Uma análise preliminar do controle de forks no mecanismo de consenso Proof-of-Stake com Tempo Discreto

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 preliminary analysis of the blockchain consensus mechanism based on Discrete-Time Proof-of-Stake (DTPoS). The text explains the mechanisms that controls the creation of new blocks and the elimination of forks in the chain. Next, the consensus mechanism is analyzed in a generic way, through different scenarios in the view of a given node. It is shown how the algorithm reaches consensus even if network delays occur, provided that the duration time of a work round is larger than the largest delay between any two connected nodes in the system.*

Resumo. *Este artigo apresenta uma análise preliminar do mecanismo de consenso para blockchains baseado em Proof-of-Stake com Tempo Discreto (PoSTD). O texto explica os mecanismos que controlam a criação de novos blocos e a eliminação de forks na cadeia. Em seguida, o mecanismo de consenso é analisado de forma genérica, através de diferentes cenários na visão de um determinado nó. É mostrado como o algoritmo atinge o consenso mesmo com atrasos na rede, desde que o tempo de duração de uma rodada de trabalho seja superior ao maior dos atrasos de comunicação entre dois nós quaisquer conectados no sistema.*

1. Introdução

Os algoritmos de consenso em blockchains permitem que uma transação entre duas partes seja validada sem que exista uma terceira parte confiável. 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 propostas de consenso não baseadas em Proof-of-Work (PoW) [Nakamoto 2009] estão despertando interesse, em parte porque este mecanismo enfrenta problemas ligados ao desperdício de energia elétrica em sua operação, já que, apesar de todos os nós colocarem muito esforço no processo de criar um novo bloco, apenas o trabalho de um deles é aproveitado. Além disso, o PoW acaba privilegiando os nós que possuem mais recursos financeiros disponíveis para investir em hardware dedicado (*asics - application-specific integrated circuit*), que tem desempenho muito melhor que processadores de propósito geral ou GPUs. Isto faz com que o processo de geração dos novos blocos esteja centralizado em poucos e poderosos nós, enquanto

a maioria da rede não tem capacidade suficiente de investimento para competir em igualdade de condições. Esta centralização é indesejada em uma rede *peer-to-peer*, pois alguns desses poucos nós privilegiados poderiam forçar um falso consenso por meio de um ataque conhecido como Ataque de 51%, resultado de um conluio entre os detentores de mais da metade do poder computacional total disponível. Tal ataque permitiria que registros antigos na blockchain fossem adulterados, viabilizando, por exemplo, que uma dada criptomoeda fosse utilizada duas vezes, por exemplo.

O Proof-of-Stake (PoS) é uma proposta alternativa ao PoW, onde a prova de posse do *stake* é necessária para um nó poder criar blocos e receber eventuais recompensas envolvidas com este processamento [King 2013]. Existem algumas variações desse mecanismo de consenso, principalmente relacionadas ao modelo de remuneração dos nós criadores de blocos. No algoritmo Casper [Buterin and Griffith 2017] não existe remuneração por blocos. Ao invés disso, o protocolo cria um grupo de validadores capazes de votar em blocos especiais chamados *checkpoints*. Todos os nós que participam dessa votação são remunerados de maneira proporcional ao *stake* que eles depositaram como garantia.

O objetivo deste trabalho é iniciar uma análise mais profunda de um novo mencanismo de consenso chamado PoSTD, proposto por Maehara et. al [Maehara et al. 2019], considerando a visão local que um nó possui da rede e da cadeia de blocos. É mostrado que, desde que algumas condições básicas de operação sejam respeitadas, é possível garantir o consenso entre os diversos nós que competem pelo direito de criar novos blocos.

2. Proof-of-Stake com Tempo Discreto (PoSTD)

Nessa seção serão explicados os detalhes básicos de funcionamento do protocolo de consenso baseado em Proof-of-Stake com Tempo Discreto [Maehara et al. 2019]. Este mecanismo é baseado em sorteios que ocorrem em rodadas de tempo com duração T_0 segundos. A cada intervalo de tempo T_0 um novo ciclo (rodada) se inicia, permitindo que os nós participem novamente do sorteio para gerar o próximo bloco da cadeia. Caso um nó não seja sorteado, ele deve aguardar uma próxima rodada para nova tentativa. Esta espera evita que nós com alto poder computacional usem suas vantagens contra os mais simples.

Para poder criar o próximo bloco, um nó deve participar de um sorteio que consiste em gerar um número pseudoaleatório obtido através da submissão de alguns parâmetros a uma função hash de 256 bits (SHA-256). O objetivo é que o número obtido do hash seja menor que um valor pré-determinado (conhecido como "desafio") e compartilhado entre todos os nós que compõem a rede *peer-to-peer*. Segundo Maehara et. al [Maehara et al. 2019] este hash é utilizado como prova de que um determinado participante foi sorteado. Desta forma, o valor do hash pode ser chamado de *hash_de_prova*.

O *hash_de_prova* é calculado em um dado nó por $H_b = H_{256}(ID||r||hash\_anterior)$ e usa como entradas o número da rodada atual (r), o ID exclusivo do nó e o hash do último bloco da cadeia ($hash\_anterior$). Utilizar a rodada e o hash do bloco anterior como parâmetros do sorteio permite que qualquer nó na rede verifique se a rodada inserida no bloco proposto é ou não aceitável. Além disso, o hash do bloco anterior vincula o novo bloco proposto com o último aceito na cadeia. Já o ID associa univocamente um nó a seu endereço na rede *peer-to-peer*, de maneira a evitar fraudes e ser capaz de receber possíveis recompensas em um sorteio bem sucedido.

Se sorteado, H_b é considerado o hash que representa o bloco em questão e, junto com os três parâmetros (ID, r e $hash\_anterior$), é colocado no cabeçalho do bloco criado.

Após receber um bloco, o nó pode verificar se o desafio foi satisfeito e se a rodada do mesmo é a correta, utilizando a Eq. (1) para calcular a rodada atual com base na diferença entre o tempo atual (chegada do referido bloco) e o instante de chegada do último bloco aceito como válido.

$$\begin{aligned} rodada\_atual &= último\_bloco.rodada \\ &+ \left\lfloor \frac{tempo\_atual() - último\_bloco.tempo\_chegada}{T_0} \right\rfloor \end{aligned} \quad (1)$$

Usar o tempo de chegada do bloco é uma forma de evitar que um nó destinatário tenha que lidar com problemas de sincronização de relógios em nós diferentes, já que o tempo de chegada pode ser considerado, para efeitos práticos, igual ao tempo de criação do bloco.

Um ponto importante no protocolo é a definição do valor T_0, que deve ser maior que o maior dos atrasos entre dois nós conectados quaisquer. Dessa forma, o valor de T_0 não considera possíveis desconexões de um ou múltiplos nós, já que isso seria uma condição anormal de operação. Quando um nó passa por uma longa desconexão, eventuais blocos propostos por ele podem ser rejeitados pela rede quando a conexão for reestabelecida, pois ele provavelmente estará dessincronizado. Um nó que perde o sincronismo com a cadeia, poderá ter que ressincronizar com seus *peers*, já que blocos que ele ainda não conhece já podem ter sido criados por outros nós. Considerando os tempos atuais de atraso nas redes de dados, o valor de T_0 deverá ser definido na ordem de segundos. Mesmo que a produção de blocos esteja condicionada à espera por uma nova rodada, percebe-se que este esquema é mais eficiente que o adotado no protocolo mais disseminado atualmente: o PoW do Bitcoin, que só produz um novo bloco a cada 10 minutos, em média.

Os *forks* também são tratados pelo protocolo PoSTD. Eles ocorrem quando são criados dois ou mais blocos sucessores de um mesmo bloco e qualquer mecanismo de consenso deve dificultar seu surgimento e a evolução de possíveis cadeias secundárias originadas nestes pontos de bifurcação. No PoSTD os *forks* são controlados através de penalizações impostas a um determinado participante que cria blocos em cadeias secundárias [Maehara et al. 2019]. As penalizações produzem como efeito prático um aumento da dificuldade de produção de blocos nas cadeias mais curtas e, como consequência, a cadeia principal tem maior probabilidade de aumentar sua distância (em número de blocos) em relação a elas. Em linhas gerais, a penalização é zero quando o bloco é criado para a cadeia principal e maior que zero para cadeias secundárias, isto é, menores que a principal. A motivação para esta estratégia reside no fato de que é possível que nós distantes tenham visões diferentes sobre quais cadeias são secundárias e qual é a principal, permitindo, ainda que com menor probabilidade, a criação de blocos para cadeias que o nó local entende serem secundárias.

Para a obtenção de consenso, não basta desestimular a criação de $forks$: é preciso também haver uma forma de eliminar as cadeias que não têm mais chances de crescer e se tornar a cadeia principal. Na próxima seção são detalhadas e avaliadas as formas que o mecanismo de consenso PoSTD usa para a eliminação de *forks* (cadeias secundárias). São analisados vários cenários que representam diferentes situações em que cadeias secundárias devem ou não ser removidas.

3. Análise da evolução de cadeias na visão de um nó

É de fundamental importância para qualquer algoritmo de consenso a existência de uma condição para remoção de cadeias secundárias, sempre que não exista mais a possibilidade da mesma se tornar a cadeia principal (a cadeia mais longa). A análise da chegada ou da criação de um novo bloco em um nó permite entender como o protocolo PoSTD trata a evolução das cadeias principais e secundárias. Como o mesmo protocolo é usado em todos os nós do sistema, entende-se que, com o passar do tempo, ele leve a maioria dos nós a um consenso sobre a remoção ou não de determinados blocos da *blockchain*.

Nesse cenário, onde múltiplas cadeias podem estar presentes em um nó, identificaremos um bloco pela notação $B_{i,j,k}$ onde os subíndices indicam:

- i – hash H_b do bloco ;
- j – rodada de criação;
- k – hash do bloco anterior $(hash\_anterior)$.

Nas próximas subseções serão discutidas as decisões que um nó deve tomar, baseado em eventos que ocorrem dentro de uma rodada e na visão local que ele possui da cadeia e da rede. São tratados os eventos de chegada de blocos de maneira atrasada e em fase com a rodada atual local. Os atrasos são provocados por congestionamentos e por tempos de propagação na rede. Além disso há as diferenças de tempo causadas por atrasos ou avanços do relógio de um nó em relação aos de outros nós, problemas que serão objeto de um trabalho futuro. Serão discutidos eventos de chegada e de geração de blocos em um nó específico.

3.1. Cenário 1: nó não produz e nem recebe nenhum novo bloco em uma rodada

Esse estado faz com que o nó permaneça com todas as cadeias locais exatamente como estavam antes da rodada analisada, ou seja, na visão desse nó, nenhum outro participante na rede gerou algum bloco ou então um bloco foi gerado mas não chegou. Além disso, o nó não conseguiu gerar um novo bloco, pois não foi sorteado durante a rodada. Vamos considerar que no início de uma rodada r qualquer o nó está com a visão de duas cadeias e elas estão empatadas (exatamente com a mesma quantidade de blocos, devido a um fork ocorrido anteriormente). Os últimos blocos dessas cadeias foram gerados na rodada $r - j$, como mostra a Fig. 1.

Consideremos então que o Nó Remoto 1 propôs um bloco na rodada r, mas este não chegou até o Nó Local antes do final dessa rodada. Assim, no início da rodada $r + 1$, a visão do Nó Local permanece a mesma.

3.2. Cenário 2: nó não produz, mas recebe um novo bloco em uma rodada

Ainda analisando a Fig 1 nota-se que um bloco gerado em uma rodada r, pode chegar ao Nó Local durante a rodada $r + 1$. A partir da Eq. 1 o nó calcula a rodada esperada do bloco e a diferença com a rodada contida no cabeçalho do bloco. Como esta diferença é de uma unidade (ou seja, um intervalo de no máximo T_0), então considera-se que é um bloco válido, pois teve que atender o desafio da rodada para ser criado e o tempo até sua chegada não violou o limite T_0. Na Fig.1 vemos que, no início da rodada $r + 2$, a cadeia, representada por $B_{c,r-i,p} \rightarrow B_{d,r-j,c} \rightarrow B_{e,r,d}$, é a mais longa, já que seu valor foi atualizado na chegada do último bloco.

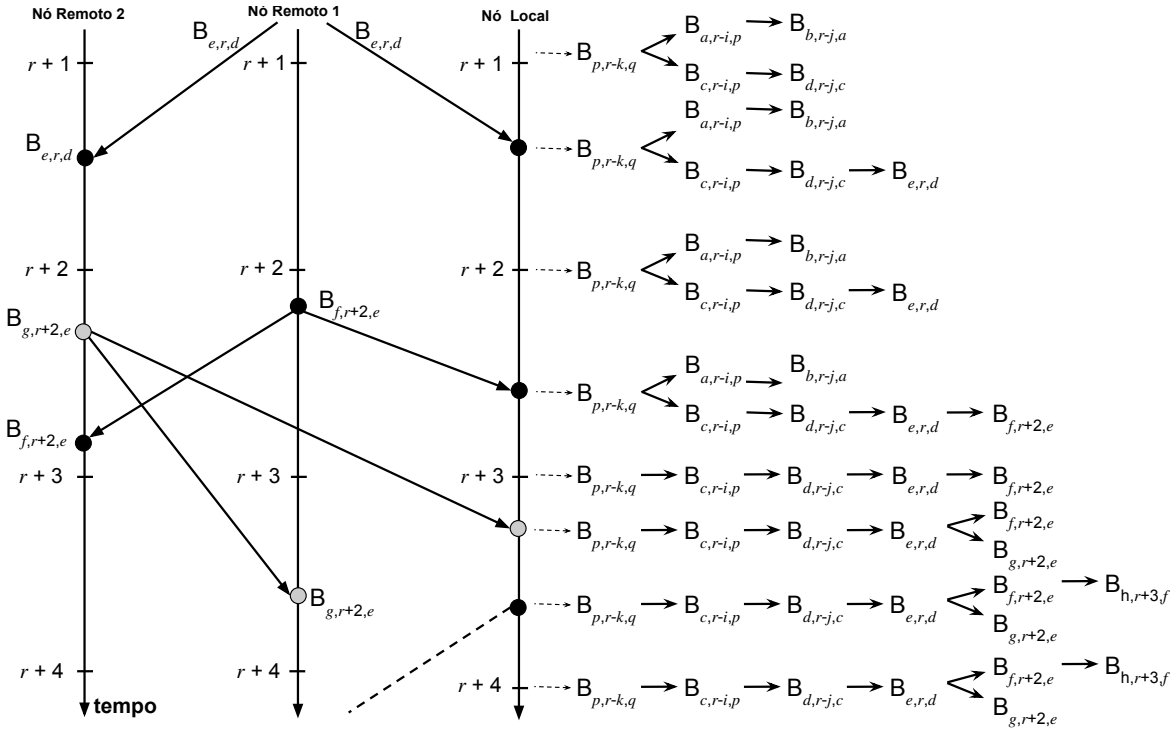

Figura 1. Impacto da geração e recepção de blocos na criação e eliminação de *forks* na cadeia de blocos.

O Nó Local irá checar se existem cadeias que podem ser abandonadas no início de cada rodada. A cadeia $B_{c,r-i,p} \rightarrow B_{d,r-j,c}$ possui um bloco a menos em relação à cadeia mais longa e é uma candidata a eliminação no início de $r + 2$. Após o fim da rodada $r + 1$, esta cadeia mais curta não poderá mais se igualar em tamanho à cadeia mais longa se receber um bloco gerado na rodada r, pois este estará com um atraso maior que T_0. Entretanto, ela ainda poderá receber um bloco criado na rodada $r + 1$ e voltar a se igualar em tamanho à cadeia mais longa. Por este motivo, o protocolo não remove a cadeia mais curta. Ou seja, ele só remove uma cadeia ao final de uma rodada se a diferença de tamanho com outras mais longas for maior que um.

3.3. Cenário 3: nó produz e recebe blocos em uma rodada

No início da rodada $r + 3$ o Nó Local fica com apenas uma cadeia, pois como o bloco $B_{f,r+2,e}$ foi recebido dentro da rodada $r + 2$, a cadeia mais curta ficou com dois blocos de diferença em relação a cadeia principal e foi eliminada na troca de rodadas. Posteriormente, o bloco $B_{g,r+2,e}$ chega dentro do intervalo de uma rodada (T_0) e é aceito como um ponto de *fork* para uma nova cadeia. O Nó Local nesse momento volta a ter duas cadeias empatadas, porém logo depois ele próprio consegue passar no desafio e propor um novo bloco ($B_{h,r+3,f}$), aumentando outra cadeia. Na transição de $r + 3$ para $r + 4$ o nó continua com as duas cadeias, já que a diferença de tamanho entre elas é de apenas um bloco, situação semelhante à ocorrida na transição de $r + 1$ para $r + 2$.

3.4. Outros cenários

São várias as combinações possíveis de chegada e de geração de blocos em um determinado nó que suporta a blockchain. Entretanto, elas podem ser mapeadas em quatro

cenários, sendo três destes os apresentados acima. O cenário não detalhado é aquele em que um nó produz e transmite um novo bloco para seus pares, não recebendo nenhum outro durante a rodada. Tal cenário irá implicar na mesma atualização de cadeias do nó como explicado para o cenário em que ele só recebe um bloco (Cenário 2).

4. Conclusões e Trabalhos Futuros

O mecanismo de consenso baseado em PoS com Tempo Discreto é uma abordagem conservadora para *blockchains* públicas. A criação de blocos em rodadas discretas garante previsibilidade, pois evita que blocos sejam criados e aceitos fora de intervalos bem definidos, diminuindo a ocorrência de *forks* e suas cadeias secundárias indesejadas.

A análise preliminar apresentada neste artigo mostrou que o mecanismo de consenso é eficiente para a eliminação de cadeias secundárias indesejáveis. Mostrou ainda que ele é bem simples, pois evita a complexidade de outros como, por exemplo, o baseado em validações de *checkpoints* proposto no protocolo PoS Casper do Ethereum. Boa parte desta simplicidade do protocolo decorre do fato de ele trabalhar com a criação, recebimento e verificação de blocos apenas em intervalos de tempo bem definidos. Isto facilita a compreensão e avaliação de seu funcionamento e ainda permite a criação de blocos em uma taxa muito superior que aquela obtida no protocolo PoW do Bitcoin, por exemplo.

Como trabalhos futuros, planejamos validar experimentalmente todos os cenários de criação e eliminação de cadeias secundárias (*forks*). Além disso procuraremos determinar o valor mínimo do intervalo T_0 que ainda garanta o funcionamento correto do protocolo, sem criar instabilidades ou gerar *forks* desnecessariamente.

Referências

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

Buterin, V. and Griffith, V. (2017). Casper the friendly finality gadget. `https://arxiv.org/pdf/1710.09437v2`. (Acessado em 10/05/2019).

King, S. (2013). Primecoin: Cryptocurrency with prime number proof-of-work. `http://primecoin.io/bin/primecoin-paper.pdf`. (Acessado em 20/06/2019).

Maehara, Y., Martins, D. F. G., and Henriques, M. A. A. (2019). Proof-of-stake baseado em tempo discreto. Anais SBRC 2019, http://sbrc2019.sbc.org.br/wp-content/uploads/2019/05/wblockchain2019.pdf, pág. 112-125.

Nakamoto, S. (2009). Bitcoin: A peer-to-peer electronic cash system. `https://bitcoin.org/bitcoin.pdf`. (Acessado em 10/01/2019).