Good-case Early-Stopping Latency of Synchronous Byzantine Reliable Broadcast: The Deterministic Case

Timothé Albouy, Davide Frey, Michel Raynal, François Taïani

Good-case Early-Stopping Latency of Synchronous Byzantine Reliable Broadcast: The Deterministic Case

Timothé Albouy
Univ Rennes, Inria, CNRS, IRISA

Davide Frey
Univ Rennes, Inria, CNRS, IRISA

Michel Raynal
Univ Rennes, Inria, CNRS, IRISA

François Taïani
Univ Rennes, Inria, CNRS, IRISA

Abstract

This paper considers the good-case latency of Byzantine Reliable Broadcast (BRB), i.e., the time taken by correct processes to deliver a message when the initial sender is correct, and an essential property for practical distributed systems. Although significant strides have been made in recent years on this question, progress has mainly focused on either asynchronous or randomized algorithms. By contrast, the good-case latency of deterministic synchronous BRB under a majority of Byzantine faults has been little studied. In particular, it was not known whether a good-case latency below the worst-case bound of $t + 1$ rounds could be obtained under a Byzantine majority. In this work, we answer this open question positively and propose a deterministic synchronous Byzantine reliable broadcast that achieves a good-case latency of $\mathsf{max}(2, t + 3 - c)$ rounds, where t is the upper bound on the number of Byzantine processes, and c the number of effectively correct processes.

2012 ACM Subject Classification Theory of computation → Distributed algorithms

Keywords and phrases Reliable Broadcast, Byzantine Faults, Synchronous Systems, Good-case latency, Deterministic Algorithms.

Digital Object Identifier 10.4230/LIPIcs.DISC.2022.32

Funding This work was partially supported by the French ANR project ByBLoS (ANR-20-CE25-0002-01), and by the PriCLeSS project granted by the Labex CominLabs excellence laboratory of the French ANR (ANR-10-LABX-07-01).

Acknowledgements The authors would like to thank the anonymous reviewers whose careful reading and suggestions helped them improve their paper.

1 Introduction

Introduced in the eighties [14, 20], Byzantine reliable broadcast (BRB) and Byzantine Broadcast (BB) are two fundamental abstractions of distributed computing [5, 7, 9, 10, 19, 22, 23, 26, 25]. BRB assumes that one particular process, the sender, broadcasts a message to the rest of the system and that correct (a.k.a. honest) processes all deliver the value initially broadcast if the sender is correct or that, if it is not, either all agree on some value or none delivers any value. BB further requires that all correct processes always deliver some value.[1] BRB and BB play a crucial role in many practical distributed applications, from

[1] In this paper, we will tend to conflate the two problems, as the protocols we discuss solve both BB and BRB.

primary-backup state machine replication (SMR) (see, for instance, the discussion in [3]), to broadcast-based money transfer [6, 8, 12, 18].

Good-case latency In broadcast-based money transfer algorithms, for instance, a cryptocurrency is implemented by merely broadcasting the transfer operations originating from one participant (or in some sharded versions [8] from one *authority*) to the rest of the system. These algorithms do not require consensus, and their performance is directly related to the underlying (Byzantine-tolerant) reliable broadcast algorithm they use. Transfers issued by correct participants are guaranteed to terminate and only involve a single broadcast operation invoked by the issuer. As a result, the latency of these algorithms—as experienced by correct participants—solely depends on the *good-case latency* of the BRB algorithm they use, defined as the time taken for all correct parties to deliver a broadcast message when the initial broadcaster is correct [3]. The *good-case latency* of Byzantine-tolerant broadcast algorithms plays a similarly central role in the performance of SMR algorithms, with vast practical consequences for the performance of BFT replication systems, including consortium [2, 17] and committee-based blockchains [11].

Synchronous networks In this paper, we focus on the *good-case latency* of BRB algorithms subject to an *arbitrary number of Byzantine failures* (i.e., we assume $n > t$, where n is the number of processes, and t is an upper bound on the number of Byzantine processes). We further assume that processes can use signatures to authenticate messages. We follow in this respect [16] and [27], and in part [3]. Since BRB cannot be solved even in a partially synchronous model when $t \geq n/3$ [15, 20, 24], we also assume a *synchronous* network, in which messages are delivered during the same round in which they are sent. Although synchronous wide-area networks are challenging to realize in practice, they can be approximated with high probability by using sufficiently high timeouts. Synchronous algorithms are further intriguing in their own right and can yield insights into the nature of distributed computing that are useful beyond their specific use.

Randomized synchronous BRB algorithms The study of randomized synchronous BRB and BB algorithms tolerating arbitrary many Byzantine faults has progressed substantially in recent years [3, 16, 27]. In particular, the solution proposed by Wan, Xiao, Shi, and Devadas [27] and optimized by Abraham, Nayak, Ren, and Xiang [3] presents sublinear worst- and good-case latency bounds in expectation (boiling down to constant numbers of rounds when t, the maximal number of Byzantine processes, is assumed to be a fraction of n). However, these works all rely on *randomization*. Moreover, they generally assume a *weakly adaptive adversary*, an adversary that cannot erase messages sent "just before" a process becomes Byzantine, where "just before" means in the same round. (A notable exception is the solution presented in [26], which tolerates a strongly-adaptive adversary by exploiting time-lock puzzles.) Further, these works do not leverage a lower number of actual faults to provide an *early stopping* property [13]: their latency only involves n, the number of processes, and t, the upper bound on the number of Byzantine processes, but not c, the *effective* number of correct processes. As a result, they cannot exploit a low number of actual failures to provide better latency performance.

This paper's contribution In contrast to randomized solutions, the good-case latency of *deterministic* synchronous BRB and BB algorithms has been little studied. By definition, a deterministic Byzantine-tolerant broadcast algorithm tolerates a strongly adaptive adversary

(one that can remove messages "after the fact"). In the worst case, however, its latency is lower-bounded by $t+1$ rounds [14, 15], and optimal algorithms in this respect have been known since the eighties [14, 20].

An unsolved question to this date is thus whether a good-case latency lower than $t+1$ rounds can be achieved using a deterministic algorithm subject to an arbitrary number of Byzantine faults. In this paper, we answer this question positively and propose a deterministic synchronous Byzantine reliable broadcast that achieves a good-case latency of $\mathsf{max}(2, t+3-c)$ rounds, where t is the upper-bound on the number of Byzantine processes, and c the number of effectively correct processes ($c \geq n-t$). The algorithm we propose does not require correct processes to know either n or c. Moreover, and differently from recently proposed solutions to this problem [3, 16, 27], our solution:

- is deterministic (which is why it trivially tolerates a strongly adaptive adversary),
- only relies on signatures, eschewing richer cryptographic primitives (e.g. distributed random coins [16, 27], verifiable random functions [21, 27] or time-lock puzzles [26]),
- ensures delivery in just 2 rounds in good cases as soon as the effective number of correct processes, c, is at least $t+1$, thus improving on all existing solutions.[2]

The early stopping nature of our solution lends it a substantial advantage even when the effective number of correct processes c is less than $t+1$. For instance, assuming $t < 3/4 \times n$, and an intermediate situation where only $\lfloor t/2 \rfloor$ processes have effectively been compromised, the good-case latency of our algorithm outperforms that of the best-known randomized algorithm up to $n \leq 43$, and is at least as good up to $n \leq 51$, making it competitive in a wide range of small- to medium-scale practical distributed systems.

Our algorithm, although not trivial, remains surprisingly simple. It exploits patterns in signature chains, thus extending an idea as old as the problem itself [14, 20].

2 Background and Related Work

The Synchronous Byzantine Reliable Broadcast problem was first introduced in [24] by Lamport, Shostak, and Pease, who proposed in [20] a deterministic solution based on signature chains that requires $t+1$ rounds (both in good and bad cases), where $t < n$ is an upper bound on the number of Byzantine processes present in the system. This worst-case round complexity was shown by Dolev and Strong [14] to be optimal for deterministic algorithms. This result was later refined by Dolev, Reischuk, and Strong who showed that $\mathsf{min}(n-1, n-c+2, t+1)$ rounds are necessary to realize Synchronous Byzantine Broadcast [13], where $c \geq n-t$ is the effective number of correct processes in a given run. They also present in the same paper a deterministic signature-free algorithm that achieves this bound provided that $n > \mathsf{max}(4t, 2t^2-2t+2)$. The salient properties of this algorithm are summarized in the first column of Table 1 and compared to more recent works and to this paper (last column).

In recent years, substantial progress has been achieved to circumvent the hard bound of $t+1$ rounds for deterministic BRB and BB algorithm by exploiting *randomization*, and generally assuming a weakly adaptive adversary, i.e., an adversary that can adaptively corrupt processes, but cannot remove messages sent in the round when a process becomes Byzantine. Assuming a majority of Byzantine processes, Fitzi and Nielsen proposed in [16] a randomized algorithm that achieves Byzantine Agreement in an expected number of $\lfloor (3t-n)/2 \rfloor + 7 + O(1)$

[2] More generally, our good-case latency is *early stopping* [13], in that, in good cases, our algorithm will stop earlier when the effective number of correct processes c increases.

	Dolev, Reischuk & Strong [13]	Fitzi & Nielsen [16]	Wan et al. [27] + Abraham et al. [3]	This paper
Deterministic	**yes**	no	no	**yes**
Strong adversary	**yes**	no	no	**yes**
Early stopping	**yes**	no	no	**yes**
Dishonest major$^{\text{ity}}$	no	**yes**	**yes**	**yes**
$n >$	$\mathsf{max}(4t, 2t^2{-}2t{+}2)$	–	–	–
Worst-case lat$^{\text{ency}}$	$\mathsf{min}(n{-}c{+}2, t{+}1)$	$\mathsf{max}(7, \lfloor\frac{3t-n}{2}\rfloor{+}7){+}O(1)^*$	$O\big((\frac{n}{n-t})^2\big)^*$	$t+1$
Good-case lat$^{\text{ency}}$	2	$\mathsf{max}(6, \lfloor\frac{3t-n}{2}\rfloor+6)$	$\lceil\frac{n}{n-t}\rceil + \lfloor\frac{n}{n-t}\rfloor$	$\mathsf{max}(2, t{+}3{-}c)$

Table 1 Assumptions, guarantees, and latencies of synchronous signature-based BRB algorithms (* indicates an expected number of rounds)

rounds[3], and a good-case latency of $\lfloor(3t-n)/2\rfloor+6$ deterministic rounds.

In 2020, Wan, Xiao, Shi, and Devadas presented a randomized algorithm that achieves BB in $O\big((\frac{n}{n-t})^2\big)$ expected synchronous rounds [27]. Last year, in an in-depth study of the good-case latency of BB and BRB algorithms [3] (extended version in [4]), Abraham, Nayak, Ren, and Xiang proved a lower bound of $\lfloor n/(n-t)\rfloor-1$ rounds for the good-case latency of synchronous BRB. They then explained how the solution presented in [27] can be optimized to deliver a good-case latency of $\lceil n/(n-t)\rceil+\lfloor n/(n-t)\rfloor$ rounds (about $2n/(n-t)\pm1$) assuming a weakly adaptive adversary.[4]

The properties of these earlier works are summarized in Table 1, together with those of the algorithm we propose. Among these works, only [13] is deterministic and therefore tolerates a strongly adaptive adversary. It imposes, however, a strong constraint on n ($n > \mathsf{max}(4t, 2t^2{-}2t{+}2)$) and does not tolerate a majority of Byzantine processes, which the other algorithms do. Conversely, the algorithms of [3, 16, 27] all tolerate an arbitrary number of Byzantine processes, but contrary to the solution we present, they rely on randomization under a weakly adaptive adversary and do not exploit executions in which the number of Byzantine processes is less than the upper bound t. (They are not early stopping.)

3 Computing Model and Specification

3.1 System model

Process model The system is composed of n synchronous sequential processes denoted $\Pi = \{p_1, ..., p_n\}$. Each process p_i has an identity; all the identities are different and known by all processes. To simplify, we assume that i is the identity of p_i.

Regarding failures, up to t processes can be Byzantine, where a Byzantine process is a process whose behavior does not follow the code specified by its algorithm [20, 24]. Let us notice that Byzantine processes can collude to fool the non-Byzantine processes (also called correct processes). Let us also notice that, in this model, the premature stop (crash) of a process is a Byzantine failure. c denotes the number of processes that effectively behave

[3] More precisely, this expected number of rounds can be broken down into a deterministic number of synchronous rounds followed by an expected number of asynchronous rounds. The exact breakdown depends, in turn, on the choice of shared random coin used in the algorithm.

[4] Although correct processes can deliver their message in about $2\times n/(n-t)$ rounds in this optimized algorithm, they must continue to participate in the algorithm for about the same amount of time, leading to an overall execution time of circa $4\times n/(n-t)$ rounds in good-cases.

correctly in an execution. Both c and n remain unknown to correct processes, but they are used to analyze the properties of our algorithm.

Network model Processes communicate by exchanging messages through a reliable synchronous network, in which messages are delivered in the round in which they were sent.

Security model As earlier works in this area [13, 16, 20, 24, 27], we assume a PKI (Public Key Infrastructure) that provides an ideal signature scheme. Processes can sign the messages they send, verify signatures, and forward content signed by other processes.

3.2 Byzantine Reliable Broadcast

Following [3, 16, 27], we consider a one-shot Byzantine-tolerant reliable broadcast (BRB for short) in which the sending process p_{sender} is known beforehand. The BRB abstraction provides two operations, brb_broadcast and brb_deliver. brb_broadcast(m) is invoked by the sending process p_{sender}. When this happens, we say that p_{sender} brb-broadcasts m. When a process p_i invokes brb_deliver(m) we say that p_i brb-delivers m. The BRB abstraction is specified by the following five properties.

- Safety:
  - BRB-Validity: If a correct process p_i brb-delivers a message m and p_{sender} is correct, then p_{sender} has brb-broadcast m.
  - BRB-No-duplication: A correct process p_i brb-delivers at most one message.
  - BRB-No-duplicity: No two different correct processes brb-deliver different messages.
- Liveliness:
  - BRB-Local-delivery: If p_{sender} is correct and brb-broadcasts a message, then at least one correct process p_j eventually brb-delivers some message.
  - BRB-Global-delivery: If a correct process p_i brb-delivers a message, then all correct processes brb-deliver a message.

4 A deterministic synchronous BRB algorithm

4.1 Underlying intuition

Signature chains The original BRB algorithm of Lamport, Shostak, and Pease uses *signature chains* to propagate what each process knows of the system's state [20]. A signature chain (or chain for short) starts by a message m signed by the sending process, e.g. $(m, i_{\text{sender}}, \sigma_{p_{\text{sender}}})$, where i_{sender} is the identify of the sending process, and $\sigma_{p_{\text{sender}}}$ is a signature of (m, i_{sender}) with p_{sender}'s private key. Such a chain is of length 1, as it contains one signature. A chain of length ℓ is extended by appending the identity $i_{\ell+1}$ of a process $p_{i_{\ell+1}}$ not present in the chain, followed by $p_{i_{\ell+1}}$'s signature of the resulting sequence:

$$(m, i_{\text{sender}}, \sigma_{p_{\text{sender}}}, i_2, \sigma_{p_{i_2}}, .., i_\ell, \sigma_{p_{i_\ell}}, i_{\ell+1}, \sigma_{p_{i_{\ell+1}}}).$$

As in [14, 20], we use the compact notation $m : p_{\text{sender}} : p_{i_2} : .. : p_{i_{\ell+1}}$ to represent such a chain.

Valid chains In Lamport, Shostak, and Pease's original algorithm [20], further formalized in [14], and algorithms based on the same idea [16], correct processes only accept *valid* signature chains, i.e., signature chains that are acyclic and whose length matches the current round. These conditions constrain the disruption power of Byzantine processes by limiting how long they can hide a message from correct processes. In [14, 20], a message is considered

for delivery when backed by at least one chain containing $t+1$ signatures: the length of the chain $(t+1)$ ensures that Byzantine processes cannot reveal some message m to only a subset of correct processes, while hiding it from others, and thus guarantees that all correct processes use the same set of messages to decide which message should be delivered (using a deterministic choice function).

From chains to certificates The protocol we propose generalizes this intuition in a simple, albeit non-trivial, way. Instead of single chains, our algorithm uses *sets of chains* forming a particular *pattern* to trigger delivery. We call these chain patterns *certificates*. We constrain how a certificate might be propagated to limit how long Byzantine processes can hide a valid certificate from correct processes. A given certificate for a message m has a "weight" representing how many processes are "backing" m. To back a message m, a process must have witnessed it at the latest by the end of round 2. The heavier a certificate, the more quickly a correct process can make a delivery decision in the absence of any contradictory information. This approach is beneficial when the initial sender is correct , allowing correct processes to terminate in this case in $\mathsf{max}(2, t+3-c)$ rounds[5].

4.2 Notations

We use the following notations:

- $m : p_{i_1} : p_{i_2} : \cdots : p_{i_\ell}$ is a chain of signatures (or *chain* for short) as in [14, 16, 20]. We say that the *length* of the chain is ℓ. A *valid* chain must start with p_{sender} (i.e. $p_{\text{sender}} = p_{i_1}$), only contain valid signatures, and be acyclic (a process' signature can only appear once in a given chain). As in [14], we assume a filter function removes any invalid chain from the reception queue of correct processes, so that correct processes only receive valid chains. In particular, correct processes will only accept chains of length R during round R. As a shortcut, we might therefore say that a process p_i *has signed a chain* π *in round* R to mean that p_i's signature is the R^{th} signature in π.
- π being a chain of signatures, $\mathsf{message}(\pi)$ denotes the message at the start of the chain. We therefore have $\mathsf{message}(m : p_{\text{sender}} : p_{i_2} : \cdots : p_{i_\ell}) = m$. By extension, if E is a set of chains, $\mathsf{message}(E)$ is the direct image of E by $\mathsf{message}()$.
- M being a set of messages, $\mathsf{choice}(M)$ deterministically returns one of the messages, i.e., the same message m is returned by all correct processes for the same input set M. The function $\mathsf{choice}()$ can be implemented in various ways: e.g., the message with the smallest value or smallest time-stamp. If M is empty, $\mathsf{choice}(M)$ returns $\perp$.
- $\gamma = (p_{i_{k,\gamma}})_{k \in [1..\ell]} \in \Pi^\ell$ being a sequence of ℓ processes, for simplicity, we will use the notation $: \gamma :$ as a shorthand for the fragment of signature chain $: p_{i_{1,\gamma}} : \cdots : p_{i_{\ell,\gamma}} :$. For instance, $m : p_{\text{sender}} : \gamma : p_i$ thus means $m : p_{\text{sender}} : p_{i_{1,\gamma}} : \cdots : p_{i_{\ell,\gamma}} : p_i$. We will similarly equate the sequence γ with its supporting set $\mathsf{set}(\gamma) = \{p_{i_{k,\gamma}}\}_{k \in [1..\ell]}$ when unambiguous. Thus $q \in \gamma$ means $q \in \{p_{i_{k,\gamma}}\}_{k \in [1..\ell]}$, $|\gamma| = |\{p_{i_{k,\gamma}}\}_{k \in [1..\ell]}| = \ell$, $X \cup \gamma = X \cup \{p_{i_{k,\gamma}}\}_{k \in [1..\ell]}$.
- $\gamma = (p_{i_{k,\gamma}})_{k \in [1..\ell]} \in \Pi^\ell$ being a sequence of ℓ processes, we note $\mathsf{truncate}_k(\gamma)$ the subsequence of γ that contains up to its first k elements $(p_{i_{k,\gamma}})_{k \in [1..\mathsf{min}(\ell,k)]}$. If $|\gamma| \leq k$ in particular, $\mathsf{truncate}_k(\gamma) = \gamma$.

[5] Similarly to other works studying synchronous broadcast with a dishonest majority [3, 16, 27], the presented algorithm considers crashed processes as Byzantine, providing no guarantees for them. A simple change, however, which adds one extra round, can ensure that crashed processes that brb-delivers benefit from the BRB-No-duplicity and BRB-Global-delivery properties. See footnote 6.

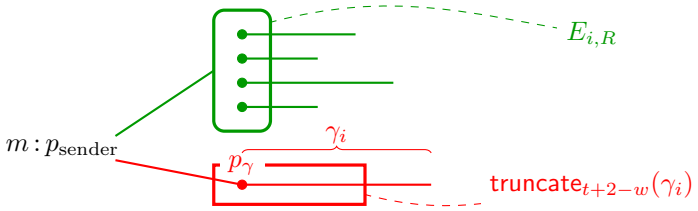

Figure 1 The pattern of signature chains forming a certificate of weight $w = 6$ (p_{sender}, p_γ and the processes of $E_{i,R}$) at round R for message m at p_i. The certificate must verify $\mathsf{set}(\mathsf{truncate}_{t+2-w}(\gamma_i)) \cap E_{i,R} = \emptyset$, which ensures its "conspicuity" (Lemma 2).

5 Description of the algorithm

5.1 Overview

Certificates and revealing chains The algorithm revolves around the notion of *certificate*, which can be informally described as a set of signature chains for a given message m that fits a particular pattern. The *weight* of a certificate is defined as the number of processes whose signature appears within the first two positions of some chains of the certificate. These processes are said to be *backing* m in the certificate.

Just counting and propagating the round-2 signatures that correct processes observe is, however, not enough, as it does not prevent Byzantine processes from hiding part of a certificate from correct processes until the very last moment (round $t + 1$ in our case). The certificates we use therefore add an additional constrain that limits the disruption power of Byzantine processes: a certificate of weight w must contain a "*revealing chain*" $m : p_{\text{sender}} : \gamma_i$ (shown in red in Figure 1) whose makeup must "differ sufficiently" from the backing processes documented by the certificate. "Differ sufficiently" means that besides the processes in position 1 (p_{sender} in all cases) and position 2 (p_γ in Figure 1), the processes from position 3 until position $t + 3 - w$ of this revealing chain should not be backing processes.

This constraint limits what Byzantine processes can do when the sender is Byzantine and allows correct processes to use an early delivery condition that is safe both in good and bad cases. When p_{sender} is Byzantine (bad case), Byzantine processes may collude to forge competing certificates for different messages. When doing so, however, Byzantine processes can only use up to t signatures and must decide whether to invest these t signatures in the backing part of each certificate (thus increasing the certificate's weight) or in the revealing chain of the certificate (thus delaying the time at which the message of a forged certificate must be revealed to correct processes, but reducing the certificate's weight).

Certificate conspicuity The position $t+3-w$ of the revealing chain enforces this constraint. The signatures from positions 3 to $t + 3 - w$ correspond to $(t + 3 - w) - 3 + 1 = t + 1 - w$ processes. Added to the w processes backing the certificate ($E_{i,R} \cup \{p_{\text{sender}}, p_\gamma\}$ in Figure 1), this represents $t + 1 - w + w = t + 1$ processes. These $t + 1$ processes must contain a correct process; therefore, Byzantine processes that seek to forge a certificate must include the signature of a correct process at the latest in round $t + 3 - w$. This correct process ensures that the message of a forged certificate must be revealed to all correct processes at the latest by round $t + 3 - w$. We call the round $R_w = t + 3 - w$ the *conspicuity round* for weight w, and this property *Certificate Conspicuity*.

By contrapositive, certificate conspicuity allows correct processes to ascertain the *nonex-*

istence of a certificate of a given weight for a message. This ability to be sure that a given certificate does not exist, and the ability to propagate certificates that do, are the key ingredients that allow our algorithm to terminate (much) faster than other chain-based deterministic algorithms [14, 16, 20] in good cases, more precisely in $\mathsf{max}(2, t+3-c)$ rounds, where $c = n - f$ is the number of effective correct processes.

An example of certificate Figure 1 shows a certificate of weight $w = 6$ for a message m observed by p_i at round R: each horizontal line represents a chain of signatures that starts with $m : p_{\text{sender}}$, the green and red dots represent processes that have witnessed $m : p_{\text{sender}}$ in round 2 (and are therefore backing m), and $m : p_{\text{sender}} : \gamma_i$ is the revealing chain, such that the process appearing from position 3 to $t + 3 - w$ ($t - 3$ here) in $m : p_{\text{sender}} : \gamma_i$ (or equivalently from position 2 to $t + 2 - w$ in γ_i) do not appear in position 2 of any of the certificate's chains (equivalently the $t+2-w$ truncation of γ_i, noted $\mathsf{truncate}_{t+2-w}(\gamma_i)$, does not appear in $E_{i,R}$, the set of processes in position 2 in chains of the certificate other than $m : p_{\text{sender}} : \gamma_i$).

The certificate depicted in Figure 1 is of weight $w = 6$, as it proves that 6 distinct processes are backing m, i.e. they have signed a chain containing m in round 1 (for p_{sender}) or 2 (for the others). These processes are p_{sender}, p_γ (the first process in γ_i), and the four processes of $E_{i,R}$.

A special case: delivery in round 2 A special case occurs when the weight of a certificate reaches $w = t + 1$. When this happens, any process p_i observing the certificate knows that either p_{sender}, p_γ, or one of the processes of $E_{i,R}$ is correct and, therefore, that all correct processes must have received a chain containing m by round 2. Conversely if p_i has not received any chain containing a message m' by round 2, p_i knows that a certificate of weight $t + 1$ cannot possibly exist for m'. As a result, a correct process that observes a certificate a weight $t + 1$ for m and is not aware of any other message $m' \neq m$ by round 2 can safely brb-deliver m, as no other message will be able to "beat" m with a heavier certificate, even if the sender p_{sender} is Byzantine.

Weak non-intersecting quorums The reasoning for $w = t + 1$ mirrors the mechanism of intersecting quorums used in asynchronous systems and requires a majority of correct processes (or $n > 2t$) to be guaranteed to occur when the sender is correct. The proposed certificate mechanism leverages the additional guarantees that a synchronous system brings to generalize this idea to weaker non-intersecting "quorums", whose ability to trigger a brb-delivery decision requires additional temporal information (waiting until the conspicuity round $R_w = t + 3 - w$).

5.2 Algorithms

In the pseudo-code of our algorithm, we use the operation $\mathsf{broadcast}(m)$ as a shorthand for **for all** $p_j \in \Pi$ **do** send m to p_j **end for**.

For readability, the pseudo-code for the sending process p_{sender} is presented separately in Algorithm 1. To brb-broadcast m, p_{sender} simply signs m and produces the signature chain $m : p_{\text{sender}}$ and broadcasts a protocol message $\text{MSG}(\{m : p_{\text{sender}}\})$ containing this chain to all correct processes, before brb-delivering m locally.

Algorithm 2 constitutes the core of the proposed BRB. It uses up to $t + 1$ synchronous rounds (lines 5-19). Each round is divided into a communication step (lines 6-8), during which processes broadcast and receive messages exchanged during the round, and a computation

Algorithm 1 brb-broadcast operation executed by p_{sender}

```
1 In synchronous round R = 1 do
2 |  broadcast MSG({m : p_sender})
3 |  brb_deliver(m)
4 end round
```

Algorithm 2 Certificate-based Synchronous BRB code for $p_i \neq p_{\text{sender}}$

```
1  Init:
2  |  view_i ← [∅, ..., ∅] ▷Array of size t+1 containing the chains observed by p_i in each round
3  |  ready_i ← false ; to_be_bcast_{i,1} ← ∅
4  end init
5  In each synchronous round R ∈ [1..t+1] do
   |  Communication step
6  |  broadcast MSG(to_be_bcast_{i,R})                    ▷p_i receives its own broadcast
7  |  view_i[R] ← {π ∈ chains_{j,R}, such that MSG(chains_{j,R}) ∈ received_{i,R}}
8  |  to_be_bcast_{i,R+1} ← {π : p_i | π ∈ view_i[R] ∧ p_i ∉ π}
   |  Computation step
9  |  if ready_i then quit()
10 |  known_msgs_{i,R} ← message(⋃_{r∈[1..R]} view_i[r])
11 |  if R = t+1 then
12 |  |  W_i ← {w ∈ ℕ | ∃m ∈ known_msgs_{i,t+1} : certificate_i(m, w)} ; if W_i = ∅ then quit()
13 |  |  wmax_i ← max(W_i)
14 |  |  candidate_msgs_i ← {m ∈ known_msgs_{i,t+1} | certificate_i(m, wmax_i)}
15 |  |  brb_deliver(choice(candidate_msgs_i))
16 |  elseif known_msgs_{i,R} = {m} ∧ certificate_i(m, t+3−R) then
17 |  |  brb_deliver(m) ; ready_i ← true
18 |  end if
19 end round
```

step (lines 9-18) during which they handle received messages and prepare the messages to be sent during the next round. $received_{i,R}$ represents the messages received by process p_i during round R. It is directly updated by the (synchronous) network layer.

R is a global variable containing the sequence number of the current round. $to\_be\_bcast_{i,R}$ contains the signature chains to be broadcast by p_i during round R. In the first round $p_i \neq p_{\text{sender}}$ broadcasts an empty protocol message $\text{MSG}(\emptyset)$. p_i stores in the array $view_i[R]$ the signature chains it receives during round R (line 7). Chains that do not already contain p_i's signature are signed by p_i and stored for broadcasting in the next round (line 8).

p_i's behavior in the computation step depends on whether p_i has reached round $t+1$ or not. In earlier rounds, p_i used the conspicuity property of certificates to detect if a message m is backed by a certificate "heavy enough" that cannot be beaten by any other message $m' \neq m$ (condition at line 16). If this is the case, m is brb-delivered at line 17, and the flag $ready_i$ is toggled to stop the algorithm in the next round.[6] "Heavy enough" means that p_i should

[6] The extra round of communication induced by $ready_i$ is needed to ensure all correct processes observe the same certificate as p_i. However, by delivering as soon as the condition of line 16 is true, the

Algorithm 3 Certificate function

```
1 Function certificate_i(m, w) is
2     S_{i,R} ← { q ∈ Π | m : p_sender : q ∈ truncate_2 ( ⋃_{r'∈[2..t+1]} view_i[r'] ) }
3     return ∃r_i ∈ [2..t+1], ∃γ_i ∈ Π^{r_i−1} : { m : p_sender : γ_i ∈ view_i[r_i] ∧
                                                 { |S_{i,R} \ truncate_{t+2−w}(γ_i)| ≥ w − 2
4 end function
```

observe a certificate of weight at least $w_R = t + 3 - R$ for m. The value w_R is simply the weight whose conspicuity round turns out to be R, as $t+3-w_R = t+3-(t+3-R) = R$ (see Section 5.1). This implies that, by round R, all certificates of weight at least $w_R = t + 3 - R$ must have become conspicuous and allows p_i to make a safe brb-delivery.

If p_i reaches round $t + 1$ without having delivered any message (line 11), it tallies all messages known to it and keeps only messages backed by a certificate with maximal weight. p_i uses a deterministic function choice to break any tie that may appear.

The code for the function $\mathsf{certificate}_i$ executed by p_i is shown in Algorithm 3. $\mathsf{certificate}_i$ first computes the set of all length-2 prefixes of signature chains known to p_i (set $S_{i,R}$ at line 2), and seeks to find a "revealing chain" $m : p_{\text{sender}} : \gamma_i$ known to p_i so that after removing the $t + 2 - w$ truncation[7] of γ_i from $S_{i,R}$, enough distinct processes[8] remain to ensure w processes have signed m by round 2 (see Figure 1 and Section 5.1.)

In terms of vocabulary, we say that p_i *observes a certificate* of weight w for a message m during round R if $\mathsf{certificate}_i(m, w) = \texttt{true}$ during the computation step of round R at p_i (lines 9-18 of Algorithm 2). Note that because $view_i[R]$ is initially empty and is only modified once (during round R, line 2 of Algorithm 2), $\mathsf{certificate}_i(m, w)$ is stable (once true during some round, $\mathsf{certificate}_i(m, w)$ remains true in all subsequent rounds). For the same reasons, and by definition of $\mathsf{certificate}_i$ (Algorithm 3), if $\mathsf{certificate}_i(m, w) = \texttt{true}$ for some weight w, then $\mathsf{certificate}_i(m, w') = \texttt{true}$ for any smaller weight $w' \leq w$.

6 Proof of correctness

▶ **Theorem 1.** *Algorithm 2 implements a Synchronous Byzantine Reliable Broadcast object. If the initial sender p_{sender} is correct, correct processes brb-deliver in at most $\mathsf{max}(2, t+3-c)$ rounds, where c is the effective number of correct processes.*

Remark 1 Note that if $n > 2t$, then because $c \geq n-t$, we have $c \geq t+1$, and $\mathsf{max}(2, t+3-c) = 2$, all correct processes deliver in at most 2 rounds when the sender is correct.

Remark 2 Algorithm 2 can easily be adapted to solve Byzantine Broadcast by modifying line 12 to brb-deliver some default value (e.g., $\perp$) when the set W_i is empty.

---

algorithm does not ensure that crashed processes benefit from the BRB-No-duplicity and BRB-Global-delivery properties. These additional guarantees can be provided at the cost of one extra round by postponing the brb-delivery of m by one round from line 17 to line 9. See footnote 5.

[7] The $t + 2 - w$ truncation of γ_i equals the subchain of $m : p_{\text{sender}} : \gamma_i$ between positions 2 and $t + 3 - w$.

[8] $\mathsf{certificate}_i(m, w)$ uses the threshold $w - 2$ at line 3 to take into account that p_{sender} and the first process of γ_i (called p_γ in Figure 1) are also backing m. See our discussion in Section 5.1.

6.1 Preliminary lemmas

Theorem 1 hinges on two fundamental properties: *Certificate Conspicuity* (Lemma 2) and *Certificate Final Visibility* (Lemma 7). Lemmas 4-5 are used to prove Lemma 7.

As explained earlier, *Certificate Conspicuity* forces Byzantine processes that seek to hide a certificate of weight w for a message m to reveal at the latest by round $t+3-w$ (the "conspicuity round" of w) that m exists. *Certificate Final Visibility* ensures that when the initial sender p_{sender} is malicious, if a correct process reaches round $t+1$, then this correct process observes all the certificates ever observed by other correct processes.

▶ **Lemma 2** (Certificate Conspicuity). *Let p_i and $p_j \neq p_i$ be correct processes, m a message, and $R \in [2..t+1]$ a round. If p_i observes a certificate of weight at least $t+3-R$ for m at some point of its execution (i.e.* $\mathsf{certificate}_i(m, t+3-R) = \texttt{true}$*) and p_j executes round R, then $m \in known\_msgs_{j,R}$ at round R at p_j.*

Sketch of proof. (Detail in the appendix.) Let us note r_i and γ_i a round and a process sequence that render true the condition at line 3 in the definition of $\mathsf{certificate}_i$ for p_i (Alg. 3). The proof depends on whether r_i (the round in which p_i observes the revealing chain $m : p_{\text{sender}} : \gamma_i$, cf. Alg. 3) occurs before or after the round R, the round during which we seek to prove that all correct processes are aware of m. If $r_i < R$, because p_i forwards all chains it has not signed yet, all correct processes observe a chain containing m at the latest by round $r_i + 1 \leq R$. If $r_i \geq R$, the fact that only the first process of $\mathsf{truncate}_{t+2-w}(\gamma_i)$ can be backing m in p_i's certificate implies that $(t+2-w)+(w-2)+1 = t+1$ processes (counting the $(t+2-w)$-prefix of γ_i, the remaining processes of $S_{i,R}$ not in the prefix, and p_{sender}) have signed a chain containing m during the first R rounds of the protocol. One of them must be correct, yielding the lemma. ◀

The following corollary from Lemma 2 states that if a correct process has not seen a message m by round $R \geq 2$, then no certificate of weight $\geq t+3-R$ will ever exist.

▶ **Corollary 3.** *Let p_i and $p_j \neq p_i$ be correct processes, m a message, and $R \in [2..t+1]$ a round. If $m \notin known\_msgs_{i,R}$ at round R at line 16 of Algorithm 2 at p_i, then for any $R' \geq R$,* $\mathsf{certificate}_j(m, t+3-R') = \texttt{false}$ *during all of p_j's execution.*

Proof. Consider p_i and p_j two correct processes. If $m \notin known\_msgs_{i,R}$ at round R of p_i, then by contrapositive of Lemma 2, then $\mathsf{certificate}_j(m, t+3-R) = \texttt{false}$ during all of p_j's execution. Because of the inequality at line 3 of the definition certificate (Alg. 3), $\mathsf{certificate}_j(m, w)$ implies $\mathsf{certificate}_j(m, w')$ for any $w' \leq w$, and therefore $\mathsf{certificate}_j(m, t+3-R) = \texttt{false}$ implies $\mathsf{certificate}_j(m, t+3-R') = \texttt{false}$ for any $R' \geq R$. ◀

In the coming lemmas, we use the following quantity to prove the final visibility of certificates (Lemma 7), which is central to establishing the BRB-No-duplicity property.

$$\text{Let } T_{2,i}[R] \text{ denote } \mathsf{truncate}_2\left(\bigcup_{r' \in [2..R]} view_i[r']\right). \tag{1}$$

$T_{2,i}[R]$ contains all length-2 prefixes $m : p_{\text{sender}} : q$ observed by p_i by round R, i.e. p_i's knowledge during round R of the processes that have signed m by the end of round 2.

The following lemma states that all length-2 prefixes known by a correct process $p_i \neq p_{\text{sender}}$ at round R are known by all other correct processes by round $R+1$.

▶ **Lemma 4.** *Let p_i and $p_j \neq p_i$ be two correct processes, such that p_i executes the computation step (lines 6-8) of at least the $R \leq t$ first rounds, and p_j executes the communication step of at least the first $R+1$ rounds. Then we have $\forall R \in [1..t]$, $T_{2,i}[R] \subseteq T_{2,j}[R+1]$.*

Proof. Note that since p_i and p_j execute Algorithm 2, they are both different from p_{sender}. We prove the lemma by induction.

- Case $R = 1$: $\bigcup_{r' \in [2..1]} view_i[r'] = \emptyset$, and therefore $T_{2,i}[1] = \emptyset$, trivially proving the case.
- Induction case: Let us assume $T_{2,i}[R] \subseteq T_{2,j}[R+1]$ for some $R \in [1..t-1]$.

$$\begin{aligned}
T_{2,i}[R+1] &= \mathsf{truncate}_2\left(\bigcup_{r' \in [2..R+1]} view_i[r']\right), \\
&= \mathsf{truncate}_2\left(\bigcup_{r' \in [2..R]} view_i[r'] \cup view_i[R+1]\right), \\
&= \mathsf{truncate}_2\left(\bigcup_{r' \in [2..R]} view_i[r']\right) \cup \mathsf{truncate}_2\big(view_i[R+1]\big), \\
&= T_{2,i}[R] \cup \mathsf{truncate}_2\big(view_i[R+1]\big), \\
&\subseteq T_{2,j}[R+1] \cup \mathsf{truncate}_2\big(view_i[R+1]\big) && \text{by case assumption,} \\
&\subseteq T_{2,j}[R+2] \cup \mathsf{truncate}_2\big(view_i[R+1]\big) && \text{as } T_{2,j}[R+1] \subseteq T_{2,j}[R+2].
\end{aligned}$$

We now need to show that $\mathsf{truncate}_2\big(view_i[R+1]\big) \subseteq T_{2,j}[R+2]$ to complete the proof. Consider $m : p_{\text{sender}} : \gamma \in view_i[R+1]$, with $\gamma \in \Pi^R$. By assumption, $p_i \neq p_{\text{sender}}$, we must therefore distinguish two cases depending whether p_i appears in γ or not.

- Case 1: If $p_i \in \gamma$, p_i has signed a chain $m : p_{\text{sender}} : \gamma'$ at line 8 of Alg. 2 during a round $R' < R+1$ (where $\gamma' : p_i$ is a prefix of γ), and p_i has broadcast the chain $m : p_{\text{sender}} : \gamma' : p_i$ to all processes (since p_i is correct) at line 6 during the communication step of the following round $R'+1 \leq R+1$. Therefore $m : p_{\text{sender}} : \gamma' : p_i \in view_j[R'+1]$, which implies

$$\mathsf{truncate}_2(m : p_{\text{sender}} : \gamma) = \mathsf{truncate}_2(m : p_{\text{sender}} : \gamma' : p_i) \in \mathsf{truncate}_2(view_j[R'+1]) \subseteq T_{2,j}[R+2].$$

- Case 2: If $p_i \notin \gamma$, p_i signs $m : p_{\text{sender}} : \gamma$ during round $R+1$ and as above broadcasts $m : p_{\text{sender}} : \gamma : p_i$ at round $R+2$ to all processes. (By construction, the fact that p_i executes the computation step of round $R+1 \leq t$ implies that it executes the communication step of round $R+2$.) This similarly implies

$$\mathsf{truncate}_2(m : p_{\text{sender}} : \gamma) = \mathsf{truncate}_2(m : p_{\text{sender}} : \gamma : p_i) \in \mathsf{truncate}_2(view_j[R+2]) \subseteq T_{2,j}[R+2].$$

These two cases show that $\mathsf{truncate}_2\big(view_i[R+1]\big) \subseteq T_{2,j}[R+2]$, which concludes the proof of the lemma. ◀

The following lemma shows that if p_{sender} is Byzantine then all correct processes agree on the length-2 prefixes they have observed by round $t+1$.

▶ **Lemma 5.** *Let p_{sender} be Byzantine, and p_i and p_j be two correct processes that execute the communication step of round $t+1$, then $T_{2,i}[t+1] = T_{2,j}[t+1]$.*

Sketch of proof. (Detail in the appendix.) The proof uses the fact that the length-2 prefixes that p_i receives in round $t+1$ have been propagated by $t+1$ processes. One of these processes must be correct, and because p_{sender} is Byzantine, it must be a process that signed the chain at the earliest in round 2, implying that the length-2 prefix is also known to p_j. This observation, together with Lemma 4 yields the proof. ◀

▶ **Corollary 6.** *Let p_{sender} be Byzantine, p_i and p_j be two correct processes, such that p_i executes the computation step of at least the first $r \in [1..t+1]$ rounds, and p_j executes the communication step of all $t+1$ rounds. Then we have $T_{2,i}[r] \subseteq T_{2,j}[t+1]$.*

Proof. The proof follows either from Lemma 5 or 4, depending on whether $r = t+1$ or not.

- If $r = t + 1$, the corollary follows trivially from Lemma 5.
- If $r < t + 1$, this follows from Lemma 4, and observing that $T_{2,j}[r + 1] \subseteq T_{2,j}[t + 1]$. ◀

The following lemma states that if p_{sender} behaves maliciously (for instance, by sending different messages in round 1), if a certificate of weight w exists for a message m (meaning that it is observed at some point by some correct process), then all correct processes that reach round $t + 1$ observe a certificate of weight w for m by round $t + 1$, a property we have dubbed *Final Certificate Visibility*.

▶ **Lemma 7** (Certificate Final Visibility). *Let p_{sender} be Byzantine, and p_i and p_j be correct processes such that p_i observes* $\mathsf{certificate}_i(m, w) = \texttt{true}$ *at some round $R \in [1..t + 1]$, and p_j executes the communication step of round $t + 1$. Then p_j observes* $\mathsf{certificate}_j(m, w) = \texttt{true}$ *at round $t + 1$.*

Proof. Assume a correct process p_i observes $\mathsf{certificate}_i(m, w) = \texttt{true}$ at some round R. Consider p_j another correct process that reaches round $t + 1$. Without loss of generality, assume $p_j \neq p_i$ (as the case $p_i = p_j$ is trivial).

In the following, r_i and γ_i denote a round and a process sequence that render true the condition at line 3 for p_i in the definition of $\mathsf{certificate}_i$ (Algorithm 3).

Let $E_{i,R}$ be the value of $S_{i,R} \setminus \mathsf{truncate}_{t+2-w}(\gamma_i)$ at line 3 of Alg. 3 at p_i in round R

$$E_{i,R} = \left\{ q \in \Pi \setminus \mathsf{truncate}_{t+2-w}(\gamma_i) \;\middle|\; m : p_{\text{sender}} : q \in \mathsf{truncate}_2\left(\bigcup_{r' \in [2..t+1]} view_i[r']\right) \right\}. \quad (2)$$

By lemma assumption, $|E_{i,R}| \geq w - 2$. Furthermore, as $view_i[r']$ is initially empty and only updated during round r', during round R, $\forall r' > R : view_i[r'] = \emptyset$. At round R, we therefore have $\mathsf{truncate}_2\left(\bigcup_{r' \in [2..t+1]} view_i[r']\right) = \mathsf{truncate}_2\left(\bigcup_{r' \in [2..R]} view_i[r']\right) = T_{2,i}[R]$. We can therefore rewrite (2) into

$$E_{i,R} = \{ q \in \Pi \setminus \mathsf{truncate}_{t+2-w}(\gamma_i) \mid m : p_{\text{sender}} : q \in T_{2,i}[R] \}. \quad (3)$$

The rest of the proof distinguishes two cases, depending on whether $p_j \in \gamma_i$ or not, with the case $p_j \notin \gamma_i$ leading to more sub-cases.

- Case $p_j \in \gamma_i$: As p_j is correct, $p_j \in \gamma_i$ implies p_j has signed and therefore received a chain $m : p_{\text{sender}} : \gamma_j$ at line 8 of Alg. 2 during some round $r_j < r_i$, where γ_j is a prefix of γ_i. As a result of this broadcast, at all rounds higher or equal to r_j, we have

  $$m : p_{\text{sender}} : \gamma_j \in view_j[r_j]. \quad (4)$$

  As γ_j is a prefix of γ_i, $\mathsf{truncate}_{t+2-w}(\gamma_j)$ is also a prefix of $\mathsf{truncate}_{t+2-w}(\gamma_i)$, which implies using (3)

  $$\begin{aligned} E_{i,R} &\subseteq \{ q \in \Pi \setminus \mathsf{truncate}_{t+2-w}(\gamma_j) \mid m : p_{\text{sender}} : q \in T_{2,i}[R] \}, \\ &\subseteq \{ q \in \Pi \setminus \mathsf{truncate}_{t+2-w}(\gamma_j) \mid m : p_{\text{sender}} : q \in T_{2,j}[t + 1] \} \qquad \text{using Corollary 6.} \end{aligned}$$

  As $|E_{i,R}| \geq w - 2$ (since $\mathsf{certificate}_i(m, w) = \texttt{true}$ at round R by lemma assumption), this last inclusion yields

  $$|\{ q \in \Pi \setminus \mathsf{truncate}_{t+2-w}(\gamma_j) \mid m : p_{\text{sender}} : q \in T_{2,j}[t + 1] \}| \geq w - 2. \quad (5)$$

  Equations (4) and (5) render true line 3 of $\mathsf{certificate}_j(m, w)$ (Alg. 3) at round $t + 1$, proving the lemma.

- Case $p_j \notin \gamma_i$: If p_j's signature is not in γ_i the approach to find r_j and γ_j that fulfill line 3 of Alg. 3 for p_j depends on the value of r_i and on whether p_i appears in the set $E_{i,R}$ of length-2 prefixes.
  Line 3 of Alg. 3 implies $\mathit{view}_i[r_i] \neq \emptyset$. As $\mathit{view}_i[r_i]$ is initially empty and only updated in the communication step of round r_i, this implies $R \geq r_i$.
  - Case $r_i = t+1$: $R \geq r_i$ yields $R \geq t+1$. As $R \leq t+1$ by lemma assumption, we conclude $R = t+1$. Since $p_{\text{sender}} \notin \gamma_i$ due to the acyclic nature of signature chains accepted by correct processes, $|\{p_{\text{sender}}\} \cup \gamma_i| = |\{p_{\text{sender}}\}| + |\gamma_i| = 1 + r_i - 1 = t+1$ (by case assumption). $\{p_{\text{sender}}\} \cup \gamma_i$ therefore contains at least one correct process p_k. As p_{sender} is Byzantine (by lemma assumption), $p_k \in \gamma_i$. this means that p_k has broadcast to all processes a chain $m : p_{\text{sender}} : \gamma_j$ during some round $r_j \leq t+1$, where γ_j is a prefix of γ_i. p_j has received this chain, and we therefore have

    $$m : p_{\text{sender}} : \gamma_j \in \mathit{view}_j[r_j]. \tag{6}$$

    Furthermore, since γ_j is a prefix of γ_i, we have

    $$\begin{aligned} E_{t+1,i} &\subseteq \{q \in \Pi \setminus \mathsf{truncate}_{t+2-w}(\gamma_j) \mid m : p_{\text{sender}} : q \in T_{2,i}[t+1]\}, \\ &\subseteq \{q \in \Pi \setminus \mathsf{truncate}_{t+2-w}(\gamma_j) \mid m : p_{\text{sender}} : q \in T_{2,j}[t+1]\}, \quad \text{using Lemma 5.} \end{aligned}$$

    As above, $|E_{t+1,i}| \geq w-2$ (since $R = t+1$, see above) implies that this last set contains at least $w-2$ elements. This fact with (6) renders true line 3 of $\mathsf{certificate}_j(m,w)$ (Alg. 3) at round $t+1$, proving the lemma.
  - Case $r_i < t+1$:
    * Sub-case $p_i \notin E_{i,R}$ or $|\gamma_i| \geq t+2-w$: Line 3 of Alg. 3 implies $m : p_{\text{sender}} : \gamma_i \in \mathit{view}_i[r_i]$. If $p_i \notin \gamma_i$, p_i signs the chain $m : p_{\text{sender}} : \gamma_i$ (line 8, Alg. 2) and broadcasts to all processes (since p_i is correct) the chain $m : p_{\text{sender}} : \gamma_i : p_i$ during the communication step of round $r_i + 1$. If $p_i \in \gamma_i$, this means p_i has already performed these two steps (signing and broadcasting) at some earlier round.
      In both cases, p_j receives a chain $m : p_{\text{sender}} : \gamma_i' : p_i$ during some round $r_j \leq r_i + 1$. By choosing $\gamma_j = \gamma_i' : p_i$, we therefore have

      $$m : p_{\text{sender}} : \gamma_j \in \mathit{view}_j[r_j]. \tag{7}$$

      If $p_i \in \gamma_i$, then γ_j is a prefix of γ_i, $\mathsf{truncate}_{t+2-w}(\gamma_j) \subseteq \mathsf{truncate}_{t+2-w}(\gamma_i)$ and

      $$E_{i,R} \subseteq \{q \in \Pi \setminus \mathsf{truncate}_{t+2-w}(\gamma_j) \mid m : p_{\text{sender}} : q \in T_{2,i}[R]\}. \tag{8}$$

      If $p_i \notin \gamma_i$ and $|\gamma_i| \geq t+2-w$, then $\gamma_i' = \gamma_i$, $\mathsf{truncate}_{t+2-w}(\gamma_j) = \mathsf{truncate}_{t+2-w}(\gamma_i)$ and (8) still holds. Finally if $p_i \notin \gamma_i$ and $|\gamma_i| < t+2-w$, then $\gamma_i' = \gamma_i$, $\mathsf{truncate}_{t+2-w}(\gamma_j)$ contains an extra terminal p_i compared to $\mathsf{truncate}_{t+2-w}(\gamma_i)$. However, by case assumption $|\gamma_i| < t+2-w$ implies $p_i \notin E_{i,R}$, and therefore (8) continues to hold. Using Corollary 6 on (8) we therefore have independently of γ_i

      $$E_{i,R} \subseteq \{q \in \Pi \setminus \mathsf{truncate}_{t+2-w}(\gamma_j) \mid m : p_{\text{sender}} : q \in T_{2,j}[t+1]\}.$$

      As above, using $|E_{i,R}| \geq w-2$ on the above inclusion yields

      $$|\{q \in \Pi \setminus \mathsf{truncate}_{t+2-w}(\gamma_j) \mid m : p_{\text{sender}} : q \in T_{2,j}[t+1]\}| \geq w-2. \tag{9}$$

      Equations (7) and (9) render true line 3 of $\mathsf{certificate}_j(m,w)$ (Alg. 3) at round $t+1$ (since $r_j = r_i + 1 \leq t+1$ by case assumption), proving the lemma.

* Sub-case $p_i \in E_{i,R}$ and $|\gamma_i| < t+2-w$: $p_i \in E_{i,R}$ means that p_i has signed $m : p_{\text{sender}}$. Since p_i is correct, it has therefore broadcast to all processes $m : p_{\text{sender}} : p_i$ during round 2. As a result, from round 2 onward

$$m : p_{\text{sender}} : p_i \in \mathit{view}_j[2]. \tag{10}$$

By definition $r_i \geq 2$ (line 3 of Alg. 3), and therefore $|\gamma_i| = r_i - 1 \geq 1$, implying γ_i is non-empty (and $t + 2 - w > 1$ by sub-case assumption). Consider p_γ the first process in γ_i. As $|\gamma_i| < t + 2 - w$ (sub-case assumption), $p_\gamma \in \mathsf{truncate}_{t+2-w}(\gamma_i)$, and therefore $p_\gamma \notin E_{i,R}$, by definition of $E_{i,R}$ (Equations 2 and 3).
We construct a set of length-2 prefixes for p_j by removing p_i and adding p_γ from/to $E_{i,R}$. More precisely, we have

$$\begin{aligned}
&\big(E_{i,R} \setminus \{p_i\}\big) \cup \{p_\gamma\} \\
&\quad = \big(\{q \in \Pi \setminus \mathsf{truncate}_{t+2-w}(\gamma_i) \mid m : p_{\text{sender}} : q \in T_{2,i}[R]\} \setminus \{p_i\} \big) \cup \{p_\gamma\}, \\
&\quad = \big(\{q \in \Pi \setminus \{p_i\} \mid m : p_{\text{sender}} : q \in T_{2,i}[R]\} \setminus \mathsf{truncate}_{t+2-w}(\gamma_i) \big) \cup \{p_\gamma\}, \\
&\quad \subseteq \{q \in \Pi \setminus \{p_i\} \mid m : p_{\text{sender}} : q \in T_{2,i}[R]\} \cup \{p_\gamma\}.
\end{aligned} \tag{11}$$

Moreover as $m : p_{\text{sender}} : \gamma_i \in \mathit{view}_i[r_i]$ (line 3 of Alg. 3), and $R \geq r_i$, $m : p_{\text{sender}} : p_\gamma = \mathsf{truncate}_2(p_{\text{sender}} : \gamma_i) \in T_{2,i}[R]$. Because $p_\gamma \neq p_i$ (since $p_i \in E_{i,R}$ and $p_\gamma \notin E_{i,R}$), this last statement implies $p_\gamma \in \{q \in \Pi \setminus \{p_i\} \mid m : p_{\text{sender}} : q \in T_{2,i}[R]\}$, which with (11) yields

$$\begin{aligned}
\big(E_{i,R} \setminus \{p_i\}\big) \cup \{p_\gamma\} &\subseteq \{q \in \Pi \setminus \{p_i\} \mid m : p_{\text{sender}} : q \in T_{2,i}[R]\}, \\
&\subseteq \{q \in \Pi \setminus \{p_i\} \mid m : p_{\text{sender}} : q \in T_{2,j}[t+1]\} \quad \text{using Corollary 6.}
\end{aligned}$$

As $p_i \in E_{i,R}$ by case assumption, and $p_\gamma \notin E_{i,R}$, $|\big(E_{i,R} \setminus \{p_i\}\big) \cup \{p_\gamma\}| = |E_{i,R}| - 1 + 1 = |E_{i,R}|$. As $|E_{i,R}| \geq w - 2$ (since $\mathsf{certificate}_i(m, w) = \texttt{true}$ at round R by lemma assumption), the above inclusion leads to

$$\big|\{q \in \Pi \setminus \{p_i\} \mid m : p_{\text{sender}} : q \in T_{2,j}[t+1]\}\big| \geq \big|\big(E_{i,R} \setminus \{p_i\}\big) \cup \{p_\gamma\}\big| = |E_{i,R}| \geq w - 2. \tag{12}$$

By choosing $\gamma_j = p_i$ and $r_j = 2$, (10) and (12) render true line 3 of $\mathsf{certificate}_j(m, w)$ (Alg. 3) at round $t + 1$ (since $t \geq 1$, and $t + 1 \geq 2$), proving the lemma. ◀

6.2 Proof of Theorem 1

Sketch of proof. (Detail in the appendix.)

- BRB-Validity follows from the use of (secure) signatures and the fact that correct processes only accept valid signature chains.
- BRB-No-duplication is ensured by construction of the algorithm, and BRB-Local-delivery from the code executed by p_{sender} when it is correct.
- BRB-No-duplicity follows from BRB-Validity when p_{sender} is correct. When p_{sender} is Byzantine, the no-duplicity follows from the Conspicuity of the certificates (Lemma 2) and from their Final Visibility (Lemma 7).
  Certificate conspicuity ensures that if two processes p_i and p_j brb-deliver a message before round $t + 1$, the process with the "weaker" certificate (say p_j) must be aware of p_i's message when it brb-delivers its own message, and therefore must brb-deliver the same message as p_i, due to the condition at line 17.

If p_i brb-delivers before round $t+1$ and p_j during round $t+1$, then the Final Visibility of certificates guarantees that p_j will observe p_i's certificate at line 12. Corollary 3 implies that p_i's message is guaranteed to have the "heaviest" certificate, ensuring agreement. Finally, if both p_i and p_j deliver in round $t+1$, Final Visibility guarantees they see the same set of messages and certificates and that they brb-deliver the same message.

- The good-case latency of the algorithm, $\mathsf{max}(2, t+3-c)$ rounds, follow from the observation that all correct processes observe a certificate of weight c by the end of round 2 when the initial sender, p_{sender}, is correct. As no other message exists in the system, the condition at line 16 ensures that all correct processes have delivered p_{sender}'s message at the latest either by the end of round 2 or by the end of round $t+3-c$, whichever occurs first.
- BRB-Global-delivery follows from the above reasoning when p_{sender} is correct. When p_{sender} is Byzantine, the property follows from Certificate Finality. ◀

7 Conclusion

Considering n-process synchronous distributed systems where up to $t < n$ processes can be Byzantine, this paper explored the good-case latency of deterministic Byzantine reliable broadcast (BRB) algorithms, the time taken by correct processes to deliver a message when the initial sender is correct.

In contrast to their randomized counterparts, no deterministic BRB algorithm was known that exhibited a good-case latency better than $t+1$ (the worst-case bound) under a majority of Byzantine processes. This article has proposed a novel deterministic synchronous BRB algorithm that substantially improves on this earlier bound and provides a good case latency of $\mathsf{max}(2, t+3-c)$ rounds, where t is the upper bound on the number of Byzantine processes, and c the number of effectively correct processes in the considered run.

The algorithm that has been presented extends the "signature chain mechanism" first proposed four decades ago and allows correct processes to brb-deliver much earlier when the context is favorable. In particular, when the sender is correct, and there are enough effectively correct processes ($c > t$), our algorithm delivers in 2 rounds, thus outperforming all known dishonest-majority BRB algorithms (whether deterministic or randomized).

Several crucial open questions remain, in particular, whether the upper bound of $t+3-c$ rounds can be further improved (for instance, using techniques employed in sub-linear randomized algorithms [4, 27]). In terms of lower bounds, one might ask whether the lower bound of $\lceil n/(n-t) \rceil - 1$ shown in [4] can be refined to include the effective number of correct processes c, and whether this same lower bound can be strengthened in the deterministic case, for instance considering the fact that Byzantine Agreement cannot be solved in a (worst-case) sub-linear communication complexity using algorithms that tolerate a strongly adaptive adversary (which include deterministic algorithms) [1].

References

1 Ittai Abraham, T-H. Hubert Chan, Danny Dolev, Kartik Nayak, Rafael Pass, Ling Ren, and Elaine Shi. Communication complexity of byzantine agreement, revisited. In *ACM Symposium on Principles of Distributed Computing (PODC)*, pages 317–326, New York, NY, USA, 2019.

2 Ittai Abraham, Dahlia Malkhi, Kartik Nayak, Ling Ren, and Maofan Yin. Sync hotstuff: Simple and practical synchronous state machine replication. In *IEEE Symposium on Security and Privacy (S&P)*, pages 106–118, 2020.

3 Ittai Abraham, Kartik Nayak, Ling Ren, and Zhuolun Xiang. Good-case latency of byzantine broadcast: A complete categorization. In *ACM Symposium on Principles of Distributed Computing (PODC)*, pages 331–341, 2021.

4 Ittai Abraham, Kartik Nayak, Ling Ren, and Zhuolun Xiang. Good-case latency of byzantine broadcast: A complete categorization. In *arXiv:2102.07240v2*, pages 1–38, 2021.

5 Hagit Attiya and Jennifer L. Welch. *Distributed computing - fundamentals, simulations, and advanced topics (2. ed.)*. Wiley series on parallel and distributed computing. Wiley, 2004.

6 Alex Auvolat, Davide Frey, Michel Raynal, and François Taïani. Money transfer made simple: a specification, a generic algorithm and its proof. *Bulletin of EATCS*, 132, 2020.

7 Alex Auvolat, Davide Frey, Michel Raynal, and François Taïani. Byzantine-tolerant causal broadcast. *Theoretical Computer Science*, 885:55–68, 2021.

8 Mathieu Baudet, George Danezis, and Alberto Sonnino. Fastpay: High-performance byzantine fault tolerant settlement. In *ACM Advances in Financial Technologies*, pages 163–177, 2020.

9 Gabriel Bracha. Asynchronous byzantine agreement protocols. *Information & Computation*, 75(2):130–143, 1987.

10 Christian Cachin, Rachid Guerraoui, and Luís E. T. Rodrigues. *Introduction to Reliable and Secure Distributed Programming (2. ed.)*. Springer, 2011.

11 Jing Chen and Silvio Micali. Algorand: A secure and efficient distributed ledger. *Theoretical Computer Science*, 777:155–183, 2019.

12 Daniel Collins, Rachid Guerraoui, Jovan Komatovic, Petr Kuznetsov, Matteo Monti, Matej Pavlovic, Yvonne-Anne Pignolet, Dragos-Adrian Seredinschi, Andrei Tonkikh, and Athanasios Xygkis. Online payments by merely broadcasting messages. In *Dependable Systems and Networks (DSN)*, pages 26–38. IEEE, 2020.

13 Danny Dolev, Ruediger Reischuk, and H Raymond Strong. Early stopping in byzantine agreement. *Journal of the ACM*, 37(4):720–741, 1990.

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

15 Cynthia Dwork, Nancy A. Lynch, and Larry J. Stockmeyer. Consensus in the presence of partial synchrony. *Journal of the ACM*, 35(2):288–323, 1988.

16 Matthias Fitzi and Jesper Buus Nielsen. On the number of synchronous rounds sufficient for authenticated byzantine agreement. In *International Symposium on Distributed Computing (DISC)*, pages 449–463. Springer, 2009.

17 Rachid Guerraoui, Nikola Knezevic, Vivien Quéma, and Marko Vukolic. The next 700 BFT protocols. In *EuroSys*, pages 363–376. ACM, 2010.

18 Rachid Guerraoui, Petr Kuznetsov, Matteo Monti, Matej Pavlovic, and Dragos-Adrian Seredinschi. The consensus number of a cryptocurrency. *Distributed Computing*, 35(1):1–15, 2022.

19 Damien Imbs and Michel Raynal. Trading off t-resilience for efficiency in asynchronous byzantine reliable broadcast. *Parallel Processing Letters*, 26(4):1650017:1–1650017:8, 2016.

20 Leslie Lamport, Robert E. Shostak, and Marshall C. Pease. The Byzantine Generals Problem. *ACM Transactions on Programming Languages and Systems*, 4(3):382–401, 1982.

21 Silvio Micali, Michael O. Rabin, and Salil P. Vadhan. Verifiable random functions. In *40th IEEE Symposium on the Foundations of Computer Science (FOCS)*, pages 120–130, 1999.

22 Achour Mostéfaoui, Moumen Hamouma, and Michel Raynal. Signature-free asynchronous byzantine consensus with t 2<n/3 and o(n^2) messages. In *ACM Symposium on Principles of Distributed Computing (PODC)*, pages 2–9. ACM, 2014.

23 Kartik Nayak, Ling Ren, Elaine Shi, Nitin H. Vaidya, and Zhuolun Xiang. Improved extension protocols for byzantine broadcast and agreement. In *International Symposium on Distributed Computing (DISC)*, volume 179 of *LIPIcs*, pages 28:1–28:17, 2020.

24 Marshall C. Pease, Robert E. Shostak, and Leslie Lamport. Reaching agreement in the presence of faults. *Journal of the ACM*, 27(2):228–234, 1980.

25 Michel Raynal. *Fault-Tolerant Message-Passing Distributed Systems - An Algorithmic Approach*. Springer, 2018.

26 Jun Wan, Hanshen Xiao, Srinivas Devadas, and Elaine Shi. Round-efficient byzantine broadcast under strongly adaptive and majority corruptions. In *18th Theory of Cryptography Conference (TCC)*, LNCS 12550, pages 412–456. Springer, 2020.

27 Jun Wan, Hanshen Xiao, Elaine Shi, and Srinivas Devadas. Expected constant round byzantine broadcast under dishonest majority. In *18th Theory of Cryptography Conference (TCC)*, LNCS 12550, pages 381–411. Springer, 2020.

A Appendices

A.1 Proofs of preliminary lemmas

▶ **Lemma 2** (Certificate Conspicuity). *Let p_i and $p_j \neq p_i$ be correct processes, m a message, and $R \in [2..t+1]$ a round. If p_i observes a certificate of weight at least $t+3-R$ for m at some point of its execution (i.e.* $\mathsf{certificate}_i(m, t+3-R) = \texttt{true}$*) and p_j executes round R, then $m \in known\_msgs_{j,R}$ at round R at p_j.*

Proof. Assume a correct process p_i observes $\mathsf{certificate}_i(m, t+3-R) = \texttt{true}$. In the following, r_i and γ_i denote a round and a process sequence that render true the condition at line 3 in the definition of $\mathsf{certificate}_i$ (Algorithm 3).

The remainder of the proof distinguishes two cases, depending on whether $r_i < R$ or not.

- Case $r_i < R$: $m : p_{\text{sender}} : \gamma_i \in view_i[r_i]$ implies that p_i receives $m : p_{\text{sender}} : \gamma_i$ during the communication step of round $r_i < R$ (line 7 of Algorithm 2). If $p_i \notin \gamma_i$, p_i signs the chain (line 8, Alg. 2), and broadcast it during the communication step of round $r_i + 1 \leq R$ (line 6 of the same algorithm). If $p_i \in \gamma_i$, p_i has signed a chain $m : p_{\text{sender}} : \gamma'$ earlier, and broadcast the result before or during round r_i.

  In both cases, this means all other correct processes receive some chain $m : p_{\text{sender}} : \gamma'_i : p_i$ either during or before round R, and therefore that $m \in known\_msgs_{j,R}$ at round R for all correct processes p_j that execute round R.
- Case $r_i \geq R$: Let us note $\gamma_{R-1} = \mathsf{truncate}_{R-1}(\gamma_i)$. $m : p_{\text{sender}} : \gamma_i \in view_i[r_i]$ at line 3 means that $|\gamma_i| = r_i - 1$. Since by case assumption $r_i \geq R$, $|\gamma_i| \geq R-1$, and therefore $|\gamma_{R-1}| = |\mathsf{truncate}_{R-1}(\gamma_i)| = R-1$.

  Let E denote $\left\{ q \in \Pi \setminus \gamma_{R-1} \;\middle|\; m : p_{\text{sender}} : q \in \mathsf{truncate}_2\left(\bigcup_{r' \in [1..r_i]} view_i[r']\right) \right\}$.

  The condition of line 3 implies $|E| \geq t+1-R$. By construction of E, $\gamma_{R-1} \cap E = \emptyset$. Similarly, because correct processes only accept acyclic signature chains, $p_{\text{sender}} \notin E$. For the same reason line 3 of Algorithm 3 implies $p_{\text{sender}} \notin \gamma_{R-1} \subseteq \gamma_i$. E, γ_{R-1}, and $\{p_{\text{sender}}\}$ are therefore pair-wise disjoint. We therefore have $|E \cup \gamma_{R-1} \cup \{p_{\text{sender}}\}| = |E| + |\gamma_{R-1}| + 1 \geq (t+1-R) + (R-1) + 1 = t+1$. $E \cup \gamma_{R-1} \cup \{p_{\text{sender}}\}$ therefore contains at least one correct process, p_k.

  - If $p_k = p_{\text{sender}}$, the sender is correct, and all correct processes observe message m during round 1.
  - If $p_k \in E$, p_k has signed a chain of length 2 with message m and has broadcast this chain to all processes during round 2.
  - Finally, if $p_k \in \gamma_{R-1}$, the facts that $\gamma_{R-1} = \mathsf{truncate}_{R-1}(\gamma_i)$ and $m : p_{\text{sender}} : \gamma_i \in view_i[r_i]$ at line 3 of Alg. 3 imply that p_k has signed a chain with message m and has broadcast this chain to all processes during or before round $R \leq r_i$.

  All three cases imply that all correct processes that have not stopped earlier have observed message m at the latest by the end of the communication step of round R, i.e., that $m \in known\_msgs_{j,R}$ at line 16 of Algorithm 2.

◀

▶ **Lemma 5.** *Let p_{sender} be Byzantine, and p_i and p_j be two correct processes that execute the communication step of round $t+1$, then $T_{2,i}[t+1] = T_{2,j}[t+1]$.*

Proof. By definition

$$\begin{aligned} T_{2,i}[t+1] &= \mathsf{truncate}_2\left(\bigcup_{r' \in [2..t+1]} view_i[r']\right) \\ &= \mathsf{truncate}_2\left(\bigcup_{r' \in [2..t]} view_i[r'] \cup view_i[t+1]\right), \\ &= T_{2,i}[t] \cup \mathsf{truncate}_2\big(view_i[t+1]\big). \end{aligned}$$

Applying Lemma 4 we have $T_{2,i}[t] \subseteq T_{2,j}[t+1]$, which with the previous equality yields

$$T_{2,i}[t+1] \subseteq T_{2,j}[t+1] \cup \mathsf{truncate}_2\big(view_i[t+1]\big). \tag{13}$$

We now prove that $\mathsf{truncate}_2\big(view_i[t+1]\big) \subseteq T_{2,j}[t+1]$. Consider $m : p_{\text{sender}} : \gamma \in view_i[t+1]$. As p_i is correct, it only accepts acyclic signature chains, and $p_{\text{sender}} \notin \gamma$. This implies $|p_{\text{sender}} \cup \gamma| = |p_{\text{sender}}| + |\gamma| = 1 + t$. $\{p_{\text{sender}}\} \cup \gamma$ therefore contains at least one correct process, p_k. As p_{sender} is Byzantine by lemma assumption, $p_k \in \gamma$, and p_k therefore has signed a chain $m : p_{\text{sender}} : \gamma'$ at line 8 of Alg. 2 before or during round t, where $\gamma' : p_k$ is a prefix of γ. As a result, p_k has broadcast the resulting chain $m : p_{\text{sender}} : \gamma' : p_k$ to all other processes during the following round $R' \leq t+1$. This implies $m : p_{\text{sender}} : \gamma' : p_k \in view_j[R']$, and hence

$$\mathsf{truncate}_2(m : p_{\text{sender}} : \gamma) = \mathsf{truncate}_2(m : p_{\text{sender}} : \gamma' : p_k) \in \mathsf{truncate}_2(view_j[R']) \subseteq T_{2,j}[t+1].$$

This last equation shows that $\mathsf{truncate}_2(view_i[t+1]) \subseteq T_{2,j}[t+1]$, which injected in (13) yields $T_{2,i}[t+1] \subseteq T_{2,j}[t+1]$. By inverting p_i and p_j, by the same reasoning we obtain $T_{2,j}[t+1] \subseteq T_{2,i}[t+1]$, which concludes the Lemma's proof. ◀

A.2 Proofs of Theorem 1

The proof of Theorem 1 follows from Lemmas 8-13, which follow.

▶ **Lemma 8.** *Algorithm* 2 *verifies the* BRB-Validity *Property.*

Proof. Consider p_i a correct process.

- If $p_i = p_{\text{sender}}$, the brb-delivery of a message m at line 3 of Algorithm 1 trivially implies that p_i has executed Algorithm 1, and hence has brb-broadcast m.
- If $p_i \neq p_{\text{sender}}$, p_i may brb-deliver a message m either at lines 17 or 15 of Algorithm 2. In both cases, m belongs to some $known\_msgs_{i,R}$ variable computed at line 10, and must therefore appear in a signature chain of the form $m : p_{i_1} : \cdots : p_{i_\ell}$ received by p_i at line 7. As p_i is correct, it only accepts and processes valid chains of signatures by assumption, in which m is first signed by p_{sender} (i.e. $p_{i_1} = p_{\text{sender}}$). Since p_{sender} is correct, and we have assumed signatures to be secure, for m to be signed by p_{sender}, p_{sender} must have executed line 2 of Algorithm 1, and must therefore have brb-broadcast m. ◀

▶ **Lemma 9.** *Algorithm* 2 *verifies the* BRB-No-duplication *Property.*

Proof. Trivially, this is because once a correct process executes a brb_deliver operation (either at line 3 of Algorithm 1, or lines 17 or 15 of Algorithm 2), it terminates its execution, either immediately or at line 9 in the next round, without invoking brb_deliver. ◀

▶ **Lemma 10.** *Algorithm* 2 *verifies the* BRB-Local-delivery *property.*

Proof. The property trivially follows from the code executed by p_{sender} (Algorithm 1). If p_{sender} is correct it executes Algorithm 1 to broadcast a message m, then brb-delivers its own message at line 3. ◀

▶ **Lemma 11.** *Algorithm 2 verifies the* BRB-No-duplicity *Property.*

Proof.

- If p_{sender} is correct, p_{sender} brb-broadcasts one single message m (Algorithm 1), and by BRB-Validity (Lemma 8), all correct processes that do brb-deliver a message only brb-deliver m.
- If p_{sender} is Byzantine, consider two correct processes p_i and p_j (both necessarily different from p_{sender}) that each brb-deliver some message: p_i brb-delivers m_i and p_j brb-delivers m_j. We distinguish three cases depending on the lines at which p_i and p_j execute brb_deliver.
  - Case 1: Assume p_i and p_j both deliver their respective message at line 17 of Algorithm 2. Due to the condition at line 16, there exist two rounds R_i and R_j such that the following holds

$$\begin{aligned} &\mathit{known\_msgs}_{i,R_i} = \{m_i\} \wedge \mathsf{certificate}_i(m_i, t+3-R_i), \text{ and} \\ &\mathit{known\_msgs}_{j,R_j} = \{m_j\} \wedge \mathsf{certificate}_j(m_j, t+3-R_j). \end{aligned}$$

    Without loss of generality, assume $R_i \leq R_j$. By Lemma 2, $\mathsf{certificate}_i(m_i, t+3-R_i) = \texttt{true}$ implies that $m_i \in \mathit{known\_msgs}_{j,R_i}$ at round R_i at p_j (since $R_i \leq R_j$ implies that p_j executes round R_i). $R_i \leq R_j$ further implies $\mathit{known\_msgs}_{j,R_i} \subseteq \mathit{known\_msgs}_{j,R_j}$ by definition of $\mathit{known\_msgs}_{j,-}$, and the way view_j is initialized and updated. $m_i \in \mathit{known\_msgs}_{j,R_i}$ therefore implies that $m_i \in \mathit{known\_msgs}_{j,R_j}$. Combined with $\mathit{known\_msgs}_{j,R_j} = \{m_j\}$, this last statement yields $m_i = m_j$, proving the lemma.
  - Case 2: Assume p_i and p_j both brb-deliver their respective message at line 15 of Algorithm 2, during round $t+1$. Let us consider the two following sets, defined at round $t+1$:

$$W_i = \left\{w \in \mathbb{N} \,\middle|\, \exists m \in \mathit{known\_msgs}_{i,t+1} : \mathsf{certificate}_i(m,w)\right\},$$

    and

$$W_j = \left\{w \in \mathbb{N} \,\middle|\, \exists m \in \mathit{known\_msgs}_{j,t+1} : \mathsf{certificate}_j(m,w)\right\}.$$

    Consider $w \in W_i$. By Lemma 7, $\mathsf{certificate}_i(m,w) = \texttt{true}$ implies $\mathsf{certificate}_j(m,w) = \texttt{true}$ for p_j. Because of line 3 of the definition of certificate (Alg. 3), $\mathsf{certificate}_j(m,w) = \texttt{true}$ implies that $m \in \mathit{known\_msgs}_{j,t+1}$, and therefore that $w \in W_j$. Inverting p_i and p_j leads to $W_i = W_j$, and therefore to $\mathit{wmax}_i = \mathit{wmax}_j$.
    Using $\mathit{wmax}_i = \mathit{wmax}_j$, and following an identical reasoning on $\mathit{candidate\_msgs}_i$ and $\mathit{candidate\_msgs}_j$ produces $\mathit{candidate\_msgs}_i = \mathit{candidate\_msgs}_j$, and therefore that p_i and p_j brb-deliver the same message at line 15.
  - Case 3: Assume p_i brb-delivers m_i at line 17 of Algorithm 2, and p_j brb-delivers m_j at line 15 of the same algorithm. Due to the condition at line 16, there exists a round R_i such that the following holds

$$\mathit{known\_msgs}_{i,R_i} = \{m_i\} \wedge \tag{14}$$

$$\mathsf{certificate}_i(m_i, t+3-R_i). \tag{15}$$

As in Case 2, let us consider the following set defined at round $t+1$ at p_j:

$$W_j = \left\{ w \in \mathbb{N} \,\middle|\, \exists m \in \mathit{known\_msgs}_{j,t+1} : \mathsf{certificate}_j(m, w) \right\}.$$

Because of Lemma 7, $\mathsf{certificate}_i(m_i, t+3-R_i) = \mathtt{true}$ implies that $\mathsf{certificate}_j(m_i, t+3-R_i) = \mathtt{true}$ at p_j at round $t+1$. The condition at line 3 of the code of certificate (Algorithm 3) further means that $\mathsf{certificate}_j(m_i, t+3-R_i) = \mathtt{true}$ implies $m_i \in \mathit{known\_msgs}_{j,t+1}$, and therefore that $t+3-R_i \in W_j$. This last inclusion yields that $\mathit{wmax}_j \geq t+3-R_i$ at line 13 of Algorithm 2.
m_j is brb-delivered by p_j at line 15. Therefore by construction $\mathsf{certificate}_j(m_j, \mathit{wmax}_j) = \mathtt{true}$. Due the condition at line 3 in the code of certificate (Alg. 3), $\mathit{wmax}_j \geq t+3-R_i$ and $\mathsf{certificate}_j(m_j, \mathit{wmax}_j) = \mathtt{true}$ yield $\mathsf{certificate}_j(m_j, t+3-R_i) = \mathtt{true}$. Using Lemma 2 this last statement implies that $m_j \in \mathit{known\_msgs}_{i,R_i}$ at round R_i at p_i. Combined with (14), this leads to $m_j = m_i$, proving the case and concluding the lemma. ◀

▶ **Lemma 12.** *If p_{sender} is correct, correct processes brb-deliver the message m brb-broadcast by p_{sender} in at most $\mathsf{max}(2, t+3-c)$ rounds, where $c = n-f$ is the number of effective correct processes.*

Proof. If p_{sender} is correct, it broadcasts $m : p_{\text{sender}}$ to all correct processes (line 2 or Alg. 1), and brb-delivers its own message in round 1. Every correct process p_j other than p_{sender} receives $m : p_{\text{sender}}$ in round 1, and broadcasts $m : p_{\text{sender}} : p_j$ in round 2. At the end of round 2, a correct process p_i has therefore received at least $c-1$ length-2 signature chains for m:

$$\left\{ m : p_{\text{sender}} : p_j \;\middle|\; p_j \in \Pi_c \setminus \{p_{\text{sender}}\} \right\} \subseteq \mathit{view}_i[2], \tag{16}$$

where Π_c is the set of correct processes. In round 2, we therefore have

$$\forall q \in \Pi_c \setminus \{p_{\text{sender}}, p_i\},\ m : p_{\text{sender}} : q \in \mathsf{truncate}_2\left(\mathit{view}_i[2]\right).$$

As at round 2, $\mathit{view}_i[r'] = \emptyset$ for all $r' > 2$, this leads to

$$\forall q \in \Pi_c \setminus \{p_{\text{sender}}, p_i\},\ m : p_{\text{sender}} : q \in \mathsf{truncate}_2\left(\bigcup_{r' \in [2..t+1]} \mathit{view}_i[r']\right)$$

As $\mathsf{truncate}_{t+2-c}(p_i)$ is either the empty sequence or p_i, this further leads to

$$\Pi_c \setminus \{p_{\text{sender}}, p_i\} \subseteq \left\{ q \in \Pi \setminus \mathsf{truncate}_{t+2-c}(p_i) \;\middle|\; m : p_{\text{sender}} : q \in \mathsf{truncate}_2\left(\bigcup_{r' \in [2..t+1]} \mathit{view}_i[r']\right) \right\}.$$

Because p_{sender} and p_i are correct, $|\Pi_c \setminus \{p_{\text{sender}}, p_i\}| = c-2$ and the previous inclusion implies

$$\left| \left\{ q \in \Pi \setminus \mathsf{truncate}_{t+2-c}(p_i) \;\middle|\; m : p_{\text{sender}} : q \in \mathsf{truncate}_2\left(\bigcup_{r' \in [2..t+1]} \mathit{view}_i[r']\right) \right\} \right| \geq c-2. \tag{17}$$

Because p_i is correct, (16) further yields

$$m : p_{\text{sender}} : p_i \in \mathit{view}_i[2]. \tag{18}$$

By choosing $r_i = 2$ and $\gamma_i = p_i$, (17) and (18) render true line 3 of $\mathsf{certificate}_i(m, c)$ (Alg. 3) at round 2. In other words, p_i (and thus every correct process other than p_{sender}) observes

a certificate of weight c for m at round 2. By definition of the certificate function, and construction of $view_i$, $\mathsf{certificate}_i(m, c) = \texttt{true}$ at round 2 implies that $\mathsf{certificate}_i(m, c)$ remains true during the rest of p_i's execution. In addition, as p_{sender} is correct and signatures are secure, $known\_msgs_{i,R}$ does not contain any other message than m. Therefore, if p_i does not brb-deliver m earlier, at the latest at round $R_c = \mathsf{max}(2, t + 3 - c)$ the condition of line 16 becomes true, and p_i brb-delivers m during the same round. ◀

▶ **Lemma 13.** *Algorithm 2 verifies the* BRB-Global-delivery *property.*

Proof.

- If p_{sender} is correct, then using Lemma 12 all correct processes execute brb_deliver, the lemma is verified.
- If p_{sender} is Byzantine, consider p_i and p_j two correct processes ($\{p_i, p_j\} \cap \{p_{\text{sender}}\} = \emptyset$) and assume p_i brb-delivers some message m. Whether p_i brb-delivers m at line 15 or at line 17, the brb-delivery implies that p_i observes $\mathsf{certificate}_i(m, w) = \texttt{true}$ for some $w \in \mathbb{N}$ at some round $R_i \leq t + 1$. Let r_i and γ_i denote a round and a process sequence that render true the condition at line 3 for p_i in the definition of $\mathsf{certificate}_i$ (Algorithm 3) at round R_i. Since at round R_i, $\forall r' \in [R_i + 1..t + 1] : view_i[r'] = \emptyset$. Line 3 further implies that

$$\begin{aligned}\mathsf{truncate}_2(m : p_{\text{sender}} : \gamma_i) &\in \mathsf{truncate}_2(view_i[r_i]) \\ &\subseteq T_{2,i}[r_i] \qquad \text{since } r_i \geq 2, \text{ and by definition of } T_{2,i}[r_i].\end{aligned}$$

  Using Corollary 6 we have $T_{2,i}[r_i] \subseteq T_{2,j}[t + 1]$ and therefore $\mathsf{truncate}_2(m : p_{\text{sender}} : \gamma_i) \in T_{2,j}[t + 1]$.
  Using the definition of $T_{2,j}[t + 1]$ (Equation 1), $\mathsf{truncate}_2(m : p_{\text{sender}} : \gamma_i) \in T_{2,j}[t + 1]$ implies that

$$\exists r_j \in [2..t + 1], \exists \pi_j \in view_j[r_j] : \mathsf{truncate}_2(\pi_j) = \mathsf{truncate}_2(m : p_{\text{sender}} : \gamma_i)$$

  Because p_j is correct, it only accepts valid signature chains, which implies that π_j is of the form $\pi_j = m : p_{\text{sender}} : \gamma_j$ for some $\gamma_j \in \Pi^{r_j - 1}$ (since $\pi_j \in view_j[r_j]$). $m : p_{\text{sender}} : \gamma_j \in view_j[r_j]$ implies $\mathsf{certificate}_j(m, 2) = \texttt{true}$ at p_j at round r_j (since line 3 of Algorithm 3 is trivially true for $w = 2$), and $m \in known\_msgs_{j,r_j}$.
  If we assume p_j does not brb-deliver any message before round $t + 1$, the fact that $\mathsf{certificate}_j(m, 2) = \texttt{true}$ at p_j at round $r_j \leq t + 1$ means that $\mathsf{certificate}_j(m, 2) = \texttt{true}$ at p_j at round $t + 1$, due to the definition of the function certificate (Algorithm 3). Similarly, since $known\_msgs_{j,r_j} \subseteq known\_msgs_{j,t+1}$, $m \in known\_msgs_{j,r_j}$ implies $m \in known\_msgs_{j,t+1}$. The two facts $m \in known\_msgs_{j,t+1}$ and $\mathsf{certificate}_j(m, 2) = \texttt{true}$ at p_j at round $t + 1$ imply $W_j \neq \emptyset$ at line 12 of Algorithm 2, and therefore that $candidate\_msgs_j \neq \emptyset$, leading p_j to brb-deliver some message at line 15. ◀

A.3 Numerical comparison

Assuming $t < 3/4 \times n$, and that only $\lfloor t/2 \rfloor$ processes have effectively been compromised (equivalently, $c = n - \lfloor t/2 \rfloor$), the good-case latency of our algorithm outperforms the optimized version of [27] presented in [4] up to $n \leq 43$, and is at least as good up to $n \leq 51$. (This claim follows from an exhaustive computation of the values of $\mathsf{max}(2, t + 3 - c)$ and $\lceil \frac{n}{n-t} \rceil + \lfloor \frac{n}{n-t} \rfloor$ over $n \in [3..52], t \in [1..\lfloor 3/4 \times n \rfloor - 1]$, with $c = n - \lfloor t/2 \rfloor$.)

It follows that the proposed algorithm is particularly well suited to small- and medium-size synchronous distributed systems.