A Caminho do Broadcast Hierárquico Tolerante a Falhas Bizantinas

Gabriela Stein[1,2], Luiz A. Rodrigues[1] e Elias P. Duarte Jr.[2]

[1]Universidade Estadual do Oeste do Paraná (UNIOESTE)
Cascavel, PR

[2]Universidade Federal do Paraná (UFPR), Departamento de Informática
Curitiba, PR

{gabriela.stein,luiz.rodrigues}@unioeste.br, elias@inf.ufpr.br

Abstract. *Byzantine fault-tolerant reliable broadcast is a fundamental problem in distributed systems, ensuring consistent message delivery even when processes behave arbitrarily. This article presents a strategy for reliable Byzantine dissemination based on the VCube topology, combined with digital signatures to guarantee message integrity and authenticity. The proposed algorithm tolerates up to $f = (n-1)/3$ Byzantine processes and uses dynamic trees to ensure message propagation throughout the system. Message delivery is determined by validating the accumulated signatures along the dissemination path.*

Resumo. *O broadcast confiável tolerante a falhas bizantinas é um problema central em sistemas distribuídos, garantindo a entrega de mensagens mesmo havendo processos com comportamento arbitrário na rede. Este trabalho apresenta uma estratégia de difusão bizantina confiável baseada na topologia hierárquica do VCube, combinada com o uso de assinaturas digitais para assegurar integridade e autenticidade das mensagens. A decisão de entrega do algoritmo é baseada na validação das assinaturas acumuladas ao longo da difusão. O algoritmo proposto tolera até $f = (n-1)/3$ processos bizantinos e emprega árvores dinâmicas construídas sobre o VCube para garantir a propagação de mensagens no sistema.*

1. Introdução

O *broadcast* confiável tolerante a falhas bizantinas é um problema fundamental em sistemas distribuídos, com diversas soluções existentes [Lamport et al. 1982, Bracha 1987, Amoussou-Guenou et al. 2026]. Sua principal garantia é que todas as mensagens serão entregues mesmo diante de comportamento arbitrário (bizantino) dentro do sistema [Correia et al. 2006]. Esse modelo de falhas engloba comportamento malicioso fora da especificação como a falsificação de mensagens, comunicação inconsistente além do comportamento presente nos modelos de falhas clássicos: falha *crash* (parada) onde o nó deixa de responder a estímulos externos e falha por omissão, no qual o nó não entrega todas as mensagens que deveria.

Este trabalho apresenta um esforço inicial de especificar uma estratégia hierárquica de difusão confiável em um sistema sujeito a falhas bizantinas. O algoritmo faz uso tanto da topologia hierárquica do VCube [Duarte Jr et al. 2023] quanto de um

sistema de assinaturas utilizando criptografia de chaves públicas, de modo a garantir, principalmente, a integridade e a autenticidade das mensagens sendo trocadas no sistema. Além de tolerar até $f = (n-1)/3$ processos falhos.

O restante do texto está organizado nas seguintes seções. A Seção 2 apresenta a topologia virtual VCube. A Seção 3 discute os trabalhos relacionados a difusão confiável em sistemas tolerantes a falhas bizantinas, ressaltando diferenciais de nossa proposta. A estratégia de difusão confiável bizantina é descrita na Seção 4. A Seção 5 conclui o trabalho e apresenta os trabalhos futuros.

2. VCube

O VCube é uma topologia hierárquica, que equivale a um hipercubo quando todos os processos estão corretos e o número de processos é uma potência de dois. O VCube inclui um detector de falhas crash [Chandra and Toueg 1996, Turchetti et al. 2016]. Quando processos falham, o VCube se reorganiza autonomicamente, mantendo diversas propriedades logarítmicas. A topologia é construída e atualizada a partir de informações de diagnóstico coletadas por um sistema de monitoramento de processos. Cada processo participante é responsável por testar os outros processos, verificando periodicamente se estão operando corretamente ou apresentam falhas [De Bona and Duarte Jr 2004]. Um processo é considerado correto quando responde adequadamente dentro de um tempo esperado, caso contrário, é considerado como um processo falho.

Os processos são organizados em clusters de tamanho progressivamente maior. Os testes ocorrem em rodadas: em cada rodada, um processo i testa o processo j quando i é o primeiro processo correto de algum cluster de j. A partir desse processo j, i obtém também informações atualizadas sobre o estado dos demais processos do sistema.

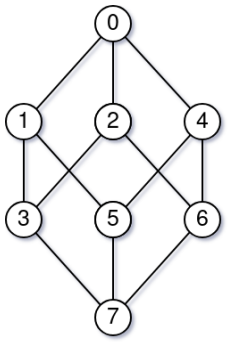

Figura 1. Organização lógica hierárquica de 8 processos.

O VCube foi utilizado para construção de diversos algoritmos distribuídos hierárquicos. [Rodrigues et al. 2017, Jeanneau et al. 2017] utilizaram a estrutura do VCube para construir *broadcast* confiável em um sistema sujeito a falhas *crash*. Em [Ruchel et al. 2024] é introduzido um algoritmo de difusão atômica. Aqueles algoritmos são baseados nas árvores autonômicas geradas pelo VCube [Rodrigues et al. 2025]. Outros algoritmos e aplicações distribuídos baseados no VCube incluem [de Araujo et al. 2019], em que o VCube foi utilizado como topologia para construir um sistema *publish/subscribe*. Um sistema de gerência de falhas de redes locais baseado no VCube foi proposto em [Duarte and De Bona 2002]. O VCube já foi utilizado também como base para a construção de redes *overlay* [Bona et al. 2006]. Uma

solução distribuída para exclusão mútua distribuída baseada no VCube é apresentada em [Rodrigues et al. 2018]. Um sistema para geração de quorums autonômicos baseados no VCube já foi proposto [Rodrigues et al. 2016]. Em [Stein et al. 2023] um detector de falhas $\Diamond P$ para sistemas assíncronos é descrito a partir do VCube, considerando apenas falhas *crash*. A vCubeChain é uma blockchain permissionada baseada no VCube [Freitas et al. 2024]. O presente trabalho se baseia nesse resultado e utiliza as árvores de difusão de mensagens descritas em [Rodrigues et al. 2014].

Uma versão do VCube foi elaborada para o diagnóstico baseado em comparações [Albini et al. 2005]. Neste tipo de diagnóstico, o testador faz uma comparação de valores, que devem ser idênticos entre os processos[Duarte Jr et al. 2011, Ziwich and Duarte 2016]. Acreditamos que esta pode ser uma base para a construção de soluções distribuídas baseadas no VCube tolerantes a falhas bizantinas.

3. Trabalhos Relacionados

Nessa seção são apresentados os principais resultados da literatura referentes a soluções de *broadcast* em sistemas tolerantes a falhas bizantinas. O modelo bizantino traz fortes desafios, em especial devido à sua característica de abranger *qualquer* comportamento arbitrário - incluindo as falhas por parada e falhas de omissão de mensagens, além de comportamentos maliciosos de falsificação e corrompimento (um processo falsifica mensagens que não recebeu ou altera mensagens no processo de retransmissão) ou qualquer comportamento inesperado arbitrário que esteja fora da especificação do sistema. Ainda mais desafiadora é a situação em que o processo atingido funciona como um adversário no sistema.

O trabalho seminal de [Lamport et al. 1982] consolida o problema do consenso em cenários de falhas bizantinas e demonstra que alcançar consenso em sistemas distribuídos com a presença de f falhas bizantinas é possível apenas se o número de participantes for no mínimo $3f + 1$, além disso, são apresentados algoritmos que permitem que processos corretos concordem mesmo na presença de participantes maliciosos.

[Dolev and Strong 1983] propuseram algoritmos para alcançar consenso bizantino em sistemas distribuídos utilizando mensagens autenticadas. Os autores demonstram que, ao empregar mecanismos de autenticação como assinaturas digitais, é possível evitar a falsificação de mensagens por processos maliciosos e simplificar o processo de consenso. O algoritmo apresentado alcança acordo em $f + 1$ rodadas, onde f representa o número máximo de processos bizantinos.

Um trabalho de referência para esta discussão é o de [Bracha 1987] que introduz um algoritmo de *broadcast* confiável bizantino. A solução apresentada necessita de três fases de troca de mensagens antes de fazer a entrega final da mensagem. Importante ressaltar que o autor utilizou o *broadcast* confiável bizantino como estratégia para solucionar o consenso em sistemas assíncronos sujeitos a falhas bizantinas.

Em [Dolev 1981], estende-se o problema de consenso bizantino para redes arbitrárias, mostrando que a possibilidade de acordo depende da conectividade da rede. Ao provar que para tolerar até f falhas bizantinas, a rede deve ter conectividade de pelo menos $2f + 1$, os autores estabeleceram que a estrutura da rede é fundamental para garantir consenso em ambientes com falhas bizantinas.

O artigo [Cachin et al. 2001] apresenta um protocolo de *broadcast* atômico utilizando primitivas de criptografia para a comunicação, trazendo os aspectos iniciais de uma integração entre tolerância a falhas bizantinas e protocolos de *broadcast* assíncronos.

Trabalhos mais recentes começam a tratar outros aspectos do *broadcast* confiável bizantino, como o artigo de [Bonomi et al. 2019], que propõe um protocolo de *broadcast* confiável bizantino para redes *multi-hop*, onde nem todos os processos estão diretamente conectados. A partir da suposição de um emissor de mensagens honesto, os autores simplificaram o problema e desenvolveram uma solução que garante propriedades como acordo e integridade mesmo na presença de processos maliciosos.

[Kozhaya et al. 2018] propuseram um algoritmo de difusão confiável com garantias de tempo real, mesmo na presença de falhas bizantinas e perdas de mensagens. O trabalho mostra que soluções tradicionais não conseguem oferecer essas garantias e introduz um mecanismo baseado em agregação de assinaturas, além de permitir que os processos se desliguem preventivamente quando detectam algum comportamento inconsistente, assegurando que mensagens de processos corretos sejam entregues em um prazo definido.

[Locher 2024] apresenta algoritmos para atingir *broadcast* confiável em sistemas distribuídos que precisam tolerar falhas bizantinas. Os algoritmos propostos se baseiam em agregação criptográfica e disseminação otimizada de mensagem para agilizar as fases de verificação e entrega, além de aprimorar a eficiência tanto em comunicação quanto em latência. O trabalho é capaz de reduzir o *overhead* dos algoritmos clássicos de 3x o tamanho de mensagem para até 1.5x em casos ideais.

Nossa proposta utiliza assinaturas agregadas para garantir a integridade e autenticidade das mensagens trocadas. Neste sentido, até onde foi possível verificar, não há uma abordagem similar quando se trata da disseminação de mensagens no *broadcast* confiável bizantino utilizando uma estrutura hierárquica.

4. A Caminho do Broadcast Bizantino Hierárquico

A difusão confiável de mensagens em um sistema tolerante a falhas bizantinas precisa garantir que as mensagens sejam entregues por todos os processos de modo a assegurar consistência e confiabilidade. A estratégia apresentada nesse artigo segue os mesmos pressupostos. As propriedades necessárias são [Cachin et al. 2011]:

- Integridade: cada processo entrega a mensagem m no máximo uma vez, e somente se m foi previamente enviada por algum processo;
- Validade: se um processo correto p envia um mensagem m, então todo processo correto, após um tempo finito, entrega m;
- Acordo: se algum processo correto entrega a mensagem m, todo processo correto também entrega m após algum tempo finito;
- Terminação: Todo processo correto entrega algum valor.

Estas propriedades serão utilizadas como base para demonstrar a corretude do protocolo proposto na Seção 4.4.

4.1. Modelo do Sistema

Considera-se um sistema distribuído composto por um conjunto finito de processos $P = \{p_0, p_1, \ldots, p_{n-1}\}$, onde n representa o número total de processos. Cada processo possui

um identificador único e se comunica com os demais por meio de troca de mensagens.

O modelo assume um sistema assíncrono, no qual não há limites conhecidos para o tempo de execução dos processos, nem para o atraso na transmissão das mensagens. Entretanto, considera-se que os canais são confiáveis no sentido de que não criam, não corrompem e não duplicam mensagens, embora possam atrasá-las arbitrariamente.

Adota-se um modelo de falhas bizantinas, no qual até f processos podem apresentar comportamento arbitrário. Processos bizantinos podem omitir mensagens, enviar mensagens incorretas ou agir de forma maliciosa e coordenada. Para garantir tolerância a falhas bizantinas, assume-se que o número total de processos $n \geq 3f + 1$.

Além disso, assume-se que os processos possuem mecanismos de autenticação de mensagens, como assinaturas digitais. Esse mecanismo garante a integridade e autenticidade das mensagens, impedindo que processos bizantinos falsifiquem a identidade de processos corretos.

Por fim, assume-se que todo processo correto executa o algoritmo e permanece ativo durante toda a execução do sistema.

4.2. Difusão no VCube

Nesse trabalho, como maneira de otimizar as trocas de mensagens requeridas pelo algoritmo, utilizou-se do trabalho de [Jeanneau et al. 2017] para realizar a difusão confiável entre os processos, considerando a estrutura de processos fornecida pelo VCube. O funcionamento do *broadcast* ocorre de forma hierárquica e recursiva: quando um processo difunde uma mensagem, ele não a envia diretamente a todos os processos (*one-to-all*). Em vez disso, ele encaminha a mensagem para um subconjunto de processos vizinhos definidos pela estrutura do VCube, e cada um desses processos repassa a mensagem para outros subconjuntos, formando uma disseminação em árvore binomial.

A Figura 2 demonstra as árvores de disseminação dos processos 0 (a), 1 (b) e 2 (c) separadas e, em conjunto em único sistema, é possível observar que existe sobreposição das árvores (d), de modo a garantir múltiplos caminhos até um processo. As árvores dos outros processos no sistema foram omitidas de forma a contribuir para a visualização.

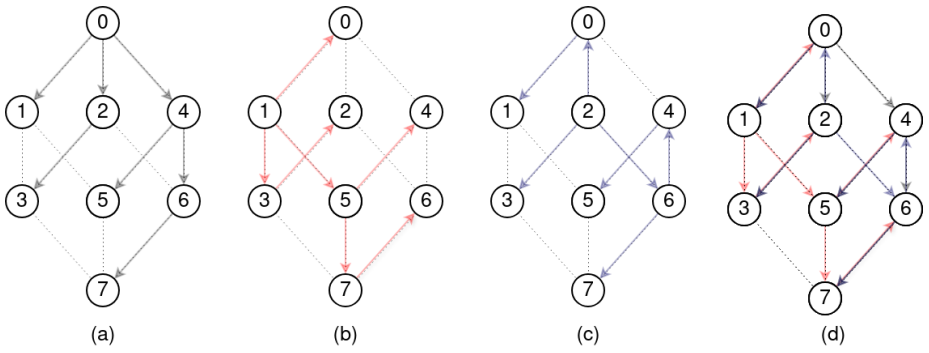

Figura 2. Árvores de disseminação de mensagens do processo 0 (a), 1 (b) e 2 (c), árvore com a sobreposição (d).

A primitiva VCUBEBROADCAST garante que mensagens enviadas por processos corretos são eventualmente entregues a todos os processos corretos. Isso acontece espe-

cialmente devido a estrutura de disseminação de árvores, onde cada processo ao receber a mensagem pela primeira vez inicia sua própria disseminação como "raiz" da árvore.

4.3. Algoritmo Proposto

Nessa seção é descrita a estratégia proposta de difusão confiável em um sistema sujeito a falhas bizantinas. O Algoritmo 1 apresenta um pseudocódigo referente a seu funcionamento. A ideia central é que uma mensagem só seja entregue quando houver evidência suficiente, em forma de assinaturas válidas, de que ela foi aceita e retransmitida por um número mínimo de processos.

No início, cada processo mantém três estruturas locais para identificar as mensagens com o id: $signed\_value[id]$ que registra o primeiro valor assinado por aquele processo; $seen[id]$ que armazena as versões já recebidas daquela mensagem junto com seus conjuntos de assinaturas e $delivered[id]$, que indica se a mensagem já foi entregue à aplicação. Inicialmente, nenhum valor foi assinado, o conjunto de mensagens recebidas é vazio, assim como nenhuma mensagem foi entregue.

Quando o emissor deseja enviar uma mensagem m, é executada a função BROADCAST(id, m) que cria a mensagem criptografada com o valor e sua assinatura. Depois, inicia a disseminação do *broadcast* disparando o procedimento VCUBEBROADCAST. Ao receber uma mensagem M o processo faz a verificação de todas as assinaturas acumuladas ao longo do caminho para a mensagem. O processo também verifica se já recebeu anteriormente exatamente essa mesma mensagem. Se a verificação de assinaturas estiver correta, M for uma mensagem nova ou uma mensagem já recebida, mas com assinaturas adicionais, o processo vai criar uma nova versão da mensagem com o conjunto de assinaturas atualizado e propagar utilizando a primitiva do VCUBEBROADCAST. A entrega da mensagem ocorre quando o conjunto de assinaturas obtiver tamanho $2f + 1$, garantindo que houve interseção suficiente entre os processos corretos para sustentar a confiabilidade da entrega.

O método utilizado para suprir a necessidade de identificar falsificação de mensagens dentro da tolerância a falhas bizantinas é um sistema de assinaturas digitais com chaves públicas. Por meio desse esquema de assinaturas, é possível manter a autenticidade e integridade das mensagens, de modo a mitigar determinados tipos de comportamentos arbitrários caracterizados como bizantinos, como a falsificação de conteúdos de mensagem ou a personificação de um processo pelo outro.

Cada processo correto assina no máximo um valor para um dado identificador id, caso receba alguma outra mensagem com o mesmo identificador já conhecido, não assina aquela mensagem novamente, guarda a mensagem e inicia o *broadcast* para a mensagem sem assinatura. Ou seja, o processo assina somente uma vez a mensagem com identificador único, mas guarda e repassa todas as mensagens duplicadas recebidas.

Para fins de simplificação da apresentação do algoritmo, assume-se que todas as mensagens tem seus respectivos resumos criptográficos assinados. Adicionalmente, considera-se que todos os processos da rede já têm as chaves públicas de todos os outros processos na rede. Além disso, cada processo, ao receber uma mensagem, realiza a validação tanto do conteúdo (verificando se não houve corrupção da mensagem) quanto das assinaturas associadas, permitindo mitigar ataques de falsificação.

Quanto à organização dos processos para fins de comunicação, a utilização do VCube aloca os processos em uma estrutura logarítmica hierárquica e a disseminação das mensagens ocorre por meio das árvores de difusão confiável geradas por cada processo. A Figura 3 apresenta o processo de difusão de mensagens para as árvores do processo 0 (preto) e subsequentemente do processo 1 (vermelho). Nela é possível observar o início da distribuição de mensagens entre os processos.

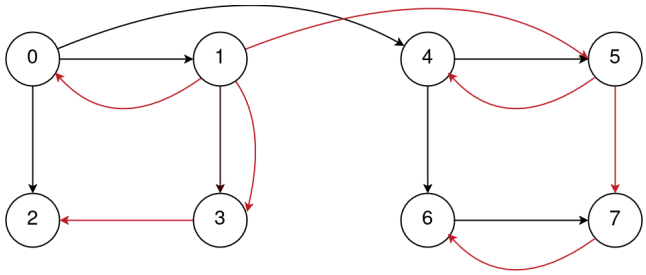

Figura 3. Árvores formadas pela troca de mensagens, ressaltando a árvore do processo 0 (preto) e árvore do processo 1 (vermelho).

O funcionamento de um caso normal de operação da difusão confiável em um sistema que não apresenta nenhuma falha bizantina é o seguinte:

1. O processo origem assina a mensagem e encaminha para os seus $\log_2(N)$ vizinhos no VCube;
2. Ao receber a mensagem, cada um desses processos vizinhos faz a verificação de assinaturas e, caso esteja correta, segue com o protocolo de difusão, ou seja, assina e encaminha a mensagem a seus $\log_2(N)$ vizinhos;
3. O *broadcast* finaliza em um processo quando recebeu pelo menos $2f + 1$ assinaturas distintas para exatamente a mesma mensagem.

Esse algoritmo garante que cada processo irá receber pelo menos $2f + 1$ assinaturas distintas, de modo que seja possível detectar situações em que qualquer mensagem não seja igual às outras, considerando que em sistemas sujeitos a falhas bizantinas, qualquer comportamento fora da especificação já é considerado um comportamento bizantino.

A definição do modelo de falhas bizantinas, engloba também dois tipos de falha que são necessários tratar dentro do nosso sistema. O primeiro deles é o modelo de falha por omissão, no qual algum processo está omitindo mensagens que deveriam ser enviadas para o restante dos processos. A seguir apresentamos o comportamento do algoritmo frente a falhas por omissão. No VCube um processo que omite mensagens será considerado falho, portanto os outros processos que comunicariam com ele, vão encontrar outros caminhos de comunicação. Desta forma, nenhum processo correto fica "isolado", devido ao funcionamento do VCube. Quando um processo é considerado falho (seja por omissão deliberada de mensagens ou atraso), os processos continuam enviando mensagens a ele, porém, enviam também para o próximo processo sem falha no mesmo cluster para garantir a propagação na árvore de difusão. O outro modelo de falhas é o modelo de falhas *crash*/parada, nas quais o processo, por qualquer motivo que seja, para de responder a qualquer estímulo externo. Mais uma vez, a situação é similar a falhas por omissão.

Um dos principais aspectos que devem ser tratados por um algoritmo de difusão bizantina é uma falha bizantina da origem das mensagens, ou seja, o processo que inicia a

Algorithm 1 VCube-SRB: Broadcast Confiável Hierárquico

1: $signed\_value[id] \leftarrow \perp$
2: $Seen[id] \leftarrow \emptyset$
3: $delivered[id] \leftarrow false$

4: **procedure** BROADCAST(id, m) ▷ executado apenas no emissor s
5: $\sigma_s \leftarrow Sign_s(id, m)$
6: $M \leftarrow (id, m, \{\sigma_s\})$
7: VCUBEBROADCAST(M)

8: **upon** `receive` $M = (id, m, \Sigma)$ **from** j
9: **if not** VERIFY(Σ, id, m) **then** ▷ Verifica validade da assinatura
10: **return**
11: **if** $Seen[id][m] \supseteq \Sigma$ **then** ▷ Ignora se não há novas informações
12: **return**
▷ Merge de assinaturas para o mesmo valor
13: $\Sigma \leftarrow Seen[id][m] \cup \Sigma$
14: $Seen[id][m] \leftarrow \Sigma$
▷ Assina apenas o primeiro valor
15: **if** $signed\_value[id] = \perp$ **then**
16: $signed\_value[id] \leftarrow m$
17: $\sigma_i \leftarrow Sign_i(id, m)$
18: $\Sigma \leftarrow \Sigma \cup \{\sigma_i\}$
▷ Propaga sempre (mesmo divergente)
19: $M' \leftarrow (id, m, \Sigma)$
20: VCUBEBROADCAST(M')
▷ Condição de entrega
21: **if** $|\Sigma| \geq 2f + 1$ **and** $\neg delivered[id]$ **then**
22: $delivered[id] \leftarrow true$
23: DELIVER(id, m)

difusão confiável. Quando duas mensagens divergentes são difundidas a partir da origem, o *broadcast* garante que somente uma das mensagem irá obter as $2f + 1$ assinaturas necessárias para entrega. Dessa forma, uma das mensagens será entregue em detrimento da outra.

Em comparação com o funcionamento de protocolos clássicos de difusão confiavel bizantina, como o protocolo de *Reliable Broadcast* de [Bracha 1987], a proposta apresentada nesse artigo, baseada em VCube, privilegia a organização topológica da disseminação por meio de árvores, de modo a reduzir a redundância de mensagens. Diferentemente de protocolos tradicionais, nos quais são necessárias várias rodadas de mensagens de confirmação, a estrutura hierárquica do VCube favorece uma menor troca de mensagens.

4.4. Corretude do Protocolo

Nesta seção, apresentamos uma versão inicial do arrazoado sobre a corretude do protocolo proposto, demonstrando que ele satisfaz as garantias clássicas de difusão confiável bizantina, a saber: Acordo, Validade, Integridade e Terminação.

O algoritmo baseia-se em dois mecanismos fundamentais: (i) a propriedade de assinatura única, que restringe cada processo correto a assinar no máximo um valor por identificador e (ii) a agregação monotônica de assinaturas digitais.

A propriedade central do algoritmo é definida a seguir.

Propriedade 1 (Assinatura Única)**.** *Cada processo correto assina no máximo um valor para uma mensagem com identificador* id*.*

Essa propriedade é garantida diretamente pelo algoritmo, pois um processo assina apenas o primeiro valor válido recebido e ignora solicitações de assinatura para valores divergentes. Essa restrição é essencial para evitar inconsistências na formação de quóruns.

A segurança do protocolo depende da interseção entre conjuntos de assinaturas suficientemente grandes, conforme formalizado no lema a seguir.

Lema 1 (Interseção de Quóruns)**.** *Sejam* Q_1 *e* Q_2 *dois conjuntos de assinaturas tais que* $|Q_1| \geq 2f + 1$ *e* $|Q_2| \geq 2f + 1$*. Então,*

$$|Q_1 \cap Q_2| \geq f + 1$$

Demonstração. Como o sistema possui $n \geq 3f + 1$ processos, quaisquer dois subconjuntos de tamanho $2f + 1$ possuem interseção de pelo menos $f + 1$ elementos. Como no máximo f processos podem ser bizantinos, ao menos um processo correto pertence à interseção. □

Esse resultado implica que dois quóruns suficientemente grandes sempre compartilham ao menos um processo correto, o que será crucial para garantir a propriedade de acordo.

Lema 2 (Integridade)**.** *Se um processo correto entrega uma mensagem* m *com origem em* p*, e* p *é correto, então* m *foi previamente enviada por* p *e é entregue no máximo uma vez por cada processo correto.*

Demonstração. Mensagens entregues por processos corretos devem conter um conjunto de assinaturas válidas, incluindo a assinatura do emissor p. Como as assinaturas digitais são não forjáveis, nenhum processo bizantino pode produzir uma mensagem válida em nome de p sem que p a tenha efetivamente assinado.

Além disso, o algoritmo garante que cada processo correto entrega no máximo uma vez por identificador id, pois a variável $delivered[id]$ impede múltiplas entregas. Portanto, a integridade é satisfeita. □

A propriedade de acordo assegura que processos corretos não divergem quanto ao valor entregue.

Teorema 3 (Acordo)**.** *Nenhum processo correto entrega valores distintos para a mesma mensagem com identificador* id*.*

Demonstração. Suponha, por contradição, que dois processos corretos entreguem valores distintos m e m'. Isso implica a existência de dois conjuntos de assinaturas Q_m e $Q_{m'}$, ambos com cardinalidade ao menos $2f + 1$. Pelo Lema de Interseção de Quóruns, $Q_m \cap Q_{m'}$ contém ao menos $f + 1$ processos. Como ao menos um desses processos é correto, ele teria assinado dois valores distintos, contradizendo a Propriedade de Assinatura Única. □

A propriedade de validade assegura que, caso o emissor seja correto, seu valor será entregue em um tempo finito.

Teorema 4 (Validade)**.** *Se o emissor é correto, então todos os processos corretos entregam o valor enviado.*

Demonstração. O emissor correto inicia a difusão da mensagem utilizando a primitiva VCUBEBROADCAST. Como essa primitiva garante entrega a termo a todos os processos corretos, e como cada processo correto assina o primeiro valor recebido, o conjunto de assinaturas para o valor correto cresce monotonicamente. Em um intervalo de tempo finito, esse conjunto atinge cardinalidade $2f + 1$, permitindo a entrega. □

A propriedade de terminação assegura que todos os processos corretos tomam uma decisão.

Teorema 5 (Terminação)**.** *Todo processo correto entrega algum valor.*

Demonstração. Devido à difusão estruturada da VCUBEBROADCAST, todas as mensagens (inclusive divergentes) são propagadas entre os processos corretos. Como os conjuntos de assinaturas são monotonicamente crescentes e cada processo correto assina no máximo um valor, em um tempo finito algum valor acumula $2f + 1$ assinaturas válidas. Assim, a condição de entrega é satisfeita em todos os processos corretos. □

O protocolo elimina a necessidade de fases explícitas de confirmação, como nos protocolos clássicos de *broadcast* bizantino, substituindo-as por um mecanismo de agregação de assinaturas. A combinação entre difusão estruturada e crescimento monotônico de evidências permite garantir simultaneamente segurança e progresso com menor complexidade de comunicação.

5. Conclusão

O presente artigo apresenta uma estratégia para difusão confiável em ambientes sujeitos a falhas bizantinas. Utiliza agregação de assinaturas para chegar ao número necessário de assinaturas válidas de processos corretos para realizar a entrega da mensagem. Além disso, a partir do emprego da estrutura de árvores dinâmicas construídas sobre o VCube garante-se a propagação de todas as mensagens entre os processos, sejam mensagens corretas ou não. A verificação para entrega dessas mensagens depende das assinaturas recebidas.

Possíveis trabalhos futuros incluem: a) Investigar se a estratégia de assinaturas aninhadas (encadeadas) em substituição à simples agregação de assinaturas. Esta abordagem pode permitir a autenticação do caminho percorrido pela mensagem, fornecendo maior rastreabilidade e resistência a ataques de recombinação de evidências. Na versão atual do algoritmo, caso o emissor bizantino envie duas mensagens distintas, é possível que uma delas obtenha um quórum válido de assinaturas e seja entregue, mesmo na presença de divergência; b) Realizar comparações entre as mensagens recebidas para identificar se há divergências. A existência de divergências poderia causar tanto o encerramento prematura da difusão como a detecção do processo com falha bizantina; c) Implementar o algoritmo bem como outras alternativas para difusão confiável bizantina e avaliá-los comparativamente em termos de desempenho, escalabilidade e custo de

comunicação; d) Investigar otimizações adicionais no algoritmo, incluindo a redução do número de mensagens trocadas, melhoria da escalabilidade e da latência, bem como a integração com mecanismos de detecção de falhas bizantinas.

Agradecimentos

O presente trabalho foi realizado com apoio da Coordenação de Aperfeiçoamento de Pessoal de Nível Superior – Brasil (CAPES) – Código de Financiamento 001.

Referências

Albini, L. C. P., Duarte Jr, E. P., and Ziwich, R. P. (2005). A generalized model for distributed comparison-based system-level diagnosis. *JPDC*, 10(3):44–56.

Amoussou-Guenou, Y., Beltrando, L., Herlihy, M., and Potop-Butucaru, M. (2026). Byzantine reliable broadcast and tendermint consensus with trusted components. *Theoretical Computer Science*, page 115763.

Bona, L. C., Duarte Jr, E. P., Mello, S. L., and Fonseca, K. V. (2006). Hyperbone: Uma rede overlay baseada em hipercubo virtual sobre a internet. *XXIV SBRC*.

Bonomi, S., Farina, G., and Tixeuil, S. (2019). Multi-hop byzantine reliable broadcast with honest dealer made practical. *JPDC*, 25(1):9.

Bracha, G. (1987). Asynchronous byzantine agreement protocols. *Information and computation*, 75(2):130–143.

Cachin, C., Guerraoui, R., and Rodrigues, L. (2011). *Introduction to reliable and secure distributed programming*. Springer Science & Business Media.

Cachin, C., Kursawe, K., Petzold, F., and Shoup, V. (2001). Secure and efficient asynchronous broadcast protocols. In *Annual International Cryptology Conference*, pages 524–541. Springer.

Chandra, T. D. and Toueg, S. (1996). Unreliable failure detectors for reliable distributed systems. *Journal of the ACM (JACM)*, 43(2):225–267.

Correia, M., Neves, N. F., and Veríssimo, P. (2006). From consensus to atomic broadcast: Time-free byzantine-resistant protocols without signatures. *The Computer Journal*, 49(1):82–96.

de Araujo, J. P., Arantes, L., Duarte Jr, E. P., Rodrigues, L. A., and Sens, P. (2019). Vcube-ps: A causal broadcast topic-based publish/subscribe system. *Journal of Parallel and Distributed Computing*, 125:18–30.

De Bona, L. C. E. and Duarte Jr, E. P. (2004). A flexible approach for defining distributed dependable tests in snmp-based network management systems. *Journal of electronic testing*, 20(4):447–454.

Dolev, D. (1981). Unanimity in an unknown and unreliable environment. In *22nd Annual Symposium on Foundations of Computer Science (sfcs 1981)*, pages 159–168. IEEE.

Dolev, D. and Strong, H. R. (1983). Authenticated algorithms for byzantine agreement. *SIAM Journal on Computing*, 12(4):656–666.

Duarte, E. P. and De Bona, L. E. (2002). A dependable snmp-based tool for distributed network management. In *Proceedings International Conference on Dependable Systems and Networks*, pages 279–284. IEEE.

Duarte Jr, E. P., Rodrigues, L. A., Camargo, E. T., and Turchetti, R. C. (2023). The missing piece: a distributed system-level diagnosis model for the implementation of unreliable failure detectors. *Computing*, 105(12):2821–2845.

Duarte Jr, E. P., Ziwich, R. P., and Albini, L. C. (2011). A survey of comparison-based system-level diagnosis. *ACM Computing Surveys (CSUR)*, 43(3):1–56.

Freitas, A. E. S., Rodrigues, L. A., and Duarte Jr, E. P. (2024). vcubechain: A scalable permissioned blockchain. *Ad Hoc Networks*, 158:103461.

Jeanneau, É., Rodrigues, L. A., Arantes, L., and Duarte Jr, E. P. (2017). An autonomic hierarchical reliable broadcast protocol for asynchronous distributed systems with failure detection. *Journal of the Brazilian Computer Society*, 23(1):15.

Kozhaya, D., Decouchant, J., and Esteves-Verissimo, P. (2018). Rt-byzcast: Byzantine-resilient real-time reliable broadcast. *IEEE Trans. Computers*, 68(3):440–454.

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

Locher, T. (2024). Byzantine reliable broadcast with low communication and time complexity. *arXiv preprint arXiv:2404.08070*.

Rodrigues, L. A., Arantes, L., and Duarte, E. P. (2014). An autonomic implementation of reliable broadcast based on dynamic spanning trees. In *EDCC*, pages 1–12.

Rodrigues, L. A., Arantes, L., and Duarte, E. P. (2016). An autonomic majority quorum system. In *AINA*, pages 524–531. IEEE.

Rodrigues, L. A., Duarte Jr, E. P., and Arantes, L. (2018). A distributed k-mutual exclusion algorithm based on autonomic spanning trees. *Journal of Parallel and Distributed Computing*, 115:41–55.

Rodrigues, L. A., Jeanneau, D., Jr., E. P. D., and Arantes, L. (2017). Uma solução de difusão confiável hierárquica em sistemas distribuídos assíncronos. In *XXXV SBRC*.

Rodrigues, L. A., Jr., E. P. D., and Arantes, L. (2025). Distributed and autonomic minimum spanning trees. *CoRR*, abs/2512.02683.

Ruchel, L. V., de Camargo, E. T., Rodrigues, L. A., Turchetti, R. C., Arantes, L., and Duarte Jr, E. P. (2024). Scalable atomic broadcast: A leaderless hierarchical algorithm. *Journal of Parallel and Distributed Computing*, 184:104789.

Stein, G., Rodrigues, L. A., Duarte Jr, E. P., and Arantes, L. (2023). Diamond-p-vcube: An eventually perfect hierarchical failure detector for asynchronous distributed systems. In *12th LADC*, pages 40–49.

Turchetti, R. C., Duarte Jr, E. P., Arantes, L., and Sens, P. (2016). A QoS-configurable failure detection service for internet applications. *JISA*, 7(1):9.

Ziwich, R. P. and Duarte, E. P. (2016). A nearly optimal comparison-based diagnosis algorithm for systems of arbitrary topology. *IEEE Transactions on Parallel and Distributed Systems*, 27(11):3131–3143.