$$P_k^{(r)} = X_k^{(r)} + \sum_{i=S_{min}}^{N^r} \binom{iN}{k} (1-p)^{iN-k} p^k P_i^{(r-1)} \tag{7}$$

3.2. Probabilidade de *fork* no *PPoS*

A Eq. 8 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.

$$\overline{p_f} = (1-p)^N + \binom{N}{1}(1-p)^{N-1}p = (1-p)^N + N(1-p)^{N-1}p \tag{8}$$

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 (k) disponíveis no início desta rodada (geradas a partir de sucessos em sorteios na rodada $r-1$). Para cada uma das k 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 ($i \leq k$) é dada por $P_k^{(r-1)} \sum_{i=0}^{k-1} \binom{k}{i} (p_f)^{k-i} (\overline{p_f})^i$. O termo $P_k^{(r-1)}$ é necessário, pois o número de cadeias disponíveis (k) no início da rodada r depende da probabilidade de que na rodada $r-1$ tenham ocorrido k 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 k em seu intervalo possível ($1 \leq k \leq N^r$) para a rodada $r-1$, resultando em $\sum_{k=1}^{N^r} P_k^{(r-1)} \sum_{i=0}^{k-1} \binom{k}{i} (p_f)^{k-i} (\overline{p_f})^i$.

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 $X_k^{(r)}$ 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. 9) mostra a parcela da probabilidade de *forks* na ausência de sucessos na rodada anterior.

$$Y^{(r)} = p_f \prod_{j=1}^{r} P_0^{(r-j)} + \sum_{t=1}^{r-1} \left(P_f^{(r-t)} \prod_{j=1}^{t} P_0^{(r-j)} \right) \tag{9}$$

O primeiro termo da Eq. 9 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. 10.

$$P_f^{(r)} = Y^{(r)} + \sum_{k=1}^{N^r} P_k^{(r-1)} \sum_{i=0}^{k-1} \binom{k}{i} (p_f)^{k-i} (\overline{p_f})^i \tag{10}$$

3.3. Número esperado de rodadas entre blocos

É importante conhecer o número esperado de rodadas entre dois blocos consecutivos no *PPoS* para que, por meio do ajuste do desafio D, os nós sejam capazes de produzir um bloco por rodada em média. Isso evita que sejam geradas grandes quantidades de rodadas ociosas, além de controlar a criação de *forks*. O número de rodadas entre blocos depende da probabilidade q de um ou mais nós serem sorteados. Este valor é obtido subtraindo de 1 o valor da probabilidade de que nenhum nó seja sorteado ($q = 1 - (1-p)^N$). Desta forma, no início da rodada $r = 0$, o número de rodadas esperadas $R^{(0)}$ para produção do próximo bloco é dado por $R^{(0)} = \lceil 1/q \rceil$.

Como ocorre com a probabilidade de produção de *forks*, o número esperado de rodadas entre blocos depende do número de cadeias disponíveis no início da rodada em análise. Consideremos o caso em que existam duas cadeias disponíveis no início de uma rodada r qualquer. Todos os nós podem produzir blocos em qualquer uma das duas cadeias, o que reduz o número esperado de rodadas entre blocos. Assim, o próximo bloco será produzido se qualquer um dos N nós passar no desafio para alguma das duas cadeias disponíveis, o que ocorre com probabilidade q para cada uma das cadeias.

De acordo com esta análise, quanto maior o número de cadeias disponíveis menor será o número de rodadas esperadas entre blocos no início de uma rodada r qualquer. O número de cadeias disponíveis no início da rodada r depende do número de sucessos alcançados pelos nós durante a rodada $r-1$. Caso tenham havido k sucessos na rodada $r-1$, é possível produzir o próximo bloco para qualquer uma das cadeias contempladas. Portanto, o número esperado de rodadas para este caso é dado pela Eq. 11.

$$P_k^{(r-1)} \frac{1}{\sum_{i=0}^{k-1} \binom{k}{i} q^{k-i}(1-q)^i} \tag{11}$$

O termo $P_k^{(r-1)}$ fornece a probabilidade de existirem k cadeias disponíveis no início da rodada r. Quando nenhum sucesso ocorre na rodada $r-1$ é necessário considerar os sucessos das rodadas menores que $r-1$ como válidos na rodada r. O parâmetro $Z^{(r)}$ é mostrado na Eq. 12 e representa a contribuição do número esperado de rodadas a partir da rodada r quando nenhum bloco foi sorteado na nas rodadas abaixo de r.

$$Z^{(r)} = \sum_{t=1}^{r} R^{(r-t)} \prod_{j=1}^{t} P_0^{(r-j)} \tag{12}$$

Deste modo, o número esperado de rodadas no início de uma rodada $r > 0$ pode ser calculado em função da soma de todas as parcelas produzidas por meio da variação do número de sorteios bem sucedidos (k) produzidos na rodada $r-1$ e ponderados pela probabilidade de que cada um deles ocorra, como mostra a Eq. 13.

$$R^{(r)} = \left\lceil Z^{(r)} + \sum_{k=1}^{N^r} P_k^{(r-1)} \frac{1}{\sum_{i=0}^{k-1} \binom{k}{i} q^{k-i}(1-q)^i} \right\rceil \tag{13}$$

4. Resultados práticos

Os resultados apresentados nessa seção foram obtidos através de uma implementação do *PPoS* 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 protocolo *PPoS* é um host conectado a uma interface de um switch OpenFlow, como mostrado na Fig. 2.

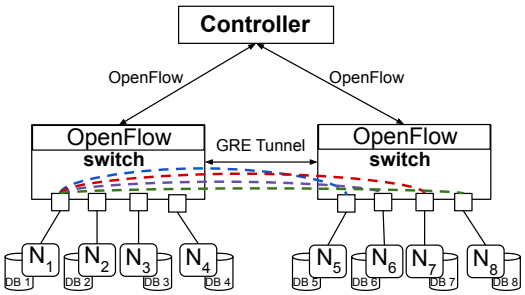

Figura 2. Arquitetura de rede do experimento prático

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 *PPoS* 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 Fig. 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 protocolo, 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.

4.1. Criação de *forks*

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

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 3. Os resultados permaneceram próximos do valor teórico esperado

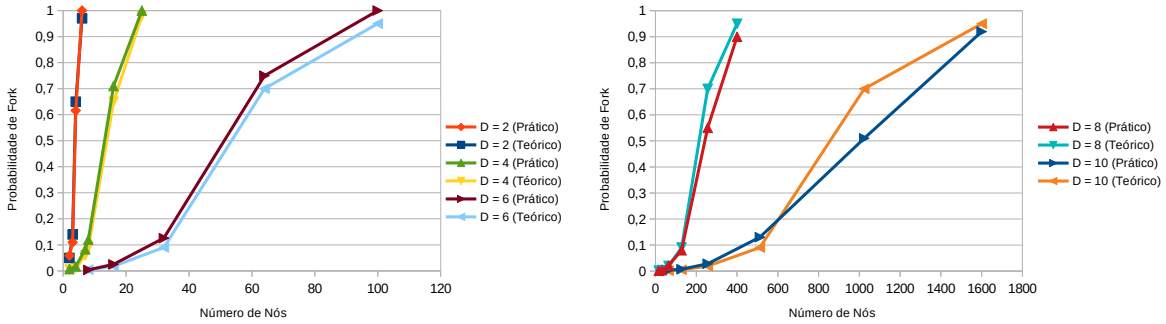

Figura 3. Probabilidade de forks para diferentes desafios

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 aumento de duas unidades no desafio D, como já era previsto pela Eq. 3: um aumento de duas unidades no desafio D provoca uma redução de quatro vezes na probabilidade de sucesso p. Portanto, um aumento no desafio D reduz a probabilidade de *forks* para a mesma quantidade de nós.

Como a maioria dos consensos determinísticos evitam os forks, o comportamento de *PPoS* neste aspecto deve ser comparado com outros protocolos que também geram forks, como ocorre na blockchain do Bitcoin, por exemplo. Em ambos os protocolos, é possível controlar a ocorrência de *forks* a partir da calibração do desafio associado à dificuldade do sorteio. Neste sentido, é importante conhecer a probabilidade de *forks* em função do número de nós para que o desafio D seja escolhido com o objetivo de minimizá-los, sem contudo deixar a blockchain ociosa. Além disso, com um desafio adequado, caso algum *fork* seja produzido, o mecanismo será capaz de convergir rapidamente para apenas uma cadeia, já que um desafio equilibrado contribui para que os nós não consigam manter as duas cadeias em condição de igualdade por muitas rodadas e, assim, a convergência ocorre seguindo os critérios estabelecidos na seção 2.2.

4.2. Número de rodadas esperadas entre blocos

O número esperado de rodadas entre blocos ($R^{(r)}$) foi analisado nas mesmas simulações da seção anterior e os resultados estão apresentados nos gráficos da Fig. 4.

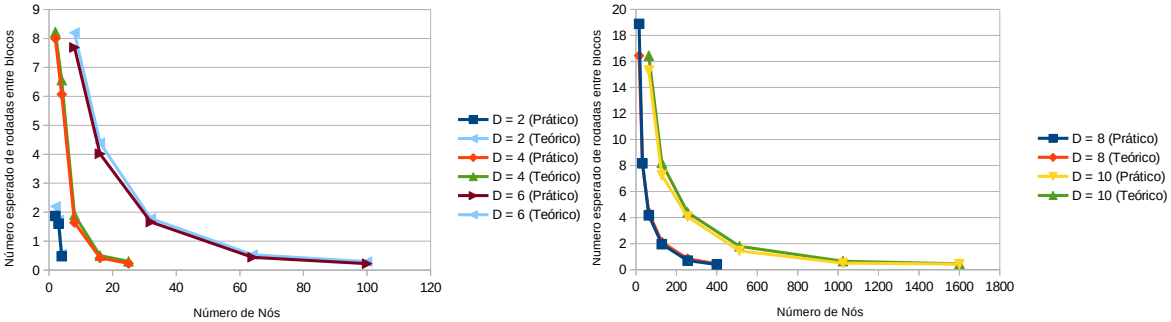

Figura 4. Número médio de rodadas entre blocos

De acordo com o gráfico da Fig. 4 o número médio de rodadas entre blocos diminui quando o número de nós aumenta para um desafio qualquer. Este comportamento é esperado já que com uma maior quantidade de nós espera-se um maior número de blocos e consequentemente um aumento da probabilidade do próximo bloco ser criado com menos rodadas. Este efeito é desejado, pois reduz o número de rodadas que o protocolo fica sem produzir blocos aumentando a eficiência de geração de blocos.

O número esperado de rodadas entre blocos não deve ficar abaixo de um. Neste caso, a rede está gerando mais de um bloco por rodada, o que contribui para a geração de *forks* indesejados. É necessário que o desafio seja calibrado para que poucos *forks* sejam gerados e também para que o protocolo não permaneça muitas rodadas sem produzir blocos. Esse objetivo é atingido quando o protocolo produz em média um bloco por rodada. Como exemplo, de acordo com a Fig. 4, o desafio deve ser 10 quando o número de nós está na faixa de 500 a 1600.

5. Conclusões e Trabalhos Futuros

O algoritmo de consenso *PPoS* é uma alternativa para blockchains públicas que busca no sincronismo de rodadas uma forma de evitar que nós com alta disponibilidade computacional possam levar grandes vantagens na produção de novos blocos. Isto acontece porque mesmo que eles possam calcular rodadas futuras e conhecer os momentos que eles serão sorteados, não terão nenhuma vantagem, já que eles devem esperar efetivamente o início de cada rodada.

O modelo teórico para a probabilidade de *forks* foi utilizado como instrumento de balizamento para os resultados práticos obtidos nas execuções do protocolo *PPoS*. Este modelo se mostrou adequado para as execuções em que o *PPoS* não considera nenhum intervalo de tolerância de rodadas ($tol = 0$). Além disso, é possível notar que mesmo nesse cenário de pior caso, onde os nós do *PPoS* tentam produzir blocos em qualquer cadeia conhecida, foi possível confirmar os 100 blocos em todos os casos analisados o que mostra que houve convergência em torno de uma única cadeia.

O comportamento do número de rodadas entre blocos está dentro do esperado. Quando aumentamos o número de nós, a rede produz muitos blocos e, com isso, o número de rodadas diminui. Ao diminuir o número de nós para o mesmo desafio, nota-se o efeito contrário, ou seja, é observado um número maior de rodadas sem produzir blocos. Os dois extremos desta análise são indesejados no *PPoS* e, portanto, espera-se que a produção de blocos esteja o mais próximo possível de um bloco por rodada, o que deve ser controlado pelo desafio D.

Como trabalhos futuros, temos a necessidade de implementar um mecanismo de controle de uso do *stake* no *PPoS* para controlar o problema do nothing at stake e de definir a identidade do nó de modo que não seja possível a utilização desse ID como uma espécie de $nonce$, evitando que o nó possa realizar diferentes sorteios em uma mesma rodada alterando sua identidade.

Referências

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

Buterin, V. and Griffith, V. (2017). Casper the friendly finality gadget. ArXiv e-prints, https://arxiv.org/pdf/1710.09437.pdf.

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.

Deirmentzoglou, E., Papakyriakopoulos, G., and Patsakis, C. (2019). A survey on long-range attacks for proof-of-stake protocols. IEEE Access. https://ieeexplore-ieee-org.ez338.periodicos.capes.gov.br/document/8653269, 7:28712 – 28725.

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., Hemo, R., Micali, S., Vlachos, G., and Zeldovich, N. (2017). Algorand: Scaling byzantine agreements for cryptocurrencies. SOSP 17 - Proceedings of the 26th Symposium and Operating Systems Principles. https://dl.acm.org/doi/abs/10.1145/3132747.3132757.

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).

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. E., Martins, D. F. G., and Henriques, M. A. A. (2019). Proof-of-stake baseado em tempo discreto. In XXXVII Simpósio Brasileiro de Redes de Computadores e Sistemas Distribuídos - Workshop -WBlockchain, Gramado-RS. http://sbrc2019.sbc.org.br/wp-content/uploads/2019/05/wblockchain2019.pdf.

Martins, D. F. G. and Henriques, M. A. A. (2019). 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 Sistemas Distribuídos - trilha principal, São Paulo-SP. https://sbseg2019.ime.usp.br/anais/196087.pdf.

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

Nguyen, C. T., Hoang, D. T., Nguyen, D. N., Niyato, D., Nguyen, H. T., and Dutkiewicz, E. (2019). Proof-of-stake consensus mechanisms for future blockchains networks: Fundamentals, applications and opportunities. IEEE Access. https://ieeexplore.ieee.org/document/8746079, 7:85727 – 85745.