vCubeChain: Uma Blockchain Permissionada Escalável

Allan Edgard Silva Freitas[1], **Luiz Antonio Rodrigues**[2] **e Elias Procópio Duarte Jr.**[3]

[1]Instituto Federal da Bahia (IFBA), Campus de Salvador, Bahia, Brasil

[2]Universidade Estadual do Oeste do Paraná (UNIOESTE), Cascavel, Paraná, Brasil

[3]Universidade Federal do Paraná (UFPR), Curitiba, Paraná, Brasil

allan@ifba.edu.br, Luiz.Rodrigues@unioeste.br, elias@inf.ufpr.br

Resumo. *Este trabalho apresenta a blockchain permissionada vCubeChain, que é escalável por definição, sendo baseada na topologia distribuída dinâmica vCube. O vCube é um detector de falhas que conecta um conjunto de n processos corretos em uma topologia virtual hierárquica, que é um hipercubo quando todos os processos estão corretos. Quando os processos falham por parada, o vCube se reorganiza mantendo diversas propriedades logarítmicas. A vCubeChain elege um líder, que utiliza uma estratégia autonômica de difusão confiável para disseminar blocos na rede. Cada bloco consiste de múltiplas transações. Múltiplos líderes concorrentes podem ser eleitos, caso o detector de falhas levante falsas suspeitas. Ainda assim, é provado que a vCubeChain se mantém íntegra, sempre retornando a um estado consistente. No artigo, além da especificação e provas de corretude, um conjunto de experimentos é apresentado demonstrando o desempenho, em particular a escalabilidade da solução em comparação a outras alternativas.*

1. Introdução

Uma blockchain é um *livro-razão* distribuído que permite o armazenamento confiável e seguro de registros de transação em um conjunto de processos conectados através de uma rede. A grande vantagem das blockchains em comparação com outras tecnologias alternativas é que suas propriedades de segurança não requerem nenhuma entidade centralizada garantidora [Gamage et al. 2020]. Um número enorme e crescente de aplicações têm sido propostas para blockchains, em campos diversos, como criptomoedas, governo digital, proteção de *copyright*, transações imobiliárias, entre outras [Chen et al. 2018].

Blockchains combinam técnicas de computação distribuída e segura para manter uma estrutura de dados – a cadeia de blocos – que garante a persistência de transações armazenadas pelos processos que integram o sistema. É possível classificar as blockchains em dois tipos básicos: permissionadas (privadas) e não-permissionadas (públicas). Uma blockchain permissionada requer que os processos sejam conhecidos e devidamente autenticados para participarem do sistema. Um exemplo é a Hyperledger Fabric [Androulaki et al. 2018]. Por outro lado, nas blockchains não-permissionadas qualquer processo pode participar. O processos não precisam sequer confiar uns nos outros. Exemplos incluem a Bitcoin [Nakamoto 2008] e Ethereum 1.0 [Wood 2014]. Estas blockchains utilizam um mecanismo de consenso baseado em "Prova-de-Trabalho" (*Proof-of-Work* - PoW) para validar novos blocos com base na capacidade de processamento dos usuários, o que resulta em alto gasto energético. O processo vencedor recebe uma recompensa,

no caso do Bitcoin a própria criptomoeda. Recentemente, em setembro de 2022, a Ethereum 2.0 passou a utilizar a estratégia de "Prova de Participação" (*Proof-of-Stake* - PoS), que requer que os usuários façam "empenho" de moeda, no caso o Ether (ETH), para se tornarem validadores da rede [Ethereum.org 2022].

Nas blockchains permissionadas, os processos confiam uns nos outros. Assim, ao invés de PoW ou PoS elas podem utilizar algoritmos clássicos de consenso, como Raft [Ongaro and Ousterhout 2014] e PBFT [Castro and Liskov 2002]. Estes algoritmos dão garantias de consistência forte, mas são caros e não escalam bem [Dolev and Lenzen 2013]. Tais blockchains são em geral baseadas em um líder, que propõe o bloco a ser armazenado e também faz a gerência dos processos membros.

Sistemas amplos não permissionados podem incluir uma quantidade expressiva de nós, mas apresentam uma vazão de transações bem limitada em comparação a sistemas de menor escala baseados em um grupo definido e algoritmos de consenso convencionais. Contudo, em redes permissionadas, o custo de mecanismos de difusão pode ser quadrático com o número de participantes, sendo um obstáculo a escalabilidade [Vukolić 2016, Guerraoui et al. 2019].

Neste trabalho apresentamos vCubeChain, uma blockchain permissionada que tem por objetivo ser escalável. A vCubeChain é baseada na topologia distribuída hierárquica vCube [Duarte et al. 2014, Ruoso et al. 2014]. A topologia virtual é mantida através de um detector de falhas que forma um hipercubo quando todos os processos estão corretos. Na medida em que processos falham por parada, o vCube se reorganiza, mantendo diversas propriedades logarítmicas. A vCubeChain é permissionada, i.e. todos os processos são devidamente autenticados. Um líder é eleito utilizando uma estratégia autonômica de difusão confiável para disseminar blocos na rede. Cada bloco consiste de múltiplas transações. Em caso de falsas suspeitas, podem haver múltiplos líderes concorrentes simultaneamente, permitindo a ocorrência de *forks* temporários como discutido em [Nakamoto 2008]. No trabalho são apresentadas provas de que mesmo nesta situação a vCubeChain se mantém íntegra, sempre retornando a um estado consistente com a conciliação de divergências. A vCubeChain foi implementada através de simulação e comparada com Bitcoin e Ethereum. Resultados demonstram a escalabilidade da solução.

O restante deste trabalho está organizado da seguinte forma. Na Seção 2 é definido o modelo de sistema utilizado, incluindo também uma descrição resumida do vCube. Na Seção 3 a vCubeChain é descrita e especificada e são apresentadas provas de correção e terminação. A simulação e resultados são descritos a seguir, na Seção 4. Trabalhos relacionados são apresentados na Seção 5. Finalmente a conclusão vem na Seção 6.

2. Modelo do Sistema

Assume-se um sistema distribuído com um conjunto P de $n > 1$ processos $\{p_0, .., p_{n-1}\}$ que se comunicam por troca de mensagens. Os processos também são chamados de *nodos*. Cada processo se comunica diretamente com qualquer outro processo, ou seja, o sistema é totalmente conectado sendo representável por um grafo completo. O sistema assume falhas de colapso (*crash*) permanentes. Cada processo pode estar em um de dois estados: um processo *correto* é aquele que nunca falha; caso contrário, é um processo *falho*. O envio e o recebimento de mensagem são operações atômicas, mas as primitivas de difusão (*broadcast*) não são atômicas. Os canais de comunicação são perfeitos. Assim, as

mensagens trocadas entre os processos nunca são perdidas, corrompidas ou duplicadas.

Os processos formam uma topologia hierárquica virtual chamada vCube [Duarte et al. 2014]. A topologia virtual é um hipercubo se não houver processos falhos. Após uma falha, o vCube se recupera autonomicamente, mantendo várias propriedades logarítmicas, como número de mensagens e distância máxima entre os processos. Até $n - 1$ processos podem falhar. O vCube implementa um detector de falhas. O sistema é considerado parcialmente síncrono com um tempo de estabilização global (GST, *Global Stabilization Time*). Informalmente, o sistema inicialmente se comporta de forma assíncrona, mas a partir do GST passa a se comportar de forma síncrona, ou seja, há limites para o tempo de execução de tarefas e transmissão de mensagens [Dolev et al. 1987]. Com isso, o detector de falhas pode cometer erros, isto é, antes do GST, um processo correto, mas lento, pode ser considerado falho. Assim, o vCube classifica os processos como corretos ou suspeitos.

2.1. O vCube

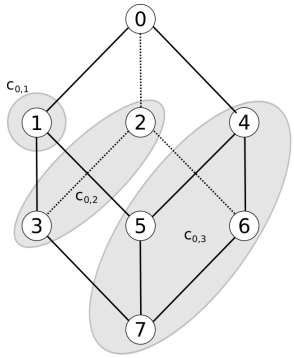
O vCube [Duarte et al. 2014] é uma topologia hierárquica virtual originalmente proposta como um algoritmo de diagnóstico distribuído [Duarte and Nanya 1998] mantida por um detector de falhas. Um processo executando o vCube pode testar outros processos para determinar se eles estão corretos ou se há suspeita de falha. O processo estará correto se o testador receber a resposta dentro do intervalo de tempo esperado. Caso contrário, o processo é suspeito. Os processos executam testes em *clusters* progressivamente maiores. Cada *cluster* $s = 1, .., \log_2 n$ possui 2^{s-1} elementos, onde n é o número total de processos no sistema. Os testes são realizados em rodadas. Em cada rodada, um processo i testa o primeiro processo correto j na lista de processos de cada *cluster* s. Caso o processo testado esteja correto, o testador obtém qualquer nova informação que o testador tenha sobre o estado de outros processos no sistema. Uma rodada é concluída após todos os processos corretos terem executado todos os testes atribuídos.

Os membros de cada *cluster* s e a ordem em que são testados por um processo i são dados pela função $c_{i,s}$, definida a seguir. O símbolo $\oplus$ representa a operação binária exclusiva OR (XOR):

$$c_{i,s} = \{i \oplus 2^{s-1}, c_{i\oplus 2^{s-1},1}, ..., c_{i\oplus 2^{s-1},s-1}\} \quad (1)$$

A Figura 1 ilustra a organização hierárquica em um hipercubo tridimensional com $n = 2^3$ processos. A tabela mostra os elementos de cada *cluster* $c_{i,s}$ para o sistema. Nesse sistema, cada processo testa três clusters. Como exemplo, o primeiro *cluster* testado por p_0 é $c_{0,1} = (1)$; os outros dois *clusters* são $c_{0,2} = (2, 3)$ e $c_{0,3} = (4, 5, 6, 7)$. Em cada rodada de testes, cada processo é testado pelo menos uma vez por um processo correto. Isso garante que, em no máximo $\log_2^2 n$ rodadas, todos os processos tenham informações de estado atualizadas localmente sobre todos os outros processos.

Em Rodrigues et al. (2014) um algoritmo de difusão confiável é apresentado para o vCube. Esse algoritmo assume um sistema síncrono, e os processos formam uma árvore geradora mínima autonômica [Rodrigues et al. 2014b] que garante que difusão termina em tempo logarítmico. Um processo correto encaminha mensagens para o primeiro processo correto de cada *cluster*. Uma versão sistemas assíncronos foi proposta em Jeanneau



s	$c_{0,s}$	$c_{1,s}$	$c_{2,s}$	$c_{3,s}$	$c_{4,s}$	$c_{5,s}$	$c_{6,s}$	$c_{7,s}$
1	1	0	3	2	5	4	7	6
2	2 3	3 2	0 1	1 0	6 7	7 6	4 5	5 4
3	4 5 6 7	5 4 7 6	6 7 4 5	7 6 5 4	0 1 2 3	1 0 3 2	2 3 0 1	3 2 1 0

Figura 1. Organização hierárquica com os *clusters* do processo 0 e a tabela completa para $c_{i,s}$ em um hipercubo de três dimensões.

et al. (2017), que considera possíveis falsas suspeitas e continua a enviar mensagens aos processos suspeitos.

3. vCubeChain: uma Blockchain Permissionada Escalável

Esta seção descreve a vCubeChain, uma blockchain permissionada escalável baseada em líder. Processos formam um vCube e utilizam o serviço subjacente de detecção de falhas para eleger um líder. O líder é sempre o processo correto de maior identificador. A vCubeChain assume processos autenticados (e.g., por meio de um mecanismo tradicional, como TLS – *Transport Layer Security* [Dierks and Rescorla 2008]). Desta forma, não há ataques de personificação nos quais um adversário finge ser o líder. Contudo, note-se que devido a falsas suspeitas e a latência para propagar a informação do estado de processos ao longo do sistema, é possível que, por algum período de tempo, mais de um dos processos considere-se o líder. Apesar disto, vCubeChain converge depois de um tempo finito para um único líder mantendo a consistência global.

O Algoritmo 1 apresenta o pseudo-código da solução. Usuários podem registrar transações em qualquer processo usando o procedimento NEWTRANSACTION. Se o processo não é o líder, envia a transação a este. O líder então valida a transação pelo procedimento PROCESSTRANSACTIONS e a inclui em um bloco candidato (procedimento ADDTOBLOCK). O líder constrói este bloco com um conjunto de transações válidas. O tamanho do bloco, i.e., o número máximo de transações em um bloco, é um parâmetro configurável. Uma vez completo o novo bloco (linha 24), o líder o dissemina na vCubeChain por meio de RBCAST (linha 26), um algoritmo de difusão confiável autonômica [Jeanneau et al. 2017]. Este algoritmo utiliza a topologia hierárquica do vCube para assegurar que a disseminação dos blocos para todo o sistema demandará um número de passos de comunicação logarítmico para se completar.

A vCubeChain adota a estrutura de dados comum a maioria das blockchains, como apresentado na Figura 2. Cada bloco, desde o primeiro - denominado gênese e proposto pelo primeiro líder – consiste em um conjunto de transações bem como do sumário criptográfico (*hash*) do bloco anterior e do próprio *hash* do bloco atual.

3.1. Proposição de Blocos

Novos blocos são propostos à vCubeChain como mostrado na Figura 3. Inicialmente, um cliente envia uma nova transação a qualquer processo da vCubeChain (passo 1). Se este

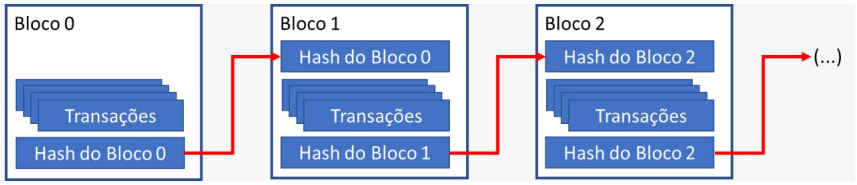


Figura 2. A estrutura de cadeia de blocos tradicional da vCubeChain.

processo não for líder, envia a transação ao mesmo (passo 2). O líder valida a transação e forma um novo bloco com um número suficientemente grande de transações válidas (passo 3). Finalmente, no passo 4, o novo bloco é disseminado ao longo da blockchain por meio do algoritmo de difusão confiável autonômica do vCube.

Contudo, devemos observar que antes do processo enviar a transação ao líder, este salva a transação em um *buffer* $pending\_transactions$ (linha 13). A transação só será removida quando o processo receber um bloco que a contém (linha 35) ou uma notificação do líder de que a mesma não é válida (de acordo com o estado do livro-razão). Se o líder atual se torna suspeito, o processo envia todas as transações deste *buffer* para o mais novo líder eleito. Se ambos líderes, suspeito e novo, persistirem transações em novos blocos, então um *fork* (bifurcação) ocorre. A vCubeChain detecta e resolve *forks* depois de um intervalo de tempo finito por meio do mecanismo descrito na subseção 3.3 para assegurar consistência global.

3.2. Eleição de Líder

Um processo aprende quem é o líder através da execução do procedimento CHECKLEADER. No caso de uma falha do líder, é eleito o o processo de maior identificador entre os processos corretos. É importante que os processos utilizem a primitiva CHECKLEADER adequadamente, para saberem se houve alguma mudança de líder. Quando um processo correto é incorretamente suspeito de ter falhado, ele pode voltar à liderança assim que o vCube contornar a suspeita incorreta. Como se assume o modelo GST, a partir de um certo instante de tempo o líder não será mais suspeito incorretamente.

No exemplo da Figura 4, em (1:) o processo 7 é o líder e acaba sendo indevidamente suspeito. Em seguida, o novo líder passa a ser o processo 6. Entretanto, logo em seguida, em (2:) a suspeita de falha do processo 7 é removida e durante um período de tempo os dois processos, 6 e 7, são percebidos como líderes por diferentes processos do sistema. Finalmente, em (3:) o sistema estabiliza e o processo 7 se torna líder único.

A estratégia proposta inclui uma abordagem baseada em consenso para eliminar incoerências resultantes da existência de múltiplos líderes, descrita a seguir.

3.3. Consenso

Dolev et al. (1987) provaram que o consenso pode ser reduzido à difusão confiável (*reliable broadcast*) com ordenação de mensagens. O algoritmo de consenso da vCubeChain é baseado naquele resultado. Em um cenário ideal, o líder recebe transações dos demais processos do sistema. O líder estabelece uma ordem local sobre as transações, após verificar sua validade, tendo em vista a cadeia de blocos já construída até o momento. Com a validade garantida, o líder propõe um bloco contendo as transações. Este bloco é disseminado utilizando o mecanismo de difusão confiável do vCube, que garante sua entrega

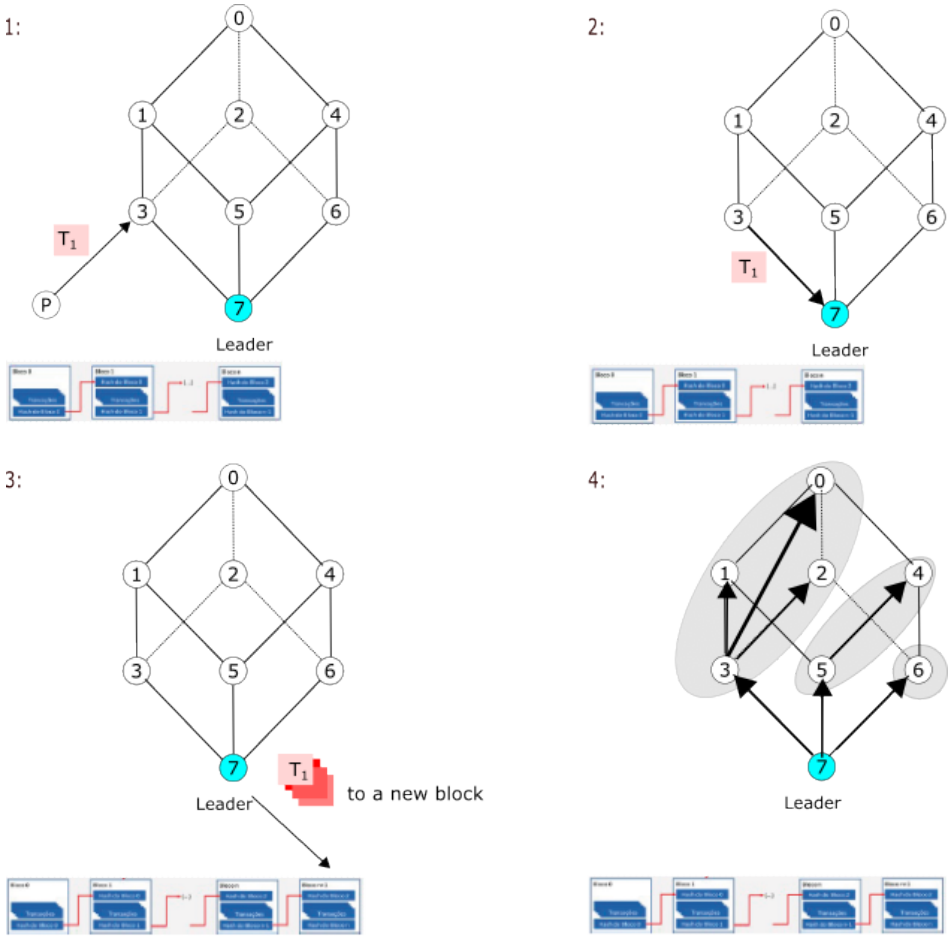


Figura 3. Proposição de blocos.

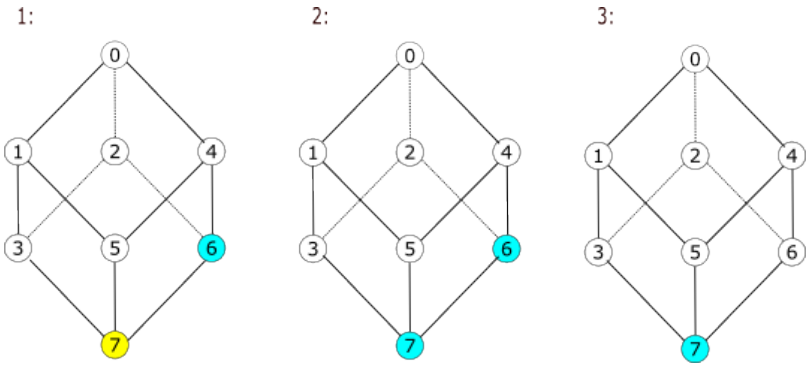


Figura 4. Exemplo de demoção indevida de liderança seguida de sua correção.

por todos os processos corretos após um intervalo de tempo finito usando uma árvore de difusão (spanning tree).

Supondo que, diante de instabilidades ou possível assimetria de informações, dois processos p e q se tomem líderes, ambos propõem blocos da cadeia conhecida. A vCubeChain, então, permite o *fork* da blockchain, ou seja, haverá dois blocos de sub-cadeias que podem ser discordantes entre si. Graças à difusão confiável, todos os processos cor-

retos terão a mesma visão da blockchain, incluindo o *fork* e subcadeias derivadas. Nesse cenário, as transações são propostas com base nas informações da subcadeia mais longa. Se o líder recebe transações derivadas do último bloco que ele não conhece, ele não o incorpora em uma proposição de bloco até que tenha conhecimento disso.

A formação de subcadeias a partir de um *fork* requer um mecanismo de ajuste de cadeia. No caso da vCubeChain, a difusão confiável envolve confirmações de recebimento do bloco pelos processos corretos. Dessa forma, o líder pode determinar quais blocos propostos são estáveis, ou seja, foram recebidos pelo conjunto de processos percebidos como corretos. Assume-se uma janela de estabilidade de B blocos. Ou seja, se um processo p permanecer líder e tiver pelo menos B blocos confirmados a frente de outra cadeia, sua cadeia pode ser considerada estável. Assim, as transações da cadeia que não persiste deverão ser reenviadas.

Essa estratégia de consenso é uma solução de compromisso baseada na premissa de que, após eventuais disputas entre líderes, a percepção dos processos corretos irá convergir. Assim, nessa convergência, o líder poderá resolver esses *forks* após um número B de blocos confirmados, dissolvendo quaisquer divergências com outro líder.

Por exemplo, na Figura 5, houve um *fork* devido aos líderes 6 e 7 conflitantes. Ou seja: o líder 7 propõe os blocos 2 a 5, mas, devido a uma falsa suspeita, o processo 6 assume-se como líder propondo o bloco $2'$. Após algum tempo, o processo 6 percebe que o processo 7 ainda está ativo e desiste da liderança. Assim, após algum período de estabilidade, o processo 7 se torna líder único. Os ACKs recebidos relacionados ao bloco quatro transmitem todos os blocos estáveis (os verdes na Figura 5). Ele também percebe quanto tempo é a diferença entre as subcadeias estáveis e usa esse comprimento ($B = 2$) para reconciliar as transações no bloco bifurcado $2'$ ou aguardar um número mais significativo de blocos estáveis. Em um intervalo de tempo finito, o bloco $2'$ será invalidado e todas as suas transações reconciliadas em um novo bloco (ou o proponente saberá que uma transação é incompatível com o estado atual do livro-razão).

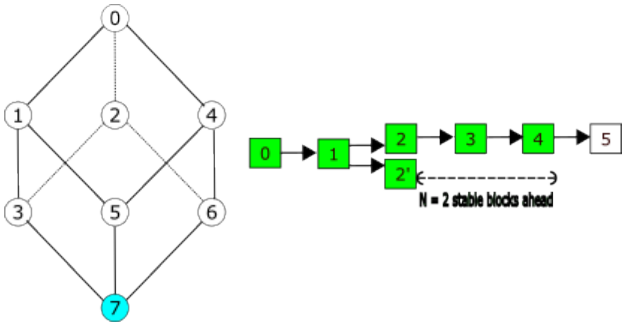


Figura 5. Blocos estáveis na visualização do líder.

Observe que o mecanismo de difusão vCube tem custo logarítmico em número de mensagens, mas em cenários de degradação antes do GST, esse custo aumenta até o quadrado do número de processos (pior caso em que todos os processos estão corretos e suspeitam indevidamente de todos os demais). O mecanismo de escolha do líder baseia-se na integração do detector de falhas vCube com uma regra determinística do algoritmo clássico de eleição de líder: nossa premissa é que, ao final, a percepção do detector de falhas irá convergir, indicando a todos os processos o mesmo líder.

Em cenários sem falhas de líder, o algoritmo é direto: a difusão de blocos em ordem pelo líder utilizando difusão confiável garante a implementação da blockchain da maneira usual. O cenário de falha do líder pode implicar, diante da instabilidade da rede, disputa e atuação de mais de um líder durante tal janela. Os processos aceitarão todos os blocos propostos disputando os líderes na ordem estabelecida na blockchain, embora com *fork*. Esse *fork* será reconciliado posteriormente com o período estável, conforme descrito acima. Além disso, mecanismos para entrada e saída espontânea de processos do conjunto de permissões podem ser facilmente implementados usando o conceito de estabilidade de difusão de informações. Devido a limitações de espaço, não faremos esta discussão neste artigo.

Algorithm 1 vCubeChain no processo i

```
 1: procedure INIT( )
 2:   correct_i ← P
 3:   pending ← ∅
 4:   CHECKLEADER( )
 5:   if IAMLEADER( ) then
 6:     append genesis_block to chain
 7:     RBCAST(Msg-type=BLOCK,
            genesis_block())

 8: procedure NEWTRANSACTION(msg m)
 9:   Let T be the new transaction with m
10:   if IAMLEADER( ) then
11:     PROCESSTRANSACTION(T)
12:   else
13:     pending ← pending ∪ {T}
14:     SENDTOLEADER(T)

15: procedure PROCESSTRANSACTIONS(T)
16:   if VALIDATE(T) then
17:     ADDTOBLOCK(T)
18:   else
19:     NOTIFYSENDER(T)

20: procedure ADDTOBLOCK(T)
21:   if !THEREISCANDIDATEBLOCK( ) then
22:     create candidate_block
23:   append T to candidate_block
24:   if BLOCKCOMPLETED( ) then
25:     append candidate_block to chain
26:     RBCAST(Msg-type=BLOCK,
            new_block = candidate_block)

27: upon receive ⟨T⟩ from p
28:   if IAMLEADER( ) then
29:     PROCESSTRANSACTION(T)
30:   else
31:     SENDTOLEADER(T)

32: upon receive ⟨BLOCK, new_block⟩ from p
33:   append new_block to chain
34:   for all T ∈ new_block do
35:     pending ← pending ∖ {T}

36: upon receive ⟨LEADER, new_leader⟩
    from p
37:   if IAMLEADER( ) and new_leader > i then
38:     pending ← pending
                  ∪ candidate_block
39:     candidate_block ← ∅
40:     leader ← new_leader
41:     for all T ∈ pending do
42:       SENDTOLEADER(T)

43: procedure CHECKLEADER( )
44:   new_leader ← max(correct_i)
45:   if new_leader ≠ leader then
46:     leader ← new_leader
47:     if IAMLEADER( ) then
48:       RBCAST(Msg-type=LEADER, new_leader)
49:       for all ⟨T, m⟩ ∈ pending do
50:         PROCESSTRANSACTIONS(T)
51:         pending ← pending ∖ {T}
52:     else
53:       for all ⟨T, m⟩ ∈ pending do
54:         SENDTOLEADER(T)

55: upon notification crash(process j)
56:   correct_i ← correct_i ∖ {j}
57:   CHECKLEADER( )

58: upon notification up(process j)
59:   correct_i ← correct_i ∪ {j}
60:   CHECKLEADER( )
```

3.4. Segurança no Funcionamento e Terminação

Embora não seja possível garantir progresso (i.e., terminação) em todos os cenários, a segurança no funcionamento é sempre mantida, apesar de falhas pelo colapso de processos e de assincronia do ambiente. Por meio do vCube, um dos processos é eleito líder, este é o proponente de novos blocos para a blockchain. O progresso é de fato garantido somente se: o líder é único e este pode se comunicar em um tempo finito com os outros processos ativos. Contudo, a segurança no funcionamento sempre se mantém mesmo em períodos de instabilidade, ou seja: quando não há um líder designado ou há mais de um processo que assume tal papel.

O algoritmo de eleição de líder baseado no vCube define o líder de forma determinística. O protocolo de difusão confiável provê um mecanismo de difusão de informação eficiente em uma execução graciosa (melhor caso). Ou seja, em um período com limites temporais associados à difusão confiável, o vCube atua como um detector de defeitos da classe P, o qual permite estabelecer um único líder correto, reconhecido por todos processos ativos. Durante tal execução graciosa, outros processos podem falhar por *crash*. Exceto em cenários degradados, deve-se esperar que tais execuções graciosas sejam a norma.

Lema 1 (Corretude)**.** *Todos os processos corretos convergem em um limite de tempo finito ao mesmo estado da blockchain (incluindo todas as transações correlatas).*

Discussão. O algoritmo apresentado para o consenso na blockchain é reduzido para a difusão confiável do vCube. Em um cenário de execução graciosa, há somente um líder correto o qual propõe blocos que serão difundidos em um limite de tempo finito para todos os processos corretos. Caso haja uma instabilidade, a informação do líder atual pode não ser difundida a todos processos corretos ao mesmo tempo. Sem um líder, novos blocos não são gerados e difundidos. Na presença de mais de um líder, estes podem propor novos blocos baseado na cadeia atual. Todos estes novos blocos são difundidos e geram subcadeias devido ao *fork* da blockchain. O mecanismo de difusão garante que a termo o mesmo conjunto de blocos criados será difundido para todos processos, mas não dirime a formação destas subcadeias.

□

Lema 2 (Terminação)**.** *Todas as transações corretas serão confirmadas em um limite de tempo finito na blockchain.*

Discussão. Em execução graciosa, há único líder e limites temporais associados à difusão confiável, que garantem a terminação desta, e por consequente do algoritmo de consenso: único líder propõe uma única sequência de blocos implementada na blockchain em uma cadeia única.

Na existência de instabilidade, como discutimos, podem haver cenários de múltiplos líderes com subcadeias derivadas disto. A partir da premissa de que em um limite de tempo finito o sistema se reconfigura ao novo líder único, face a período de sincronia longo o suficiente após tal instabilidade, este novo líder utiliza a informação da estabilidade dos blocos da rede, de modo a identificar a estabilidade da subcadeia que escolhe manter em detrimento de outras que haviam sido difundidas. Por meio de novos blocos,

as transações destas subcadeias preteridas são refeitas e os blocos associados indicados como inválidos. □

Teorema 1. *O Algoritmo 1 implementa uma solução de blockchain permissionada que garante a corretude e que, na presença de um período de estabilidade suficientemente longo, garantindo também a terminação do algoritmo.*

Demonstração. A prova segue diretamente do Lema 1 (Corretude) e Lema 2 (Terminação). □

4. Resultados de Simulação

Esta seção apresenta uma comparação da vCubeChain com o Bitcon e o Ethereum usando o simulador Blocksim [Faria and Correia 2019]. O Blocksim é um simulador de eventos discretos desenvolvido em Python que fornece modelos para avaliação de blockchains, atualmente com implementações Bitcoin e Ethereum. A rede P2P do simulador é configurada a partir de nós (representando processos) que se comunicam com base em parâmetros de latência de transmissão e atrasos de envio e recebimento definidos em arquivos de configuração no formato JSON (*JavaScript Object Notation*). Os nós são inicialmente conectados por meio de uma lista de nós vizinhos a partir dos quais a comunicação é monitorada. A seguir, é apresentado um descritivo dos modelos implementados no Blocksim.

Bitcoin. Bitcoin [Nakamoto 2008] foi implementado no BlockSim usando um limite de tamanho de bloco (1MB) e uma distribuição de probabilidade para o número de transações por bloco (usando dados reais da rede Bitcoin). O modelo prevê dois tipos de nós: mineradores e não-mineradores. Um nó não minerador apenas valida blocos ou valida e publica novas transações. Os mineradores validam e agrupam novas transações em uma fila de transações para criar blocos candidatos (processo de mineração) que posteriormente são revelados a outros nós do sistema.

Ethereum. Ethereum [Wood 2014] foi implementado de forma semelhante ao Bitcoin com nós mineradores e não-mineradores. No entanto, o Ethereum implementa o *gas limit*, que é o valor máximo de *gas* que uma transação pode gastar (entre 1 e 32.000, normalmente 21.000[1]) e o limite de *gas* para um bloco. Por exemplo, se o bloco tiver um limite de 10.000 *gas* e cada transação tiver um limite de 1.000, cada bloco poderá conter até 10 transações.

vCubeChain. A vCubeChain usa a mesma configuração do Bitcoin. No entanto, as transações são enviadas apenas para o líder (o processo com o ID mais alto), e o líder transmite os blocos para todos os outros processos usando o serviço de transmissão hierárquica vCube.

As implementações para Bitcoin e Ethereum consideram uma rede totalmente conectada na qual todos os nós se comunicam diretamente uns com os outros. A implementação da vCubeChain proposta neste trabalho utiliza a topologia virtual vCube, na qual os nós são conectados com vizinhos definidos pelas regras da topologia. Para cada mensagem enviada e recebida na rede, é calculado um tempo de processamento dependendo do tamanho da mensagem. O processo de conexão, que ocorre apenas no primeiro contato entre dois nós, simula o *handshake* do TCP com uma tripla latência.

[1] `https://ethereum.github.io/yellowpaper/paper.pdf`

4.1. Configuração dos Cenários

As soluções foram testadas com $N = 8$ e $N = 64$ nós (representando processos). Os parâmetros de configuração utilizados foram os definidos em [Faria and Correia 2019]. Para Ethereum, foi usado o valor padrão de *gas limit* de 21.000 e *block gas limit* de 2,1 milhões. Os nós foram distribuídos em três sites de acordo com a Tabela 1. Para $N = 8$, por exemplo, há dois nós em Cascavel, mais dois nós em Salvador e quatro nós em Curitiba. Para este último, um nó é configurado como minerador ao executar Bitcoin ou Ethereum, ou como líder ao executar vCubeChain. O simulador não suporta a simulação de falhas. Portanto, a avaliação se concentra na comparação de escalabilidade, principalmente no tempo de execução e no número de mensagens.

Tabela 1. Locais onde os nós são alocados.

Sites	Não-mineradores	Minerador/Líder
Cascavel	$N/4$	0
Salvador	$N/4$	0
Curitiba	$(N/2) - 1$	1

A latência em segundos entre os nós em cada site foi configurada usando os parâmetros da Figura 6. A distribuição normal (média e desvio padrão) foi usada para nós em locais diferentes e a distribuição Gama inversa foi usada para nós no mesmo local. Os valores foram calculados com base nos dados de RTT reais (*ping*) obtidos na rede da RNP (Rede Nacional de Pesquisa). Os parâmetros completos estão disponíveis no Github.

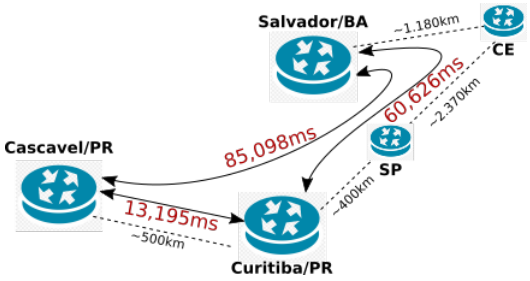
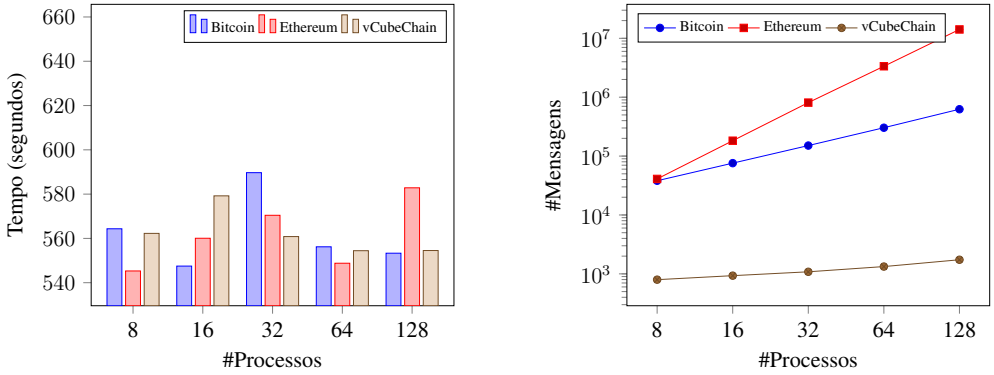


Figura 6. Latência em milissegundos entre nós em cada site.

As simulações foram executadas usando um computador Dell Inc. XPS 8950 com CPU 12th Gen Intel® Core™ i7-12700 × 20, 16 GB de RAM e 512 GB SDD, sistema operacional Ubuntu 22.04.1 LTS de 64 bits. Para cada execução, foram simuladas mil transações, divididas em mil rodadas de uma transação a cada 15 segundos. As transações foram distribuídas aleatoriamente entre os processos usando a função *transaction factory* do simulador.

4.2. Resultados

O tempo médio necessário para validar todas as transações é mostrado na Figura 7(a), calculado a partir de 30 execuções em cada cenário. Os tempos para validação das transações são semelhantes para os três algoritmos simulados. No entanto, o tempo de execução da simulação do Ethereum foi significativamente maior (várias horas), comparado com Bitcoin e Ethereum (poucos minutos).



(a) Tempo de execução

(b) Número de Mensagens

Figura 7. Tempo médio para validar todas as transações (esq.) e número médio de mensagens em escala logarítmica (dir.)

O número total de mensagens enviadas é mostrado na Figura 7(b), representada em escala logarítmica para facilitar a visualização. No modelo implementado pelo simulador, tanto o Bitcoin quanto o Ethereum utilizam uma estratégia de disseminação para o cabeçalho e o texto da mensagem. Ou seja, para cada mensagem contendo o cabeçalho da transação, é enviada uma solicitação do corpo da transação, que gera uma nova mensagem com o conteúdo da transação. O mesmo vale para os blocos. Para fins de comparação, a VCubeChain manteve esse comportamento. No entanto, o número de mensagens é muito menor devido à árvore de difusão criada pelo líder. No caso de Bitcoin e Ethereum, a transmissão é de um-para-todos. Além disso, os processos no Bitcoin enviam mensagens de inicialização no início da comunicação para difundir seu conhecimento sobre novas transações e blocos.

Quanto ao número de blocos criados, no Ethereum, o número de transações por bloco é determinado pelo "limite de gas". Para o Bitcoin e a VCubeChain foi utilizado o tamanho do bloco físico (2MB). Com isso, todas as transações foram agrupadas em um único bloco, em todos os cenários.

Observando as conexões TCP em uma execução sem falhas, vCubeChain abre menos conexões em comparação com Bitcoin e Ethereum porque só se comunica com o líder e até $\log_2(N)$ vizinhos na topologia vCube, enquanto as outras utilizam uma estratégia um-para-todos, ou seja, $(N^2 - N)/2$ conexões.

5. Trabalhos Relacionados

Blockchains não-permissionadas utilizam abordagens probabilísticas para o consenso, tais como a prova-de-trabalho da Bitcoin [Nakamoto 2008]. Estas abordagens consistem implicitamente em eleição de líder. Múltiplos processos competem para resolver o desafio criptográfico para então apresentar uma proposta de bloco para a blockchain. Múltiplos processos podem ter sucesso, i.e. resultando em uma divergência expressa por um *fork*. Nestes casos, a regra utilizada é a da sobrevivência da cadeia mais longa. Esta regra cria incentivos para que novos proponentes minerem naquela cadeia mais longa de forma a levar o consenso a convergência.

Abordagens tradicionais para o consenso, como o Paxos [Lamport 1998] e Raft

[Ongaro and Ousterhout 2014], utilizam explicitamente eleição de líder. No Paxos, por exemplo, a fase inicial é utilizada para um proponente tentar se tornar líder e efetivar sua proposta. Alternativamente, o proponente pode não conseguir maioria e tentar mais tarde, ou outro líder pode ser eleito. O Raft, por outro lado, distingue com clareza o processo de eleição do líder, que é executado de forma explícita pelo protocolo.

Tanto a estratégia do Paxos (baseada em maioria) como a estratégia do Raft (baseada em troca de mensagens de *heartbeat*) podem ter desempenho impactado quando o número de processos cresce. Além disso, apesar de terem a vantagem da previsibilidade, estas estratégias determinísticas têm custo de comunicação mais elevado. Já os algoritmos probabilísticos, apesar de mais eficientes em termos de comunicação, podem apresentar outros custos e desafios, como a execução da prova-de-trabalho.

A abordagem proposta no presente trabalho utiliza o vCube tanto como detector de falhas, como mecanismo para eleição de líder. A estratégia evita contenção: o líder é o processo sem falha de maior identificador. Quando o sistema se comporta como assíncrono, instabilidades subjacentes na rede podem levar à eleição de múltiplos líderes. Neste caso, assumimos o comportamento como na Bitcoin e outras abordagens não permissionadas, permitindo divergências (*forks*) no livro-razão, as quais são dirimidas a posteriori. Esta solução de compromisso busca reduzir a contenção no ambiente permissionado – como ocorre nos algoritmos de consenso geralmente usados neste ambiente. Desta forma, o resultado é um aumento da escalabilidade da solução.

Há abordagens como o FastPaxos [Boichat et al. 2003] que assumem uma estratégia especulativa, eficiente em execuções graciosas. Na nossa abordagem, permitimos uma maior escalabilidade no cenário degradado, uma vez que a difusão confiável permite a continuidade de novos blocos na blockchain, postergando decisões quanto a blocos em face a derivações para quando o sistema estiver em um período de estabilidade.

6. Considerações Finais

Neste artigo, foi apresentada uma proposta de blockchain permissionada baseada na detecção de falhas e no mecanismo de transmissão confiável baseados no vCube. Um líder selecionado pelo algoritmo de eleição associado a este detector de falhas realiza propostas de novos blocos. Graças às propriedades logarítmicas do vCube, o consenso é bastante eficiente em execuções sem erros. A abordagem favorece vivacidade e escalabilidade, mesmo em períodos de instabilidade do líder, assumindo convergência posterior quando ocorrem melhores condições de rede.

Entre os trabalhos futuros está a extensão do simulador para permitir a avaliação de cenários com falhas, a comparação com blockchains permissionadas e a implementação da vCubeChain como software livre.

Referências

Androulaki, E. et al. (2018). Hyperledger fabric: A distributed operating system for permissioned blockchains. In *Proc. 13th EuroSys Conference*, EuroSys '18. ACM.

Boichat, R., Dutta, P., Frølund, S., and Guerraoui, R. (2003). Deconstructing paxos. *ACM Sigact News*, 34(1):47–67.

Castro, M. and Liskov, B. (2002). Practical byzantine fault tolerance and proactive recovery. *ACM Trans. Comput. Syst.*, 20(4):398–461.

Chen, W. et al. (2018). A survey of blockchain applications in different domains. In *Proc. of the 2018 Int. Conf. on Blockchain Technology and Application*, ICBTA 2018. ACM.

Dierks, T. and Rescorla, E. (2008). The Transport Layer Security (TLS) Protocol Version 1.2. RFC 5246 (Proposed Standard). Updated by RFCs 5746, 5878, 6176.

Dolev, D., Dwork, C., and Stockmeyer, L. (1987). On the minimal synchronism needed for distributed consensus. *J. ACM*, 34(1):77–97.

Dolev, D. and Lenzen, C. (2013). Early-deciding consensus is expensive. In *Proc. of the PODC '13*, page 270–279, New York, NY, USA. ACM.

Duarte, E., Bona, L., and Ruoso, V. (2014). VCube: A provably scalable distributed diagnosis algorithm. In *5th ScalA Workshop*, pages 17–22.

Duarte, E. and Nanya, T. (1998). A hierarchical adaptive distributed system-level diagnosis algorithm. *IEEE Transactions on Computers*, 47(1):34–45.

Ethereum.org (2022). Proof-of-stake (pos). `https://ethereum.org/en/developers/docs/consensus-mechanisms/pos/`.

Faria, C. and Correia, M. (2019). Blocksim: Blockchain simulator. In *2019 IEEE International Conference on Blockchain (Blockchain)*, pages 439–446.

Gamage, H. T. M., Weerasinghe, H., and Dias, N. G. J. (2020). A survey on blockchain technology concepts, applications, and issues. *SN Computer Science*, 1:1–15.

Guerraoui, R., Hamza, J., Seredinschi, D.-A., and Vukolic, M. (2019). Can 100 machines agree? https://arxiv.org/abs/1911.07966.

Jeanneau, D. et al. (2017). An autonomic hierarchical reliable broadcast protocol for asynchronous distributed systems with failure detection. *JBCS*, 23(15).

Lamport, L. (1998). The part-time parliament. *ACM Transactions on Computer Systems*, 16(2):133–169.

Nakamoto, S. (2008). Bitcoin: A peer-to-peer electronic cash system.

Ongaro, D. and Ousterhout, J. (2014). In search of an understandable consensus algorithm. In *Proc. of the 2014 USENIX Conference*, USENIX ATC'14, USA. USENIX.

Rodrigues, L. A., Arantes, L., and Duarte Jr., E. P. (2014a). An autonomic implementation of reliable broadcast based on dynamic spanning trees. In *2014 Tenth European Dependable Computing Conference*.

Rodrigues, L. A., Duarte Jr, E. P., and Arantes, L. (2014b). Árvores geradoras mínimas distribuídas e autonômicas. *SBRC*, pages 1–14.

Ruoso, V., Bona, L., and Duarte Jr, E. P. (2014). Uma estratégia de testes logarítmica para o algoritmo hi-adsd. In *Workshop de Testes e Tolerância a Falhas*, pages 31–44. SBC.

Vukolić, M. (2016). The quest for scalable blockchain fabric: Proof-of-work vs. bft replication. In *Open Problems in Network Security*. Springer Int. Publishing.

Wood, D. D. (2014). Ethereum: A secure decentralised generalised transaction ledger. `https://ethereum.github.io/yellowpaper/paper.pdf`.