Good-case Early-Stopping Latency of Synchronous Byzantine Reliable Broadcast: The Deterministic Case (Extended Version)

Timothé Albouy, Davide Frey, Michel Raynal, François Taïani

Good-case Early-Stopping Latency of Synchronous Byzantine Reliable Broadcast: The Deterministic Case (Extended Version)

Timothé Albouy, Davide Frey, Michel Raynal, François Taïani

Univ Rennes, IRISA, CNRS, Inria, 35042 Rennes, France

March 7, 2023

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. This time plays a crucial role in the performance of 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. This work answers this open question positively and proposes 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.

Keywords: Byzantine Fault, Deterministic Algorithm, Genericity, Good-Case Latency, Reliable Broadcast, Synchronous System, Weighted Predicate.

1 Introduction

Introduced in the eighties [16, 23], Byzantine reliable broadcast (BRB) and Byzantine Broadcast (BB) are two fundamental abstractions of distributed computing [5, 7, 11, 12, 22, 25, 26, 29, 28]. 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 state machine replication (SMR) (see, for instance, the discussion in [3]), to broadcast-based money transfer [6, 8, 14, 21].

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 [6, 14, 19]. These algorithms do not require consensus, and their performance is directly related to the underlying (Byzantine-tolerant)

A first version of this paper was initially published at the 36th Int. Symposium on Distributed Computing (DISC 2022).

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

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, 20] and committee-based blockchains [13].

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 [18] and [30], and in part [3]. Since BRB cannot be solved even in a partially synchronous model when $t \geq n/3$ [17, 23, 27], 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, 18, 30]. In particular, the solution proposed by Wan, Xiao, Shi, and Devadas [30] 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*.[2] Further, these works do not leverage a lower number of actual faults to provide an *early stopping* property [15]: 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. In the worst case, however, its latency is lower-bounded by $t + 1$ rounds [16, 17], and optimal algorithms in this respect have been known since the eighties [16, 23].

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. This paper answers this question positively and proposes 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, 18, 30], our solution:

- is deterministic,

[2] In addition, these randomized solutions 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 [29], which tolerates a strongly-adaptive adversary by exploiting time-lock puzzles. By contrast, a deterministic algorithm that tolerates Byzantine processes inherently tolerates a strongly-adaptive adversary, i.e. an adversary which can remove messages "after the fact".

- only relies on signatures, eschewing richer cryptographic primitives (e.g. distributed random coins [18, 30], verifiable random functions [24, 30] or time-lock puzzles [29]),
- 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.

More generally, our good-case latency is *early stopping* [15], in that, in good cases, our algorithm will stop earlier when the effective number of correct processes c increases. This provides a substantial advantage even when $c < t+1$. For instance, assuming $t < 3/4 \times n$, and an intermediate situation where only $\lfloor t/2 \rfloor$ processes are effectively Byzantine, 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.

To construct our solution, we introduce a general family of predicates used to select messages that we have termed *weight-based predicates*. These predicates exploit patterns in signature chains to help correct processes decide when they can safely deliver a message, thus extending an idea as old as the problem itself [16, 23]. We formally define this predicate family, and present a *generic* BRB algorithm that exploits its properties. We then show how Lamport, Shostak and Pease's seminal BRB algorithm [23] can be re-interpreted as a specific instance of our generic construction, and finally present our novel solution as a more advanced example with stronger properties, which yield our much-improved good-case latency.

2 Background and Related Work

The Synchronous Byzantine Reliable Broadcast problem was first introduced in [27] by Lamport, Shostak, and Pease, who proposed in [23] a deterministic solution based on signature chains. This solution 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 [16] 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 [15], 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*. Assuming a majority of Byzantine processes, Fitzi and Nielsen proposed in [18] a randomized algorithm that achieves Byzantine Agreement in an expected number of $\lfloor (3t - n)/2 \rfloor + 7 + O(1)$ 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 [30]. 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 [30] can be optimized to deliver a good-case latency of $\lceil n/(n-t) \rceil + \lfloor n/(n-t) \rfloor$ rounds (about $2n/(n-t) \pm 1$).[4]

[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.

	Dolev, Reischuk & Strong [15]	Fitzi & Nielsen [18]	Wan et al. [30] + Abraham et al. [3]	This paper
Deterministic	**yes**	no	no	**yes**
Early stopping	**yes**	no	no	**yes**
Dishonest majority	no	**yes**	**yes**	**yes**
$n >$	$\mathsf{max}(4t, 2t^2{-}2t{+}2)$	$-$	$-$	$-$
Worst-case latency	$\mathsf{min}(n{-}c{+}2, t{+}1)$	$\mathsf{max}(7, \lfloor\frac{3t-n}{2}\rfloor{+}7){+}O(1)$ *	$O\big((\frac{n}{n-t})^2\big)$ *	$t+1$
Good-case latency	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)

The properties of these earlier works are summarized in Table 1, together with those of the algorithm we propose. Among these works, only [15] is deterministic. 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, 18, 30] all tolerate an arbitrary number of Byzantine processes, but contrary to the solution we present, they rely on randomization 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 consists 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 [23, 27]. Let us notice that Byzantine processes can collude to fool 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. The integer c denotes the number of processes that effectively behave 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 Like earlier works in this area [15, 18, 23, 27, 30], 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, 18, 30], we consider a one-shot Byzantine-tolerant reliable broadcast (BRB for short) in which the sending process $p_{\rm sender}$ is known beforehand. The BRB abstraction provides two operations, brb_broadcast and brb_deliver. brb_broadcast(m) is invoked by the sending process $p_{\rm sender}$. When this happens, we say that $p_{\rm sender}$ 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.
- Liveness:
  - 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 Generic BRB Algorithm Based on Weight-based Predicates

The BRB algorithm introduced in this paper exploits patterns in sets of signature chains to detect when a (correct) process can safely brb-deliver a message m earlier than the worst-case latency $t + 1$. We introduce this new algorithm in two steps: in this section, we first present a family of predicates used to select and rank messages and a generic BRB algorithm based on this predicate family that is designed to provide interesting good-case latency values. In the next section, we then define a particular predicate belonging to this family that achieves a good-case latency of $\mathsf{max}(2, t + 3 - c)$ rounds when used in the generic algorithm of this section.

The family of predicates we introduce exploits *weights* to create a hierarchy between potential candidate messages, and relies on two central properties, *conspicuity* and *final visibility*, that allow correct processes to brb-deliver a message early yet safely in favorable circumstances. In the following, we define the properties predicates belonging to this family must fulfill, show that these properties are sufficient to implement BRB with our generic algorithm (Theorem 1), and finally illustrate how Lamport, Shostak, and Pease's (BRB-LSP in the following) seminal algorithm can be seen as a specific example of our generic construction.

4.1 Underlying intuition

Signature chains The original BRB-LSP algorithm uses *signature chains* to propagate what each process knows of the system's state [23]. A signature chain (or chain for short) starts with a message m signed by the sending process, e.g. $(m, i_{\text{sender}}, \sigma_{p_{\text{sender}}})$, where i_{sender} is the identity 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 [16, 23], we use the compact notation $m{:}p_{\text{sender}}{:}p_{i_2}{:}..{:}p_{i_{\ell+1}}$ to represent such a chain.

Valid chains In BRB-LSP [23], further formalized in [16], and algorithms based on the same idea [18], 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 [16, 23], a message is considered for brb-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 brb-delivered (using a deterministic choice function).

From chains to weight-based predicates The generic algorithm proposed in this paper generalizes this intuition in a simple, albeit non-trivial, way. Instead of single chains, our algorithm detects *sets of chains* forming a particular *pattern* to trigger delivery. Ideally, such a pattern should allow correct processes to brb-deliver early in good cases, while remaining safe in bad ones. To fulfill this goal, our algorithm adopts the same sign-and-retransmit strategy as BRB-LSP. This means that, in a good case execution, p_{sender} signs a unique message m, and this message is necessarily repeated in round 2 by the $c-1$ remaining correct processes, totalling at least c signatures "backing" m by the end of round 2 (that of p_{sender} in round 1, and the other $c-1$ correct processes in round 2). Although correct processes do not know c (they only know the lower bound $n-t$), they can thus assign a *weight* to each message m they observe, depending on the "amount" of backing this message is perceived to enjoy from other processes.

Our intuition consists in combining this notion of weight with the temporal information provided by synchronous rounds to obtain a safe yet good-case-ready *weight-based predicate*. Such a weight-based predicate serves to select candidate messages for delivery, while the perceived weight of a message serves to discriminate between competing messages when p_{sender} is Byzantine. The crux of the problem lies in ensuring that a message m selected by this predicate and brb-delivered early by a correct process p will trump (thanks to its weight) any other potential competitor m' that might surface in later rounds. We solve this difficulty by requiring two properties from a weight-based predicate:

- A weight-based predicate should be *conspicuous*, in the sense that if a process p observes a predicate of weight w for a message m, m should necessarily have become visible to all correct processes at or before *a specific revealing round* that only depends on w, and n, t. By contrapositive, this property allows correct processes to conclude to the *nonexistence* of a predicate of weight w for a message m' if they have not heard of m' after this revealing round, and is thus instrumental to determining that a message m cannot be beaten by any other.

- A weight-based predicate should also be *finally visible*, meaning that if p perceives a predicate of weight w for m, all other correct processes should also observe a predicate of weight w for m at the latest by round $t+1$.

In good cases, the conspicuity and final visibility of a weight-based predicate make it possible for a process p that envisages to brb-deliver a message m to know that (i) no other message can beat m in terms of weight (by waiting until the corresponding revealing round for m's weight), and (ii) that all correct processes will also assign a weight of w to m at the latest by round $t+1$ (thus ensuring that they will also brb-deliver m). In bad cases, conspicuity prevents Byzantine processes from tricking a correct process into delivering early while revealing contradictory information to other correct processes in later rounds, and final visibility guarantees that correct processes observe the same (weight,message) pairs in the final $t+1$ round, ensuring agreement.

The rest of this section The remainder of this section, first introduces a few notations that we use to manipulate (sets of) signature chains and messages (Section 4.2). The generic synchronous BRB algorithm (Algorithms 1 and 2) and the weight-based predicates it relies on are described in Section 4.3. We then formalize the properties that weight-based predicates and their revealing functions must fulfill for our generic algorithm to implement BRB (Section 4.4), a connection that is captured by Theorem 1 (Section 4.5). Finally, to illustrate the generality of the proposed algorithm, we show that BRB-LSP can be interpreted as a special case of our generic construction (Section 4.6).

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 [16, 18, 23]. 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 [16], 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 timestamp). If M is empty, $\mathsf{choice}(M)$ returns $\perp$.

- $\gamma = (p_{i_k})_{k\in[1..\ell]} \in \Pi^\ell$ being a sequence of ℓ processes, for simplicity, we use the notation ${:}\gamma{:}$ as a shorthand for the fragment of signature chain $:p_{i_1}{:}\cdots{:}p_{i_\ell}:$. For instance, $m{:}p_{\text{sender}}{:}\gamma{:}p_i$ thus means $m{:}p_{\text{sender}}{:}p_{i_1}{:}\cdots{:}p_{i_\ell}{:}p_i$. We similarly equate the sequence γ with its supporting set $\mathsf{set}(\gamma) = \{p_{i_k}\}_{k\in[1..\ell]}$ when unambiguous. Thus $q \in \gamma$ means $q \in \{p_{i_k}\}_{k\in[1..\ell]}$, $|\gamma| = |\{p_{i_k}\}_{k\in[1..\ell]}| = \ell$, $X \cup \gamma = X \cup \{p_{i_k}\}_{k\in[1..\ell]}$.

  For simplicity, we extend these notations to chains of signatures. For instance, if $\pi = m{:}p_{i_1}{:}p_{i_2}{:}\cdots{:}p_{i_\ell}$ is a chain and $p \in \Pi$ a process, $p \in \pi$ means $p \in \{p_{i_k}\}_{k\in[1..\ell]}$.

- If $\Gamma \subseteq \Pi^*$ is a set of process sequences (resp. a set of chains), by abuse of notation we note $\mathsf{set}(\Gamma)$ the set of processes that appear in one of the sequences of Γ (resp. whose signature appears in one of the chains of Γ):
$$\mathsf{set}(\Gamma) = \bigcup_{\gamma\in\Gamma} \mathsf{set}(\gamma).$$

- $\gamma = (p_{i_k})_{k\in[1..\ell]} \in \Pi^\ell$ being a sequence of ℓ processes, we note $\mathsf{subchain}(\gamma, k_1, k_2)$ the subsequence of γ that contains its $k_1{}^{\text{th}}$ to $k_2{}^{\text{th}}$ elements (with both $p_{i_{k_1}}$ and $p_{i_{k_2}}$ included). The resulting sub-sequence is truncated accordingly if γ does not contain enough elements. Formally, we have:
$$\mathsf{subchain}(\gamma, k_1, k_2) = (p_{i_k})_{k\in[k_1..\mathsf{min}(\ell,k_2)]}.$$
  If $|\gamma| \leq k$ in particular, $\mathsf{subchain}(\gamma, 1, k) = \gamma$.

  As above, we extend this definition to chains of signatures. If $\pi = m{:}p_{i_1}{:}p_{i_2}{:}\cdots{:}p_{i_\ell}$ is a chain then $\mathsf{subchain}(\pi, k_1, k_2) = (p_{i_k})_{k\in[k_1...\mathsf{min}(\ell,k_2)]}$.

- $\pi = m{:}p_{i_1}{:}p_{i_2}{:}\cdots{:}p_{i_\ell}$ being a chain of signature, we note $\mathsf{truncate}_k(\pi)$ the chain in which only the first k signatures are kept (or all of π if $|\pi| \leq k$): $\mathsf{truncate}_k(\pi) = m{:}p_{i_1}{:}p_{i_2}{:}\cdots{:}p_{i_{\mathsf{min}(k,\ell)}}$.

4.3 A generic weight-based synchronous BRB algorithm

Algorithms 1 and 2 describe a general construction for synchronous deterministic broadcast algorithms that lends itself to good-case latency and early-stopping properties. Our approach is modular and hinges

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

Algorithm 1: brb-broadcast operation executed by $p_{\rm sender}$ at round $R = 1$

```
1  Init: view_i ← ∅; delivered_i ← false; to_bcast_{i,r} ← ∅ for all r ∈ [1..t + 1];
2  In each synchronous round R ∈ [1..t + 1] do
       Communication step
3      if R ≥ 2 then broadcast MSG(to_bcast_{i,R});
4      if delivered_i then quit().
       Computation step                ▷ wait that all messages MSG(−) of round R are in received_{i,R}
5      view_i ← view_i ∪ {π ∈ chains | MSG(chains) ∈ received_{i,R}};
6      to_bcast_{i,R+1} ← {π:p_i | π ∈ view_i[R] ∧ p_i ∉ π};
7      known_msgs_{i,R} ← {m | ∃ m:π ∈ view_i};
8      if known_msgs_{i,R} = {m} ∧ ∃ w ∈ ℕ+ : (WBP(m, w, view_i)) ∧ R ≥ (reveal_round(w)) then
9          brb_deliver(m); delivered_i ← true;
10     else if R = t + 1 then
11         weights_i ← {w ∈ ℕ+ | ∃ m ∈ known_msgs_{i,R} : (WBP(m, w, view_i))};
12         if weights_i ≠ ∅ then
13             candidate_msgs_i ← {m ∈ known_msgs_{i,R} | (WBP(m, max(weights_i), view_i))};
14             brb_deliver(choice(candidate_msgs_i));
15         else brb_deliver(⊥).
```

Algorithm 2: Skeleton of a Weight-based Synchronous BRB algorithm (code for $p_i \neq p_{\rm sender}$). The use of WBP() and reveal_round() are highlighted using rounded boxes.

on two functions: a message-selection predicate (noted WBP, standing for *weight-based predicate*), and a revealing-round function (noted reveal_round).

For readability, Algorithm 1 presents the code for the sending process $p_{\rm sender}$ separately. To brb-broadcast m, $p_{\rm sender}$ simply signs m, produces the signature chain $m{:}p_{\rm sender}$, and broadcasts a protocol message MSG($\{m{:}p_{\rm sender}\}$) containing this chain to all correct processes before brb-delivering m locally. Here, and as in the rest of the paper, the operation broadcast(m) is used as a shorthand for "**for all** $p_j \in \Pi$ **do** send m to p_j **end for**".

Algorithm 2 constitutes the core of the generic BRB algorithm we propose. It uses up to $t + 1$ synchronous rounds (lines 2-15). R is a global variable containing the current round number. Each round is divided into a communication step (lines 3-4), in which processes broadcast and receive messages exchanged during the round, and a computation step (lines 5-15) in which they handle received messages and prepare the messages to be sent during the next round. The set $received_{i,R}$ represents the messages received by process p_i during round R. It is directly updated by the (synchronous) network layer.

The set $to\_bcast_{i,R}$ contains the signature chains to be broadcast by p_i during round R. In the first round, $p_i \neq p_{\rm sender}$ remains silent. Process p_i accumulates in the set $view_i$ the signature chains it receives during each round (line 5). The notation $view_i[R]$ used at line 6 is a shortcut to denote the chains of $view_i$ that contain exactly R signatures and have therefore just been received. More generally we use the notation $view_i[r] \stackrel{\rm def}{=} \{\gamma \in view_i : |\gamma| = r\}$. The chains of length R that do not already

contain p_i's signature are signed by p_i and stored for broadcasting in the next round (line 6).

Process p_i's behavior in the computation step is driven by the predicate WBP and the function reveal_round. The function WBP takes three parameters: a message m, a positive weight $w \in \mathbb{N}^+$, and a set of *valid* chains, $view$, which captures a process's current view. It returns a Boolean value, which when true, indicates that m can be considered as a possible message to be brb-delivered with a weight w according to the information contained in $view$. More formally this can be expressed as

$$\begin{array}{rccc} \mathsf{WBP}: & \mathcal{M} \times \mathbb{N}^+ \times \mathcal{P}\big(\mathsf{valid}(\mathcal{M} \times \Pi^*)\big) & \rightarrow & \{\mathtt{true}, \mathtt{false}\} \\ & (m, w, view) & \mapsto & \mathsf{WBP}(m, w, view), \end{array}$$

where $\mathcal{M}$ is the set of possible messages, $\mathbb{N}^+$ the set of positive integers, and $\mathcal{P}\big(\mathcal{M} \times \mathsf{valid}(\Pi^*)\big)$ the powerset of valid signature chains. In terms of vocabulary, we say that p_i *observes a predicate* of weight w for a message m during round R if $\mathsf{WBP}(m, w, view_i) = \mathtt{true}$ during the computation step of round R at p_i once the new value of $view_i$ has been computed (lines 8-15 of Algorithm 2). The function reveal_round is closely linked to WBP, and helps a process decide when a message of weight w (according to the predicate WBP) is safe to brb-deliver.

$$\begin{array}{rccc} \mathsf{reveal\_round}: & \mathbb{N}^+ & \rightarrow & [1..t+1] \\ & w & \mapsto & \mathsf{reveal\_round}(w). \end{array}$$

How p_i uses the information provided by WBP and reveal_round depends on whether p_i has reached round $t+1$ or not. In earlier rounds, p_i uses the conspicuity property of the predicate WBP to detect if a message m is backed by a predicate "heavy enough" that cannot be beaten by any other message $m' \neq m$ (condition at line 8). If this is the case, m is brb-delivered at line 9, and the flag $delivered_i$ is toggled to stop the algorithm in the next round.[5] "Heavy enough" means that w, the weight of the predicate observed by p_i, should have revealing round $\mathsf{reveal\_round}(w)$ of at most R. This implies that, by round R, all predicates of weight w or more must have become conspicuous and allows p_i to make a safe brb-delivery because it knows that no message m' can exhibit a predicate heavier or equal to w, ensuring that m will prevail in case of conflicts possibly detected by other correct processes. Figure 1 illustrates the above mechanism.

If p_i reaches round $t+1$ without having brb-delivered any message (line 10), it tallies all messages known to it and keeps only messages backed by a predicate WBP with maximal weight $\mathsf{max}(weights_i)$. Process p_i uses a deterministic function $\mathsf{choice}()$ to break any tie between messages. Finally, if no message is known to p_i ($weights_i = \emptyset$), p_i brb-delivers a default special value, $\perp$ (line 15).

4.4 λ_{good}-BRB-robustness and BRB guarantees

The pair $(\mathsf{WBP}, \mathsf{reveal\_round})$ is said to be λ_{good}-*BRB-robust* if WBP and $\mathsf{reveal\_round}()$ exhibit the following properties when used in Algorithm 2, where λ_{good} is an integer that depends on n, t, and c.

The properties are grouped into three blocks. The first block, *Monotony and Local Conspicuity*, states four intuitive properties on the stability of the WBP predicate and the reveal_round function when their weight and view parameters change. The second block, *Safety*, formalizes the conspicuity and the final visibility of a weight-based predicate, two properties that are essential to prove the safe termination of Algorithm 2. Finally, the third block, *Liveness*, contains a single property that describes the behaviour of WBP in good cases, which determines the good-case latency of Algorithm 2.

The tight connection between these properties and BRB broadcast is captured by Theorem 1, described just afterwards. Theorem 1 states that if a pair $(\mathsf{WBP}, \mathsf{reveal\_round})$ is λ_{good}-BRB-robust (i.e.

[5] The extra round of communication induced by $delivered_i$ is needed to ensure all correct processes observe the same predicate WBP as p_i. However, by delivering as soon as the condition of line 8 is true, the 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 9 to line 4.

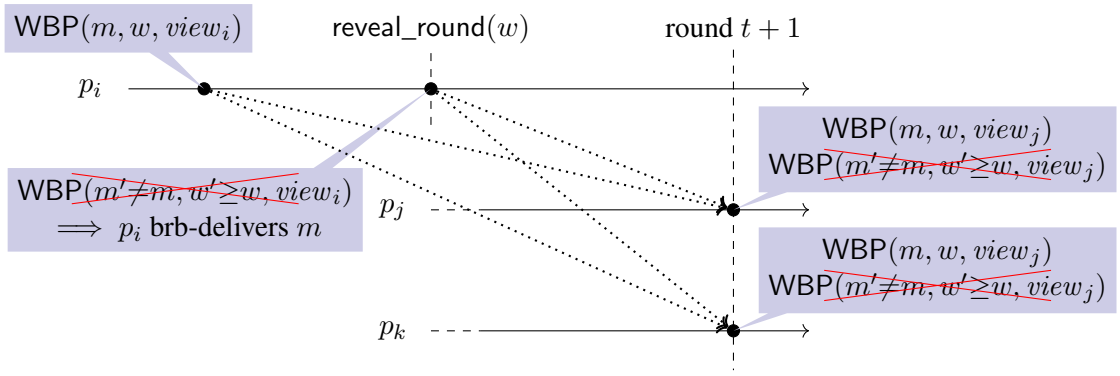

Figure 1: If a correct process p_i observes $\mathsf{WBP}(m, w, view_i)$ or reaches round $\mathsf{reveal\_round}(w)$ only with message m as a candidate, then these events are eventually "propagated" to all other correct processes (here p_j and p_k), at the latest in round $t+1$ in bad cases, and immediately in good cases.

fulfills the properties listed below), then Algorithm 2 implements a BRB broadcast that exhibits a good-case latency of $\mathsf{max}(2, \lambda_{\mathsf{good}})$.

In the following $w, w' \in \mathbb{N}^+$ are weights; $m \in \mathcal{M}$ is a message; and $view$ and $view'$ are sets of valid signature chains.

- Monotony and Local Conspicuity
  - WBP-WEIGHT-MONOTONY: If WBP holds for a given weight, it should also hold for any smaller weight: $\forall w \geq w' : \mathsf{WBP}(m, w, view) \implies \mathsf{WBP}(m, w', view)$.
  - WBP-VIEW-MONOTONY: If WBP holds for a given view, it should also hold for any larger view (in the sense of set inclusion): $\forall view \subseteq view', \mathsf{WBP}(m, w, view) \implies \mathsf{WBP}(m, w, view')$.
  - WBP-REVEALING-ROUND-MONOTONY: A heavier predicate should exhibit an earlier revealing round: $\forall w > w' : \mathsf{reveal\_round}(w) \leq \mathsf{reveal\_round}(w')$.
  - WBP-LOCAL-CONSPICUITY: A message exhibits a weight-1 predicate in a view if and only if it appears in the view: $\mathsf{WBP}(m, 1, view) \iff m \in \mathsf{message}(view)$
- Safety
  - WBP-CONSPICUITY: Consider p_i, p_j two correct processes different from p_{sender}, and w a weight. If p_i observes $\mathsf{WBP}(m, w, view_i)$ during its execution, and p_j executes round $\mathsf{reveal\_round}(w)$, then m is known to p_j at the latest by round $\mathsf{reveal\_round}(w)$.
  - WBP-FINAL-VISIBILITY: Consider p_i, p_j two correct processes different from p_{sender}. If p_{sender} is Byzantine and p_i observes $\mathsf{WBP}(m, w, view_i)$ during its execution, and p_j executes round $t+1$, then p_j observes $\mathsf{WBP}(m, w, view_j)$ at round $t+1$.
- Liveness
  - WBP-GOOD-CASE-LIVENESS: If p_{sender} is correct and brb-broadcasts a message m, then all correct processes $p_i \neq p_{\text{sender}}$ observe $\mathsf{WBP}(m, w_{\mathsf{good}}, view_i)$ during round 2, where $w_{\mathsf{good}} \in \mathbb{N}^+$ is a weight such that $\mathsf{reveal\_round}(w_{\mathsf{good}}) = \lambda_{\mathsf{good}}$.

Theorem 1. *If the pair* $(\mathsf{WBP}, \mathsf{reveal\_round})$ *is* λ_{good}*-BRB-robust, where* λ_{good} *is an integer that depends on* n*,* t*, and* c*, then Algorithm* 2 *implements a Synchronous Byzantine Reliable Broadcast object with a worst-case latency of* $t+1$ *rounds. Furthermore, if the initial sender* p_{sender} *is correct, correct processes brb-deliver in at most* $\mathsf{max}(2, \lambda_{\mathsf{good}})$ *rounds.*

4.5 Proof of Theorem 1

The proof of Theorem 1 follows from Lemmas 1-6, which follow.

Lemma 1. *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 9 or 14 of Algorithm 2. In both cases, m belongs to some $known\_msgs_{i,R}$ variable computed at line 7, 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 5. 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 2. *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 9 or 14 of Algorithm 2), it terminates its execution, either immediately or at line 4 in the next round, without invoking brb_deliver. □

Lemma 3. *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 4. *If the pair* $(\mathsf{WBP}, \mathsf{reveal\_round})$ *is* λ_{good}*-BRB-robust, then, 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 1), 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 9 of Algorithm 2. Due to the condition at line 8, there exist two rounds R_i and R_j such that the following holds

    $$known\_msgs_{i,R_i} = \{m_i\} \wedge \exists\ w_i \in \mathbb{N}^+ : \big(\mathsf{WBP}(m_i, w_i, view_i) \wedge R_i \geq \mathsf{reveal\_round}(w_i)\big),$$

    and

    $$known\_msgs_{j,R_j} = \{m_j\} \wedge \exists\ w_j \in \mathbb{N}^+ : \big(\mathsf{WBP}(m_j, w_j, view_j) \wedge R_j \geq \mathsf{reveal\_round}(w_j)\big).$$

    Without loss of generality, assume $w_i \geq w_j$. By WBP-REVEALING-ROUND-MONOTONY, $\mathsf{reveal\_round}(w_i) \leq \mathsf{reveal\_round}(w_j)$, which leads by case assumption to $R_j \geq$

$\mathsf{reveal\_round}(w_i)$. Process p_j therefore executes round $\mathsf{reveal\_round}(w_i)$, and WBP-CONSPICUITY applies to $\mathsf{WBP}(m_i, w_i, view_i)$ that p_i observes. We conclude that m_i is known to p_j by round $\mathsf{reveal\_round}(w_i)$, i.e. formally $m_i \in known\_msgs_{j,\mathsf{reveal\_round}(w_i)}$. Since $view_i$ keeps growing with each passed round, $R_j \geq \mathsf{reveal\_round}(w_i)$ implies $known\_msgs_{j,\mathsf{reveal\_round}(w_i)} \subseteq known\_msgs_{j,R_j}$, and therefore $m_i \in known\_msgs_{j,R_j}$. Since $known\_msgs_{j,R_j} = \{m_j\}$ by case assumption, this leads to $m_i = m_j$, proving the case.

– Case 2: Assume p_i and p_j both brb-deliver their respective message at line 14 or 15 of Algorithm 2, during round $t + 1$. Let us consider the two following sets, defined at round $t + 1$:

$$weights_i = \{w \in \mathbb{N}^+ \mid \exists\ m \in known\_msgs_{i,t+1} : \mathsf{WBP}(m, w, view_i)\},$$

and

$$weights_j = \{w \in \mathbb{N}^+ \mid \exists\ m \in known\_msgs_{j,t+1} : \mathsf{WBP}(m, w, view_j)\}.$$

Consider $w \in weights_i$, and $m \in known\_msgs_{i,t+1}$ a message such that $\mathsf{WBP}(m, w, view_i)$. Because p_{sender} is Byzantine (by assumption), WBP-FINAL-VISIBILITY applies and $\mathsf{WBP}(m, w, view_i)$ for p_i implies $\mathsf{WBP}(m, w, view_j)$ for p_j at round $t + 1$. By WBP-WEIGHT-MONOTONY and WBP-LOCAL-CONSPICUITY, $\mathsf{WBP}(m, w, view_j)$ at round $t + 1$ implies $\mathsf{WBP}(m, 1, view_j)$ (since $w \geq 1$ by construction), $m \in known\_msgs_{j,t+1}$, and therefore that $w \in weights_j$. Inverting p_i and p_j leads to $weights_i = weights_j$, and therefore to $\mathsf{max}(weights_i) = \mathsf{max}(weights_j)$. Using this last equality, we conclude that either p_i and p_j both have empty $weights_i$ and $weights_j$ sets and get to the else branch at line 15 and both brb-deliver $\perp$, or they both pass the condition at line 12, in which case they must both obtain the same $candidate\_msgs_i$ and $candidate\_msgs_j$ sets, and therefore brb-deliver the same message at line 14.

– Case 3: Assume p_i brb-delivers m_i at line 9 of Algorithm 2 during some round R_i, and p_j brb-delivers m_j at line 14 or 15 of the same algorithm during round $t + 1$. Due to the condition at line 8, there exists a weight $w_i \in \mathbb{N}^+$ such that the following holds

$$known\_msgs_{i,R_i} = \{m_i\} \wedge \mathsf{WBP}(m_i, w_i, view_i) \wedge R_i \geq \mathsf{reveal\_round}(w_i). \quad (1)$$

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

$$weights_j = \{w \in \mathbb{N}^+ \mid \exists\ m \in known\_msgs_{j,t+1} : \mathsf{WBP}(m, w, view_j)\}.$$

As in Case 2, WBP-FINAL-VISIBILITY applies and $\mathsf{WBP}(m_i, w_i, view_i) = \texttt{true}$ at p_i implies $\mathsf{WBP}(m_i, w_i, view_i) = \texttt{true}$ at p_j at round $t + 1$. This fact and WBP-LOCAL-CONSPICUITY further implies $m_i \in known\_msgs_{j,t+1}$, and therefore that $w_i \in weights_j$ (and as $weights_j$ is not empty, p_j cannot brb-deliver at line 15). This last inclusion yields that $\mathsf{max}(weights_j) \geq w_i$ at line 13 of Algorithm 2.

Because m_j is brb-delivered by p_j at line 14, we have by construction $\mathsf{WBP}(m_j, \mathsf{max}(weights_j), view_j) = \texttt{true}$ at line 13 of p_j. By WBP-WEIGHT-MONOTONY, $\mathsf{WBP}(m_j, \mathsf{max}(weights_j), view_j)$ and $\mathsf{max}(weights_j) \geq w_i$ imply $\mathsf{WBP}(m_j, w_i, view_j) = \texttt{true}$ at p_j. Applying WBP-CONSPICUITY, and the fact that $R_i \geq \mathsf{reveal\_round}(w_i)$, this last statement implies that $m_j \in known\_msgs_{i,R_i}$ at round R_i at p_i. Combined with (1), this leads to $m_j = m_i$, proving the case and concluding the lemma. □

Lemma 5. *If the pair* $(\mathsf{WBP}, \mathsf{reveal\_round})$ *is* λ_{good}*-BRB-robust, where* λ_{good} *is an integer that depends on* n*,* t*, and* c*, and if* p_{sender} *is correct, then correct processes brb-deliver the message* m *brb-broadcast by* p_{sender} *in at most* $\mathsf{max}\big(2, \lambda_{\mathsf{good}}\big)$ *rounds.*

Proof. If p_{sender} is correct and brb-broadcasts a message m, it brb-delivers its own message in round 1 (Algorithm 1). By WBP-GOOD-CASE-LIVENESS all other correct processes $p_i \neq p_{\text{sender}}$ observe $\mathsf{WBP}(m, w_{\mathsf{good}}, view_i)$ during round 2. By WBP-VIEW-MONOTONY, and given that $view_i$ can only grow in Algorithm 2, $\mathsf{WBP}(m, w_{\mathsf{good}}, view_i)$ remains true during all subsequent rounds $R \geq 2$ that p_i executes. In addition, as p_{sender} is correct and signatures are secure, $known\_msgs_{i,R}$ does not contain any other message than m. As a result, at the latest in round $\mathsf{max}(2, \mathsf{reveal\_round}(w_{\mathsf{good}})) = \mathsf{max}(2, \lambda_{\mathsf{good}})$, the condition of line 8 becomes true for the message m and the weight w_{good}, and p_i delivers m. □

Lemma 6. *In the worst case, a correct process executing Algorithm* 2 *brb-delivers a message in* $t + 1$ *rounds.*

Proof. By construction of Algorithm 2, processes are guaranteed to brb-deliver at the latest in round $t + 1$ (through the condition at line 10), either at line 14 or 15 (if they have not brb-delivered early at line 9). □

Lemma 7. *Algorithm* 2 *verifies the* BRB-GLOBAL-DELIVERY *property.*

Proof. This property derives trivially from Lemma 6: all correct processes brb-deliver a message at the latest in round $t + 1$. □

4.6 Revisiting Lamport, Shostak, and Pease's algorithm (BRB-LSP)

BRB-LSP [23] can be expressed in the framework of Algorithm 2 by choosing the following definitions and values for WBP, reveal_round, and λ_{good}:

$$\mathsf{WBP}_{\mathsf{LSP}}(m, w, view) \stackrel{\text{def}}{=} \big(m \in \mathsf{message}(view)\big),$$
$$\mathsf{reveal\_round}_{\mathsf{LSP}}(w) \stackrel{\text{def}}{=} t + 1,$$
$$\lambda^{\mathsf{LSP}}_{\mathsf{good}} \stackrel{\text{def}}{=} t + 1.$$

With the above definition of WBP and reveal_round, Algorithm 2 systematically brb-delivers in round $t + 1$ (since the condition at line 8 can only become true in round $t + 1$). Furthermore, when p_i only knows one message m in round $t+1$, the first branch of the "delivery" if block at line 9 is equivalent to its else branch at lines 11-15. In all cases, p_i therefore collects in round $t + 1$ all the messages it has received, and chooses one of them in a deterministic manner, thus reproducing BRB-LSP.

$\mathsf{WBP}_{\mathsf{LSP}}$ and $\mathsf{reveal\_round}_{\mathsf{LSP}}$ correspond to a border example of the use of our generic algorithm, since the weight w plays no part in their definition. However, the pair $(\mathsf{WBP}_{\mathsf{LSP}}, \mathsf{reveal\_round}_{\mathsf{LSP}})$ does fulfill the prerequisites of $(t+1)$-BRB-robustness required to apply Theorem 1, as the following theorem shows.

Theorem 2. *The pair* $(\mathsf{WBP}_{\mathsf{LSP}}, \mathsf{reveal\_round}_{\mathsf{LSP}})$ *is* $(t + 1)$*-BRB-robust.*

Proof. WBP-WEIGHT-MONOTONY and WBP-REVEALING-ROUND-MONOTONY trivially follow from the fact that neither $\mathsf{WBP}_{\mathsf{LSP}}$ nor $\mathsf{reveal\_round}_{\mathsf{LSP}}$ depend on w. Similarly, WBP-VIEW-MONOTONY and WBP-LOCAL-CONSPICUITY directly result from the definition of $\mathsf{WBP}_{\mathsf{LSP}}$.

WBP-CONSPICUITY and WBP-FINAL-VISIBILITY hinge on the central intuition underpinning BRB-LSP. First note that $\mathsf{WBP}_{\mathsf{LSP}}(m, w, view_i) = \texttt{true}$ for some process p_i, message m, and weight

w is equivalent to stating that p_i knows message m, or equivalently that p_i has received some valid chain containing m.

Consider a correct process $p_i \neq p_{\text{sender}}$ that has received some chain containing m. If p_i received m for the first time before round $t+1$, then by construction of Algorithm 2, it has sent it to all other correct processes, which must therefore also know m. If p_i received m for the first time during round $t+1$, then the chain carrying m must contain $t+1$ distinct processes (to be valid), and must therefore contain at least one correct process p_k that must have sent m to all other correct processes. These observations yield the WBP-CONSPICUITY and WBP-FINAL-VISIBILITY properties.

Finally, WBP-GOOD-CASE-LIVENESS is trivially fulfilled by definition of $\mathsf{WBP}_{\mathsf{LSP}}$. □

5 A Deterministic Good-Case BRB Latency in $\mathsf{max}\big(2, t+3-w\big)$ Rounds

5.1 Overview

Theorem 1 states that a deterministic synchronous BRB algorithm that exhibits a good case latency of $\mathsf{max}\big(2, t{+}3{-}w\big)$ can be obtained simply by finding a weight-base predicate and revealing round function that are $\mathsf{max}\big(2, t+3-w\big)$-BRB-robust. Constructing such a pair is however not immediately obvious. We present such a predicate in this section (called *weight-based predicate with good-case latency*, or $\mathsf{WBP}_{\mathsf{GCL}}$ for short) by exploiting a pattern revolving around what we have called a "revealing chain".

Weights and revealing chains The weight-based predicate we propose counts the number of processes whose signature appears within the first two positions of the valid chains a process p_i has received. These processes are said to be *backing* m in p_i's view, and their number is the predicate's weight, w.

Just counting and propagating the round-1 or -2 signatures that correct processes observe is, however, not enough, as it does not prevent Byzantine processes from selectively revealing some round-2 signatures at the very last moment (round $t+1$ in our case), thus preventing correct processes from brb-delivering earlier using this information only. The predicate we use therefore adds an additional constraint that limits the disruption power of Byzantine processes, and provides the conspicuity property required by Algorithm 2: a predicate of weight w must contain a "*revealing chain*" $m{::}\gamma$ whose makeup must "differ sufficiently" from the backing processes documented by the predicate. "Differ sufficiently" means that the processes from position 3 until position $t+3-w$ of this revealing chain (shown in red in Figure 2) should not be backing processes.

This constraint limits what Byzantine processes can do when the sender is Byzantine and ensure this predicate is both conspicuous and finally visible, and thus usable within Algorithm 2. The revealing chain does not prevent Byzantine processes from colluding to forge competing predicates for different messages in bad cases (i.e. when p_{sender} is Byzantine). 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 predicate (thus increasing the predicate's weight) or in their revealing chain (thus delaying the time at which the message of a forged predicate must be revealed to correct processes, but reducing the predicate's weight).

Predicate conspicuity The position $t+3-w$ of the revealing chain enforces the conspicuity of the predicate and yields the good-case latency $\mathsf{max}(2, t{+}3{-}c)$. This is because 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 predicate ($W_{i,R}$ in Figure 2), 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 predicate for a message m must include the signature of a correct process at the latest in round $t+3-w$. This ensures m become visible to all other correct processes by round $t+3-w$, the *revealing round* of the predicate in the terms of Section 4.3.

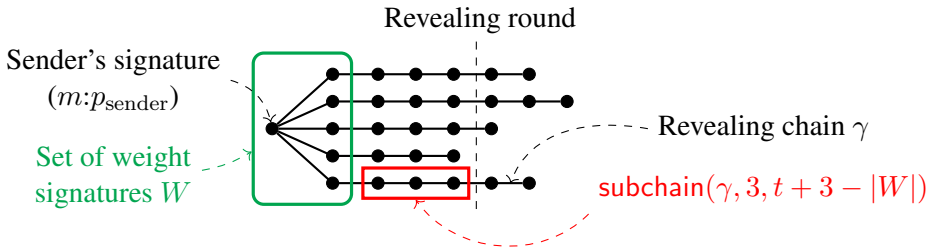

Figure 2: The pattern of signature chains forming a $\mathsf{WBP}_{\mathsf{GCL}}$ predicate of weight $|W| = 6$ at round R for message m at p_i in a setting with $t = 8$. The predicate must verify $\mathsf{subchain}(\gamma, 3, t+3-|W|) \cap W = \emptyset$, which ensures its conspicuity (Lemma 10).

An example of $\mathsf{WBP}_{\mathsf{GCL}}$ Figure 2 shows a $\mathsf{WBP}_{\mathsf{GCL}}$ predicate of weight $w = 6$ for a message m observed by p_i at round R in a setting with $t = 8$: each horizontal line represents a chain of signatures that starts with $m{:}p_{\text{sender}}$, the green rectangle (W) represents processes that have signed m in round 1 or 2 (and are therefore backing m), and $m{:}\gamma$ is the revealing chain, such that the process appearing from position 3 to $t + 3 - w$ ($= t - 3$ here) in $m{:}\gamma$ do not appear in W. The corresponding revealing round is defined as $\mathsf{reveal\_round}_{\mathsf{GCL}}(w) = \mathsf{max}(2, t + 3 - w)$. Because $w = 6$ and $t = 8$, the revealing round is 5 in this example. This function is discussed in more detail in Section 5.2.

It is important to distinguish the round upon which the $\mathsf{WBP}_{\mathsf{GCL}}$ is observed, from the revealing round and the round where the message is brb-delivered. These can be three entirely different rounds. In this example, the $\mathsf{WBP}_{\mathsf{GCL}}$ is observed at round 8 when p_i learns about the second chain from the top, but the revealing round is $\mathsf{reveal\_round}_{\mathsf{GCL}}(6) = t + 3 - 6 = 5$. However, it is possible to have the inverse scenario, where the $\mathsf{WBP}_{\mathsf{GCL}}$ is observed before the revealing round, in which case the process must wait in order to brb-deliver the message. And in the case where multiple distinct messages are observed before the revealing round is attained, the process must wait for round $t + 1$ to brb-deliver one of these messages.

A special case: delivery in round 2 A special case occurs when the weight of a predicate reaches $w = t + 1$. When this happens, any process p_i observing the predicate knows that one of the processes of W is correct and, therefore, that all correct processes must have received a chain containing m by the end of round 2 (the revealing round for the weight $t + 1$). Conversely, if p_i has not received any chain containing a message m' by round 2, p_i knows that a predicate of weight $t + 1$ cannot possibly exist for m'. As a result, a correct process that observes a predicate 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 predicate, 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 predicate 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 revealing round $t + 3 - w$).

5.2 The $\mathsf{WBP}_{\mathsf{GCL}}$ Predicate

This paper's main contribution, a synchronous deterministic Byzantine Reliable Broadcast algorithm with a good case latency of $\mathsf{max}\big(2, t + 3 - w\big)$, is obtained by injecting into the generic code of Algo-

1 $\mathsf{WBP}_{\mathsf{GCL}}(m, w, view) \stackrel{\text{def}}{=}$
2 $\quad \exists\, m{:}\gamma \in view$ such that, when noting $\quad \triangleright m{:}\gamma$ *is the revealing chain.*
3 $\quad\quad S \stackrel{\text{def}}{=} \{q \in \Pi \mid q$ backs m in round 1 or 2 in $view\}$, and
4 $\quad\quad W \stackrel{\text{def}}{=} S \setminus \mathsf{set}\big(\mathsf{subchain}(\gamma, 3, t+3-w)\big)$, the following holds
5 $\quad\quad |W| \geq w.$ $\quad \triangleright$*At least* w *processes back* m *in the first 2 rounds in* $view$.
6 $\mathsf{reveal\_round}_{\mathsf{GCL}}(w) \stackrel{\text{def}}{=} \mathsf{max}(2, t+3-w)$.

Algorithm 3: The weight-based predicate $\mathsf{WBP}_{\mathsf{GCL}}$ and its associated revealing function $\mathsf{reveal\_round}_{\mathsf{GCL}}$. The pair $(\mathsf{WBP}_{\mathsf{GCL}}, \mathsf{reveal\_round}_{\mathsf{GCL}})$ ensures delivery in $\mathsf{max}\big(2, t+3-w\big)$ rounds in good cases when used in Algorithm 2.

rithm 2 the predicate and reveal function $(\mathsf{WBP}_{\mathsf{GCL}}, \mathsf{reveal\_round}_{\mathsf{GCL}})$ shown in Algorithm 3.

$\mathsf{WBP}_{\mathsf{GCL}}(m, w, view)$ first considers S, the set of all processes that have signed[6] the message m either during round 1 or 2. $\mathsf{WBP}_{\mathsf{GCL}}(m, w, view)$ is true if $view$ contains a "revealing chain" backing m, (noted $m{:}\gamma$) such that after removing from S all processes appearing between the position 3 and $t+3-w$ of γ, the resulting set W contains at least w distinct processes that have signed m in round 1 or 2 (see Figure 2 and Section 5.1.)

The existence of γ (the revealing chain) is essential to provide the WBP-CONSPICUITY property of $(\mathsf{WBP}_{\mathsf{GCL}}, \mathsf{reveal\_round}_{\mathsf{GCL}})$. Intuitively, the reason lies in the choice of the boundaries of the subchain $\mathsf{subchain}(\gamma, 3, t+3-w)$.

- If γ contains strictly less than $t+3-w$ processes and appears in the view $view_i$ of a correct process p_i, then p_i must have known m before round $t+3-w$, and will have informed all other correct processes of m's existence at the latest by round $t+3-w$.
- If, on the other hand, γ contains $t+3-w$ processes or more, then $\mathsf{subchain}(\gamma, 3, t+3-w)$ will contain $(t+3-w)-3+1 = t+1-w$ distinct processes. In that case, line 5 ensures that $W \cup \mathsf{subchain}(\gamma, 3, t+3-w)$ contains more than $w + (t+1-w) = t+1$ processes, i.e. does contain at least one correct process. By construction, this correct process will have signed and propagated a chain backing m at the latest by round $t+3-w$.

6 Proof of Correctness

This Section proves the following Theorem and Corollary.

Theorem 3. *The pair* $(\mathsf{WBP}_{\mathsf{GCL}}, \mathsf{reveal\_round}_{\mathsf{GCL}})$ *is* $\mathsf{max}(2, t+3-c)$*-BRB-robust, where* c *is the effective number of correct processes.*

Corollary 1. *The use of* $(\mathsf{WBP}_{\mathsf{GCL}}, \mathsf{reveal\_round}_{\mathsf{GCL}})$ *in Algorithm 2 implements a synchronous deterministic Byzantine Reliable Broadcast algorithm with a worst-case latency of* $t+1$ *and a good case latency of* $\mathsf{max}\big(2, t+3-w\big)$.

Remark 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.

[6]For simplicity, we say that a process p signed m during round r in $view$ to mean that $view$ contains a chain backing m in which p's signature appears in r^{th} position. Formally, both formulations are equivalent for correct processes, but they are not for Byzantine processes, as they can sign chains whenever they wish or even use the signature of other Byzantine processes.

6.1 Preliminary lemmas

Lemma 8. *The predicate* $\mathsf{WBP}_{\mathsf{GCL}}$ *fulfills the* WBP-WEIGHT-MONOTONY *property defined in Section* 4.4.

Proof overview. (Detailed proof below.) The lemma follows from the fact that the weight w of a predicate $\mathsf{WBP}_{\mathsf{GCL}}(m, w, \mathit{view})$ counts processes that back m in round 1 or 2. A (small) technical difficulty is caused by the revealing chain ($m{:}\gamma$ in Algorithm 3), which grows when considering a smaller weight $w' \leq w$. This growth is, however, bounded by the weight difference $w - w'$, which yields the lemma. □

Detailed proof. Assume $\mathsf{WBP}_{\mathsf{GCL}}(m, w, \mathit{view})$ holds, and consider γ, S and W the chain and the sets of processes that render $\mathsf{WBP}_{\mathsf{GCL}}(m, w, \mathit{view})$ true according to Algorithm 3. Consider $w' \in \mathbb{N}^+$ such that $w' \leq w$. Let us note

$$\gamma_w = \mathsf{subchain}(\gamma, 3, t + 3 - w),$$
$$\gamma_{w'} = \mathsf{subchain}(\gamma, 3, t + 3 - w').$$

By definition of subchain, and because $w' \leq w$, γ_w is a prefix of $\gamma_{w'}$, which implies

$$\mathsf{set}(\gamma_{w'}) \supseteq \mathsf{set}(\gamma_w), \tag{2}$$
$$\Big|\mathsf{set}(\gamma_{w'}) \setminus \mathsf{set}(\gamma_w)\Big| \leq (w - w'). \tag{3}$$

Define $W' \stackrel{\text{def}}{=} S \setminus \mathsf{set}(\gamma_{w'})$. The following holds

$$\begin{aligned}
W' &= \Big(S \setminus \mathsf{set}(\gamma_w)\Big) \setminus \Big(\mathsf{set}(\gamma_{w'}) \setminus \mathsf{set}(\gamma_w)\Big), && \text{(using (2))}\\
|W'| &\geq \big|S \setminus \mathsf{set}(\gamma_w)\big| - (w - w'), && \text{(using (3))}\\
|W'| &\geq \big|W\big| - (w - w'), && \text{(by definition of } W)\\
|W'| &\geq w - (w - w') = w'. && \text{(since } \mathsf{WBP}_{\mathsf{GCL}}(m, w, \mathit{view}) \text{ holds)}
\end{aligned}$$

This last equation shows that $\mathsf{WBP}_{\mathsf{GCL}}(m, w', \mathit{view})$ holds using γ, S, and W' in Algorithm 3, concluding the proof. □

Lemma 9. *The pair* $(\mathsf{WBP}_{\mathsf{GCL}}, \mathsf{reveal\_round}_{\mathsf{GCL}})$ *fulfills the* WBP-VIEW-MONOTONY*,* WBP-REVEALING-ROUND-MONOTONY*, and* WBP-LOCAL-CONSPICUITY *properties defined in Section* 4.4.

Proof.

- WBP-VIEW-MONOTONY: Assume $\mathsf{WBP}_{\mathsf{GCL}}(m, w, \mathit{view})$ holds, and consider γ, S and W the chain and the sets of processes that render $\mathsf{WBP}_{\mathsf{GCL}}(m, w, \mathit{view})$ true according to Algorithm 3. If $\mathit{view}' \supseteq \mathit{view}$, then we still have $m{:}\gamma \in \mathit{view}'$. If we note

$$S' \stackrel{\text{def}}{=} \big\{q \in \Pi \mid q \text{ backs } m \text{ in round 1 or 2 in } \mathit{view}'\big\},$$
$$W' \stackrel{\text{def}}{=} S' \setminus \mathsf{set}\big(\mathsf{subchain}(\gamma, 3, t + 3 - w)\big),$$

  $\mathit{view}' \supseteq \mathit{view}$ implies that $S' \supseteq S$ and $W' \supseteq W$, and therefore $|W'| \geq |W| \geq w$, proving the WBP-VIEW-MONOTONY property of $\mathsf{WBP}_{\mathsf{GCL}}$.

- WBP-REVEALING-ROUND-MONOTONY is trivially fulfilled by the definition of $\mathsf{reveal\_round}_{\mathsf{GCL}}$.

- WBP-LOCAL-CONSPICUITY: Assume $\mathsf{WBP}_{\mathsf{GCL}}(m, 1, view)$ holds, and consider γ, S and W the chain and the sets of processes that render $\mathsf{WBP}_{\mathsf{GCL}}(m, 1, view)$ true according to Algorithm 3. Trivially, $m{:}\gamma \in view$ implies $m \in \mathsf{message}(view)$.

  Conversely, assume $m \in \mathsf{message}(view)$. There exists a chain γ_m such that $m{:}\gamma_m \in view$. As all chains of $view$ are valid by assumption, γ_m must start with p_{sender}'s signature. Define the sets S and W as in Algorithm 3, but using γ_m. By construction, $p_{\text{sender}} \in S$. As valid chains are acyclic, $p_{\text{sender}} \notin \mathsf{set}\big(\mathsf{subchain}(\gamma_m, 3, t+3-w)\big)$, and therefore $p_{\text{sender}} \in W$, and $|W| \geq 1$. This last statement implies that $\mathsf{WBP}_{\mathsf{GCL}}(m, 1, view)$ holds. With the previous paragraph, this proves that $\mathsf{WBP}_{\mathsf{GCL}}$ fulfills the WBP-LOCAL-CONSPICUITY, and concludes the lemma. □

Lemma 10. *The pair* $(\mathsf{WBP}_{\mathsf{GCL}}, \mathsf{reveal\_round}_{\mathsf{GCL}})$ *fulfills the* WBP-CONSPICUITY *property.*

Proof overview. (Detailed proof below.) Let us note γ_i the chain that renders true $\mathsf{WBP}_{\mathsf{GCL}}(m, w, view_i)$ for p_i (Alg. 3). The proof depends on whether $|\gamma_i| = r_i$ (the round in which p_i observes the revealing chain $m{:}\gamma_i$, cf. Alg. 3) occurs before or after the round $\mathsf{reveal\_round}_{\mathsf{GCL}}(w) \stackrel{\text{def}}{=} \mathsf{max}(2, t{+}3{-}w)$, the round during which we seek to prove that all correct processes that have not stopped earlier are aware of m. If $r_i < \mathsf{max}(2, t+3-w)$, because p_i forwards all chains it has not signed yet, all correct processes observe a chain containing m at the latest by round $r_i + 1 \leq \mathsf{max}(2, t+3-w)$. If $r_i \geq \mathsf{max}(2, t+3-w)$, the construction of $\mathsf{reveal\_round}_{\mathsf{GCL}}$ implies that at least $t+1$ processes have signed chains containing m during the first $\mathsf{max}(2, t+3-w)$ rounds of the protocol. One of them must be correct, yielding the lemma. □

Detailed proof. Assume a correct process p_i observes $\mathsf{WBP}_{\mathsf{GCL}}(m, w, view_i) = \texttt{true}$ during its execution. In the following, γ_i denotes a chain that renders $\mathsf{WBP}_{\mathsf{GCL}}$ true in Algorithm 3, and $r_i = |\gamma_i|$ its length. As p_i is correct, $m{:}\gamma_i \in view_i$ (line 2 of Algorithm 3) implies that p_i receives $m{:}\gamma_i$ during round $r_i = |\gamma_i|$. The remainder of the proof distinguishes two cases, depending on whether $r_i < \mathsf{reveal\_round}_{\mathsf{GCL}}(w) \stackrel{\text{def}}{=} \mathsf{max}(2, t+3-w)$ or not.

- Case $r_i < \mathsf{max}(2, t+3-w)$: As just mentioned, p_i receives $m{:}\gamma_i$ during the communication step of round r_i (line 5 of Algorithm 2). If $p_i \notin \gamma_i$, p_i signs the chain (line 6, Alg. 2), and broadcasts it during the communication step of round $r_i + 1 \leq \mathsf{max}(2, t+3-w)$ (line 3 of the same algorithm). If $p_i \in \gamma_i$, p_i has signed a chain $m{:}\gamma'$ earlier, and broadcast the result before or during round r_i.

  In both cases, any correct process p_j that executes round $\mathsf{max}(2, t+3-w)$ receives some chain $m{:}\gamma_i'{:}p_i$ either during or before round $\mathsf{max}(2, t+3-w)$, and therefore m is known to p_j at the latest by round $\mathsf{max}(2, t+3-w)$.

- Case $r_i \geq \mathsf{max}(2, t+3-w)$: Let us note S and W the set of processes which together with γ_i render $\mathsf{WBP}_{\mathsf{GCL}}(m, w, view_i)$ true in Algorithm 3.

  Let us note $\gamma_{3..t+3-w} \stackrel{\text{def}}{=} \mathsf{subchain}(\gamma_i, 3, t{+}3{-}w)$. Since $|\gamma_i| = r_i \geq \mathsf{max}(2, t{+}3{-}w) \geq t{+}3{-}w$, γ_i contains more than $t+3-w$ processes. As a result, by definition of the function subchain, and because γ_i is valid, and hence acyclic, $\gamma_{3..t+3-w}$ contains exactly $\mathsf{max}(0, (t+3-w)-3+1) = \mathsf{max}(0, t+1-w)$ distinct processes:

  $$\big|\mathsf{set}(\gamma_{3..t+3-w})\big| = \mathsf{max}(0, t+1-w). \tag{4}$$

  By construction, W and $\mathsf{set}(\gamma_{3..t+3-w})$ have no element in common, which yields

  $$\begin{aligned} |W \cup \mathsf{set}(\gamma_{3..t+3-w})| &= |W| + |\mathsf{set}(\gamma_{3..t+3-w})|, && (5)\\ &= |W| + \mathsf{max}(0, t+1-w), && \text{(using (4))}\\ &\geq \mathsf{max}(w, w+t+1-w), && \text{(by definition of } \mathsf{WBP}_{\mathsf{GCL}}\text{)}\\ |W \cup \mathsf{set}(\gamma_{3..t+3-w})| &\geq \mathsf{max}(w, t+1) \geq t+1. && (6) \end{aligned}$$

The set $W \cup \mathsf{set}(\gamma_{3..t+3-w})$ therefore contains at least one correct process p_k. If $p_k \in W$, p_k's signature appears in the first or second position of one of the chains of $view_i$. If $p_k \in \mathsf{set}(\gamma_{3..t+3-w})$, p_k's signature appears before position $t+3-w$ in $\gamma_i \in view_i$. Both cases imply that p_k has signed a chain with message m and has broadcast this chain to all processes that have not stopped earlier during or before round $\mathsf{max}(2, t+3-w)$, proving the lemma. □

In the following, we use the notation $view_i[r_1..r_2]$ the chains of $view_1$ that contain signatures between r_1 and r_2. In the same way $view_i[r]$ represents the chains received by p_i during round r, $view_i[r_1..r_2]$ contains the chains received between rounds r_1 and r_2:

$$view_i[r_1..r_2] \stackrel{\text{def}}{=} \{\gamma \in view_i : r_1 \leq |\gamma| \leq r_2\}.$$

Building upon the notation $view_i[r_1..r_2]$, we introduce the quantity $T_{2,i}[R]$ to prove the WBP-FINAL-VISIBILITY of $(\mathsf{WBP}_{\mathsf{GCL}}, \mathsf{reveal\_round}_{\mathsf{GCL}})$ (Lemma 13). $T_{2,i}[R]$ is defined using process p_i's view $view_i$ at round R as follows

$$T_{2,i}[R] \stackrel{\text{def}}{=} \mathsf{truncate}_2\big(view_i[2..R]\big). \tag{7}$$

$T_{2,i}[R]$ 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 at round R by a correct process $p_i \neq p_{\text{sender}}$ executing Algorithm 2 are known by all other correct processes by round $R+1$.

Lemma 11. *Let p_i and $p_j \neq p_i$ be two correct processes, such that p_i executes the computation step (lines 3-4) 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 $R = 1$: $view_i[2..1] = \emptyset$, 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 $R \in [1..t-1]$.

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

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 6 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 3 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

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

– 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

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

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 12. *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 overview. (Detailed proof below.) 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 11 yields the proof. □

Detailed proof. By definition

$$\begin{aligned}T_{2,i}[t+1] &= \mathsf{truncate}_2\left(view_i[2..t+1]\right)\\ &= \mathsf{truncate}_2\left(view_i[2..t] \cup view_i[t+1]\right),\\ &= T_{2,i}[t] \cup \mathsf{truncate}_2\big(view_i[t+1]\big).\end{aligned}$$

Applying Lemma 11 we have $T_{2,i}[t] \subseteq T_{2,j}[t+1]$, which with the previous equality yields

$$T_{2,i}[t+1] \subseteq T_{2,j}[t+1] \cup \mathsf{truncate}_2\big(view_i[t+1]\big). \tag{8}$$

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 $|\{p_{\text{sender}}\} \cup \mathsf{set}(\gamma)| = |\{p_{\text{sender}}\}| + |\mathsf{set}(\gamma)| = 1 + t$. So $\{p_{\text{sender}}\} \cup \mathsf{set}(\gamma)$ 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 6 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

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

This last equation shows that $\mathsf{truncate}_2(view_i[t+1]) \subseteq T_{2,j}[t+1]$, which injected in (8) 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. □

Corollary 2. *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 12 or 11, depending on whether $r = t + 1$ or not.

- If $r = t + 1$, the corollary follows trivially from Lemma 12.
- If $r < t + 1$, this follows from Lemma 11, and observing that $T_{2,j}[r + 1] \subseteq T_{2,j}[t + 1]$. □

Lemma 13. *The pair* $(\mathsf{WBP}_{\mathsf{GCL}}, \mathsf{reveal\_round}_{\mathsf{GCL}})$ *fulfills the* WBP-FINAL-VISIBILITY *property.*

Proof overview. (Detailed proof below.) Consider p_i, p_j two correct processes, and assume p_{sender} is Byzantine. The proof focuses on the sets of processes S and W, and on the revealing chain γ used in the predicate $\mathsf{WBP}_{\mathsf{GCL}}$ (Algorithm 3). First, the proof shows that the set S perceived by p_i is propagated to all other correct processes at the latest by round $t + 1$. More concretely, if p_i perceives a process p_k as backing m, then all correct processes also perceive p_k as backing m by round $t + 1$. The proof then shows that any revealing chain γ_i that renders $\mathsf{WBP}_{\mathsf{GCL}}$ `true` for p_i implies the existence of a revealing chain γ_j is p_j's view at round $t + 1$ that is no more "constraining" than γ_i. □

Detailed proof. Assume p_{sender} is Byzantine. Assume a correct process p_i observes $\mathsf{WBP}_{\mathsf{GCL}}(m, w, view_i) = \texttt{true}$ during some round R of its execution. 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, for clarity, we note $view_x^y$ the value of the $view$ variable in Algorithm 2 for the process p_x at round y (or more precisely, after $view$ has been updated in round y at line 5 of Algorithm 2).

Consider γ_i, $S_{i,R}$ and $W_{i,R}$ the chain and the sets of processes that render true $\mathsf{WBP}_{\mathsf{GCL}}$ for p_i in Algorithm 3 during round R.

By definition of $\mathsf{WBP}_{\mathsf{GCL}}$, $|W_{i,R}| \geq w$. Using $T_{2,i}[R]$ (7), we can express $S_{i,R}$ as follows

$$S_{i,R} = \{\mathsf{set}(\gamma) \mid m{:}\gamma \in view_i^R[1] \cup T_{2,i}[R]\}.$$

The set $view_i^R[1]$ contains the chains received by p_i in round 1, while $T_{2,i}[R]$ contains the length-2 prefixes of chains received by p_i in rounds 2 to R. If $m{:}\gamma \in view_i^R[1] \cup T_{2,i}[R]$, the processes whose signature appears in γ have therefore backed m in round 1 or 2 in $view_i^R$. Let us define $\gamma^i_{3..t+3-w} \stackrel{\text{def}}{=} \mathsf{subchain}(\gamma_i, 3, t + 3 - w)$. We can express $W_{i,R}$ as

$$\begin{aligned} W_{i,R} &= S_{i,R} \setminus \mathsf{set}\big(\gamma^i_{3..t+3-w}\big) \\ &= \big\{\mathsf{set}(\gamma) \mid m{:}\gamma \in view_i^R[1] \cup T_{2,i}[R]\big\} \setminus \mathsf{set}\big(\gamma^i_{3..t+3-w}\big). \end{aligned}$$

In the following, we will consider p_j's perception at round $t + 1$ of the processes that have backed m in round 1 or 2. The set of these processes can be defined as:

$$S_{j,t+1} = \{\mathsf{set}(\gamma) \mid m{:}\gamma \in view_j^{t+1}[1] \cup T_{2,j}[t + 1]\}.$$

Sublemma 13.1. $S_{i,R} \subseteq S_{j,t+1}$.

Proof. Consider $q \in S_{i,R}$. There exists a chain γ_q such that $q \in \gamma_q$ and $m{:}\gamma_q \in view_i^R[1] \cup T_{2,i}[R]$.

- If $m{:}\gamma_q \in T_{2,i}[R]$, by Corollary 2, $T_{2,i}[R] \subseteq T_{2,j}[t + 1]$, and $\gamma_q \in T_{2,j}[t + 1]$, which leads to $q \in S_{j,t+1}$.
- If $m{:}\gamma_q \in view_i^R[1]$, then because p_i is correct, $m{:}\gamma_q$ is a valid chain of length 1. As a result, $\gamma_q = (p_{\text{sender}})$ and $q = p_{\text{sender}}$. Since $p_i \neq p_{\text{sender}}$, p_i signs $m{:}p_{\text{sender}}$ at line 6 of Alg. 2, and broadcasts $m{:}p_{\text{sender}}{:}p_i$ to all correct processes in round 2. We therefore have $m{:}p_{\text{sender}}{:}p_i \in T_{2,j}[2] \subseteq T_{2,j}[t + 1]$ (assuming $t \geq 1$, and $t + 1 \geq 2$). As a result, $q = p_{\text{sender}} \in S_{j,t+1}$, which concludes the lemma. □

To conclude the proof, we need to find a chain $m{:}\gamma_j \in view_j^{t+1}$ that, when "removed" from $S_{j,t+1}$, produces a set $W_{j,t+1}$ containing at least w processes. Once this happens, $m{:}\gamma_j$, $S_{j,t+1}$ and $W_{j,t+1}$ will render $\mathsf{WBP}_{\mathsf{GCL}}(m,\, w,\, view_j^{t+1})$ true in round $t+1$. The existence of $m{:}\gamma_j$ comes from the following sublemma.

Sublemma 13.2. $\exists\, m{:}\gamma_j \in view_j^{t+1},\, \Big[\mathsf{set}\big(\mathsf{subchain}(\gamma_j, 3, t+3-w)\big) \cap S_{i,R}\Big] \subseteq \mathsf{set}(\gamma^i_{3..t+3-w})$.

Proof. The proof distinguishes three cases:

- Case 1: $p_i \in S_{i,R}$. Because $p_i \neq p_{\text{sender}}$ and p_i is correct, if $p_i \in S_{i,R}$, then p_i has signed a chain $m{:}p_{\text{sender}}{:}p_i$, and broadcast this chain in round 2 to all other correct processes, including p_j. As a result, $m{:}p_{\text{sender}}{:}p_i \in view_j^{t+1}$ (assuming $t \geq 1$).

  Furthermore, since $m{:}p_{\text{sender}}{:}p_i$ only contains 2 processes,

  $$\mathsf{set}\big(\mathsf{subchain}(p_{\text{sender}}{:}p_i, 3, t+3-w)\big) = \emptyset,$$

  which proves the sublemma.

- Case 2: $|\gamma_i| = t+1$. In this case, because γ_i is acyclic, it contains at least one correct process p_k. Since p_k is correct, it has signed and then broadcast a chain $m{:}\gamma_k{:}p_k$ to all processes including p_j at the latest by round $t+1$. As a result, $m{:}\gamma_k{:}p_k \in view_j^{t+1}$.

  Furthermore, by construction, $\gamma_k{:}p_k$ is a prefix of γ_i, which implies

  $$\mathsf{set}\big(\mathsf{subchain}(\gamma_k{:}p_k, 3, t+3-w)\big) \subseteq \mathsf{set}(\gamma^i_{3..t+3-w}),$$

  which proves the sublemma.

- Case 3: $p_i \notin S_{i,R} \wedge |\gamma_i| < t+1$. If $p_i \in \gamma_i$, then we can reuse the reasoning of Case 2 with $p_k = p_i$, which proves the sublemma. If $p_i \notin \gamma_i$, since $|\gamma_i| < t+1$, p_i has received γ_i in a round $r_i = |\gamma_i|$, and because it is correct, p_i has signed and broadcast the chain $m{:}\gamma_i{:}p_i$ in round $r_i + 1 \leq t+1$ to all processes, including p_j. As a result $m{:}\gamma_i{:}p_i \in view_j^{t+1}$.

  Furthermore

  $$\begin{aligned}\mathsf{set}\big(\mathsf{subchain}(\gamma_i{:}p_i, 3, t+3-w)\big) &\subseteq \mathsf{set}\big(\mathsf{subchain}(\gamma_i, 3, t+3-w)\big) \cup \{p_i\},\\ &\subseteq \mathsf{set}(\gamma^i_{3..t+3-w}) \cup \{p_i\}. \qquad \text{(by definition of } \gamma^i_{3..t+3-w})\end{aligned}$$

  As a result

  $$\begin{aligned}\mathsf{set}\big(\mathsf{subchain}(\gamma_i{:}p_i, 3, t+3-w)\big) \cap S_{i,R}, &\subseteq \big(\mathsf{set}(\gamma^i_{3..t+3-w}) \cap S_{i,R}\big) \cup \big(\{p_i\} \cap S_{i,R}\big),\\ &\subseteq \mathsf{set}(\gamma^i_{3..t+3-w}) \cap S_{i,R}.\\ &\qquad \text{(since } p_i \notin S_{i,R} \text{ by case assumption)}\end{aligned}$$

  This last inclusion concludes Case 3 and therefore the sublemma. □

In the following, we will note $m{:}\gamma_j$ the chain whose existence is given by Sublemma 13.2. Consider

$$W_{j,t+1} = S_{j,t+1} \setminus \mathsf{set}\big(\mathsf{subchain}(\gamma_j, 3, t+3-w)\big).$$

The following holds

$$\begin{aligned}W_{i,R} &= S_{i,R} \setminus \mathsf{set}\big(\gamma^i_{3..t+3-w}\big)\\ &\subseteq S_{i,R} \setminus \Big(\mathsf{set}\big(\mathsf{subchain}(\gamma_j, 3, t+3-w)\big) \cap S_{i,R}\Big) && \text{(using Sublemma 13.2)}\\ &\subseteq S_{i,R} \setminus \mathsf{set}\big(\mathsf{subchain}(\gamma_j, 3, t+3-w)\big)\\ &\subseteq S_{j,t+1} \setminus \mathsf{set}\big(\mathsf{subchain}(\gamma_j, 3, t+3-w)\big) && \text{(using Sublemma 13.1)}\\ &\subseteq W_{j,t+1}.\end{aligned}$$

Because $|W_{i,R}| \geq w$ by definition, this last inclusion yields $|W_{j,t+1}| \geq w$. This inequality shows that $\mathsf{WBP}_{\mathsf{GCL}}(m, w, view_j)$ is rendered true at p_j at round $t+1$ by using γ_j, $S_{j,t+1}$, and $W_{j,t+1}$ in Algorithm 3, thus proving the lemma. □

Lemma 14. *The pair* $(\mathsf{WBP}_{\mathsf{GCL}}, \mathsf{reveal\_round}_{\mathsf{GCL}})$ *fulfills the* WBP-GOOD-CASE-LIVENESS *property.*

Proof. When the initial sender, p_{sender}, is correct, and brb-broadcast a message m, all remaining correct processes $p_i \neq p_{\text{sender}}$ receive $m{:}p_{\text{sender}}$ in round 1, and broadcast $m{:}p_{\text{sender}}{:}p_i$ in round 2. As a result, every correct process $p_i \neq p_{\text{sender}}$ receives at least $c-1$ distinct chains backing m of the form $m{:}p_{\text{sender}}{:}p_k$ in round 2.

These chains imply that the set S at line 3 of Algorithm 3 contains at least c processes (p_{sender} and the remaining $c-1$ correct processes). Using any of the length-2 chains $m{:}p_{\text{sender}}{:}p_k$, a set W can be constructed equal to S, rendering $\mathsf{WBP}_{\mathsf{GCL}}(m, c, view_i)$ true for all correct processes other than p_{sender}.

The observation that $\mathsf{reveal\_round}_{\mathsf{GCL}}(c) = \mathsf{max}(2, t+3-c)$ concludes the lemma. □

6.2 Proof of Theorem 3 and Corollary 1

The proof of Theorem 3 follows from Lemmas 8, 9, 10, 13, and 14. The proof of Corollary 1 follows from Theorems 1 and 3.

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, namely 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 exhibits a good-case latency better than $t+1$ (the worst-case bound) under a majority of Byzantine processes. This paper 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 proposed algorithm has been presented as an instance of a *generic* BRB algorithm (from which the classical BRB algorithm from Lamport, Shostak and Pease [23] can also be derived). This generic algorithm extends the "signature chain mechanism" first proposed four decades ago. It exploits a family of weight-based predicates to identify patterns in sets of signature chains and help correct processes decide when they can safely deliver a message. A judicious choice for these patterns delivers a concrete BRB algorithm that allows correct processes to brb-deliver much earlier than earlier proposals when the context is favorable. In particular, when the sender is correct, and there are enough effectively correct processes ($c > t$), the resulting 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 $\mathsf{max}(2, t+3-c)$ rounds can be further improved (for instance, using techniques employed in sub-linear randomized algorithms [4, 30]). In terms of lower bounds, one might ask whether the lower bound of $\lfloor n/(n-t) \rfloor - 1$ 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 by observing 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].

Finally, this paper did not consider the problem of communication complexity. Signature chains can be quite costly in a practical implementation, as each new signature adds hundreds or even thousands

of bits to each network message. We conjecture that multi-signatures could help to significantly reduce this overhead by aggregating all non-backing signatures into a fixed-size structure [9]. Furthermore, an interesting follow-up would then be to study the tension between time complexity and communication complexity and how favoring one metric may hinder the other.

Declarations

Funding. This work was partially funded 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).

Competing Interests. The authors have no competing interests to declare that are relevant to the content of this article, apart from those possibly resulting from the funding sources mentioned above.

Data. Data sharing not applicable to this article as no datasets were generated or analyzed during the current study.

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, 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:2102.07240*, 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*, 132:22–43, 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] Dan Boneh, Ben Lynn, and Hovav Shacham. Short signatures from the Weil pairing. In *International Conference on the Theory and Application of Cryptology and Information Security*, pages 514–532. Springer, 2001.

[10] Dan Boneh, Ben Lynn, and Hovav Shacham. Short signatures from the weil pairing. In *7th International Conference on the Theory and Application of Cryptology and Information Security (ASIACRYPT 2001)*, pages 514–532. Springer, 2001.

[11] Gabriel Bracha. Asynchronous Byzantine agreement protocols. *Information & Computation*, 75(2):130–143, 1987.

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

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

[14] 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.

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

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

[17] 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.

[18] Matthias Fitzi and Jesper Buus Nielsen. On the number of synchronous rounds sufficient for authenticated Byzantine agreement. In *International Symposium on Distributed Computing (DISC)*, Springer LNCS 5805, pages 449–463, 2009.

[19] Davide Frey, Lucie Guillou, Michel Raynal, and François Taïani. Consensus-free ledgers when operations of distinct processes are commutative. In *16th International Conference on Parallel Computing Technologies (PaCT)*, Springer LNCS 12942, pages 359–370, 2021.

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

[21] 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.

[22] 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.

[23] 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.

[24] 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.

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

[26] 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.

[27] 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.

[28] Michel Raynal. *Fault-Tolerant Message-Passing Distributed Systems - An Algorithmic Approach*. Springer, 459 pages, 2018.

[29] 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)*, Springer LNCS 12550, pages 412–456, 2020.

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

A Communication Complexity

Because our algorithm focuses on latency, we have left aside communication complexity so far, but this would be the next aspect to consider. First off, it should be noted that the communication cost of a distributed algorithm can be approached from two perspectives, by considering two distinct metrics:

- Metric 1: The number of messages sent by correct processes,
- Metric 2: The size of individual messages sent by correct processes.

These two metrics only measure the communication cost of correct processes, as Byzantine processes can send a potentially infinite amount of messages with a potentially infinite amount of information. The two metrics can be combined in one single metric: the total amount of information sent in the network by correct processes (Metric 3). But to approach the problem in a more granular manner, it is useful to consider these 2 metrics separately. Moreover, in a practical implementation, it is often more desirable to reduce the number of messages (Metric 1) than the size of messages (Metric 2), as each new message often necessitates negotiating a new network connection (e.g., a TCP session between two endpoints) which entails a significant communication overhead. In other words, having a few large messages is often better than having a lot of small messages. In this respect, Metric 3 does not allow a fine-grained analysis of communication complexity.

In the following, "the algorithm" refers to Algorithm 2 in which we use the $\mathsf{WBP}_{\mathsf{GCL}}$ predicate described in Algorithm 3.

A.1 Metric 1: Number of messages

Currently, the algorithm follows the original BRB algorithm of Lamport, Shostak, and Pease and adopts a full-knowledge dissemination strategy: correct processes forward any signature chain they have not signed yet. This is not very efficient, and as a result, in each round, each correct process sends n messages (1 unreliable broadcast). For simplicity's sake, we will ignore the border case where a process cannot sign any new chain. This assumption leads to a worst-case collective message cost for correct processes of n messages in round 1, and $c \times n$ messages in each later round, which adds up to $n + c \times n \times R_{end}$ messages (at most), where R_{end} is the maximum number of rounds needed for the algorithm to brb-deliver messages at correct processes.[7] This upper bound can be further refined depending on whether the execution occurs in a good or bad case.

A.1.1 Good case (correct sender)

In a good case, we have $R_{end} = \mathsf{max}(2, t + 3 - c)$, but the initial sender (which is correct) only participates in the first round. These two observations lead to at most $n + (c-1) \times n \times \mathsf{max}(2, t+3-c)$ messages sent by correct processes, which boils down to $n + 2 \times n \times (n-1) = O(n^2)$ messages when all processes are correct ($c = n$, assuming $n > t$).

A.1.2 Bad case (Byzantine sender)

In a bad case, we have at worst $R_{end} = t + 1$. When $R_{end} = t + 1$, however, correct processes do not execute any extra round of communication (see Footnote 7). This, therefore, leads to at most $n + (c-1) \times n \times (t+1)$ messages sent by correct processes. If one assumes c and t are of the order of n, the overall message complexity is thus in $O(n^3)$.

[7] When $R_{end} \leq t$, Algorithm 2 brb-delivers in round R_{end}, but also executes one extra round of communication (through the $delivered_i$ variable). This extra round leads to a total of at most $R_{end} + 1$ rounds of communication: the first round followed by R_{end} subsequent rounds.

A.2 Metric 2: Size of messages

We capture the size of a message by counting how many "information items" it contains. Because the size of the fixed-size fields of a message (e.g., its application payload) becomes asymptotically negligible compared to its fields of variable size (e.g., a set of signatures), we equate this number of "information items" by the number of elements in the fields of variable size. In the case of our algorithm, this number of "information items" would be the number of signatures in all of the chains contained in a message.

In our algorithm, message size contributes heavily to communication costs. This is because messages keep growing in size at every round, as both the number of chains in each message and the length of each chain keep increasing. Let us, therefore, count the number of signatures exchanged in each round. A chain exchanged in round R must contain R signatures. A process p_i can have no more than C_{n-2}^{R-2} (the binomial coefficient "$n-2$ choose $R-2$") such chains that need to be disseminated (since these chains must start by p_{sender}'s signature, must end with p_i's signature, and are acyclic).

A.2.1 Good case (correct sender)

In a good case, in round $R \geq 2$, the chains of length R that a correct process needs to disseminate contain no more than $R \times C_{n-2}^{R-2}$ signatures, and correct processes collectively cannot send more than $(c-1) \times n \times R \times C_{n-2}^{R-2}$ signatures, which is upper bounded by $(c-1) \times n \times R \times (n-2)^{R-2}$. When all processes are correct ($c = n$, which implies that the algorithm terminates in 3 rounds: 2 for all processes to brb-deliver, plus one for broadcasting a final message, see Footnote 7), this adds up to no more than $n + 2n(n-1) + 3n(n-1)(n-2) = O(n^3)$ signatures.

A.2.2 Bad case (Byzantine sender)

In a bad case, given that the Byzantine sender may spam the system with an arbitrary number of application messages (the BRB's payload), with each message incurring its own cost in signatures, the signature cost is essentially unbounded.

A.3 Possible improvements

A.3.1 Sets instead of chains of signatures

The weight-based predicate $\mathsf{WBP}_{\mathsf{GCL}}$ (Algorithm 3, Section 5.2) does not use the order of the signatures in chains from position 3 to the end. Indeed, the predicate hinges on the cardinality of the set of signatures W (rather than some sequence of signatures), and the construction of W depends on the interplay of signatures present in the first two rounds and those present from round 3 and later. As a result, instead of a chain, every signature from position 3 to the end could be bundled in a set that does not preserve the order.

Using sets of signatures instead of chains would greatly reduce the size of the messages in good and bad cases (Metric 2), as currently, a single message that is exchanged in this algorithm can contain multiple chains that possess exactly the same signatures, but not in the same order, which creates a significant amount of redundant information in each message.

Moreover, using sets of signatures would enable the use of multisignature schemes (such as BLS [10]), which make it possible to aggregate multiple digital signatures in one single fixed-size structure, thus further reducing message sizes. An important caveat is that, even with multisignatures, the message must still contain the identity of all signatories to verify its authenticity. As a result, although the number of "authenticators" [2] is reduced, the number of "information items" remains the same.

A.3.2 Filtering received messages

To limit the size of the message when the sender is Byzantine, we conjecture that a filtering mechanism precluding multiple conflicting application messages from being backed by correct processes could help to obtain a finite value for Metric 2 in a bad case. This improvement may also lead to a decrease of Metric 1.

A.3.3 Not sending "useless" messages

There is a strong possibility that some messages of the algorithm that we present are not required to ensure its correctness. For instance, in our algorithm, a correct process signs and rebroadcasts every signature chain it receives that does not already contain its signature. However, the set of chains that it broadcasts may be a version with more signatures of an old set of chains that it has already broadcast in a previous round. As a result, broadcasting this new set of chains to the network may not decrease the revealing round for the message for any process, and not broadcasting this new "useless" message may very well have no consequence on the correct termination of the algorithm. This would reduce Metric 1 (the number of messages) in both good and bad cases.