Efficient Byzantine Reliable Broadcast in the Failure Case

Thomas Locher ✉
DFINITY, Zurich, Switzerland

Abstract

Reliable broadcast is a fundamental primitive in distributed computing that is widely used in various applications. Several new reliable broadcast algorithms have been presented in recent years, primarily focusing on reducing the communication complexity, which is the total number of exchanged bits in the worst case. While significant progress has been achieved, all proposed algorithms share a common weakness. Executions may fail, i.e., no message is ever delivered, while incurring a communication complexity equal or nearly equal to the communication complexity of executions where a message is delivered. In fact, a single Byzantine node, acting as the dedicated sender, is sufficient to trigger such executions, causing all nodes to consume bandwidth in vain.

This paper introduces the novel concept of a reliable broadcast *detector*, a distributed algorithm that can be coupled with a reliable broadcast algorithm to minimize the communication complexity of failed executions. Two concrete detectors are presented with different requirements and properties. Additionally, reliable broadcast algorithms that utilize detectors are introduced, the main algorithm guaranteeing an overhead factor, compared to an ideal failure-free execution, that tends to 2 as the network size increases. Furthermore, a lower bound is proven that an overhead factor of 5/3 is inevitable when the sender initially broadcasts the message, as is the case for the proposed algorithm. Therefore, it achieves a bound that is close to optimal for any algorithm with this property.

2012 ACM Subject Classification Computer systems organization → Fault-tolerant network topologies; Theory of computation → Distributed algorithms

Keywords and phrases asynchronous networks, reliable broadcast, communication complexity

Digital Object Identifier 10.4230/LIPIcs.OPODIS.2025.12

1 Introduction

Numerous applications require the dissemination of certain messages to all nodes in a distributed system. In contrast to a regular broadcast, where all nodes are only guaranteed to receive the message if the sender does not fail, a *reliable broadcast* provides stronger guarantees. In particular, only one message may be broadcast and either no *honest* (i.e., non-faulty) node receives it or all of them eventually do, despite the presence of *Byzantine nodes*, which may behave in arbitrary ways to interfere with the protocol execution. Applications of reliable broadcast include asynchronous atomic broadcast [14, 17, 19, 24, 28, 29, 31], distributed key generation [2, 11, 20], secure data replication [7], and secret sharing [30]. Reliable broadcast can further be used to improve throughput in Byzantine fault-tolerant consensus protocols that treat message dissemination and ordering separately [9].

In his seminal paper, Bracha showed that the reliable broadcast problem can be solved in an asynchronous network comprising n nodes, out of which at most $t < n/3$ nodes may be Byzantine [6]. The *communication complexity*, i.e., the number of bits that need to be exchanged in the worst case, of the proposed algorithm is $O(n^2|m|)$, where $|m|$ denotes the size of the broadcast message. Nearly two decades later, Cachin and Tessaro [8] presented the first reliable broadcast algorithm with an improved communication complexity of $3n|m| + O(\kappa n^2 \log(n))$ bits using *erasure coding*, where κ is the output size of a collision-resistant hash function, which is asymptotically optimal for large messages as every node must receive at least $|m|$ bits. Subsequently, several studies have focused on narrowing the gap to the lower bound of $\Omega(n|m| + n^2)$ [1, 3, 10, 26, 27].

While reliable broadcast algorithms with nearly optimal communication complexity are known, all proposed algorithms effectively exhibit the same complexity whether or not the broadcast is successful. In other words, the nodes may not receive any message as the outcome of the broadcast and yet send almost as many bits as in the worst-case execution where a message is delivered. As shown in §4, a Byzantine sender can cause such wasteful executions, even if all other nodes are honest. In any system where each node has an equal chance of becoming the next sender or where a simple round-robin scheme determines the next sender, up to a third of all consumed bandwidth can be wasted.

This problem is addressed in this paper. In §3, reliable broadcast *detectors* are introduced, which are distributed algorithms that serve to inform a reliable broadcast algorithm whether or not it is safe to start executing the main routine, with the guarantee that all honest nodes will eventually obtain one and the same message. Apart from specifying the requirements that such a detector must meet, two concrete detectors are presented. The second detector has the advantage that the condition to trigger the main reliable broadcast algorithm is reached faster when the sender is honest but requires a more complex setup due to its use of threshold cryptography. The communication complexity of both detectors is $O(\kappa n^2)$, where κ is a security parameter of constant size, and thus independent of the message size.

Reliable broadcast algorithms that make use of detectors are provided in §4. The first algorithm showcases the simplicity of defining a reliable broadcast algorithm when given a detector, which demonstrates the power of this abstraction. The second and main algorithm aims to minimize the communication complexity when a message is delivered, while keeping the communication complexity independent of the message size when no message is delivered. In the former case, the overhead factor is only $2 + O(1/n)$, i.e., roughly $2n|m|$ bits are sent in total for large n and $|m|$. This bound is slightly worse than the best known bound of $3/2$, but the algorithm is significantly more efficient in the failure case. We argue that it is quite natural for any algorithm optimizing for the failure case to first have the sender broadcast the message. For any such algorithm, at least $5/3n|m|$ bits must be exchanged in total as shown in Appendix A.2, which implies that the proposed algorithm is nearly optimal. Related work on reliable broadcast is summarized in §5 and §6 concludes the paper.

2 Model

We consider a network comprising $n \geq 3t + 1$ nodes, where t denotes the maximum number of *Byzantine nodes*, which may exhibit arbitrary faulty behavior, ranging from simple crash failures to malicious behavior including collusion. By contrast, the other $n-t$ nodes, which we call *honest*, always execute any given protocol correctly and never fail. The set of all n nodes is denoted by V. Any pair of nodes can communicate directly by exchanging messages over an authenticated channel. Throughout this paper, we consider the *asynchronous communication model*, where all messages are guaranteed to be received but after an unbounded delay, i.e., nodes cannot expect that a sent message will arrive, nor can they expect to receive a certain message, after a certain amount of time. When a node v sends a message m to all nodes, including itself, we say that node v *broadcasts* message m. If v is honest, all nodes will eventually receive the broadcast message. However, a Byzantine node may send a message that is to be broadcast only to a subset of the nodes, possibly not sending the message at all.

A *reliable broadcast* protocol ensures that either all honest nodes eventually *deliver* the sender's message, i.e., accept the message as the outcome of the reliable broadcast, or no honest node delivers it. Formally, reliable broadcast is defined as follows.

▶ **Definition 1** (Reliable broadcast)**.** *A* reliable broadcast protocol *is a distributed protocol to send a message m from a specific node called the* sender *to all nodes with the following properties.*

- ***Validity****: If the sender is honest and broadcasts m, then every honest node eventually delivers m.*
- ***Agreement****: If two honest node deliver m and m', then $m = m'$.*
- ***Integrity****: Every honest node delivers at most one message.*
- ***Totality****: If an honest node delivers a message, then all honest nodes eventually deliver a message.*

Naturally, there can be multiple instances of reliable broadcast executions running in parallel, each with a specific sender that is known a priori to all nodes. We assume that every message carries information that identifies a particular instance but do not explicitly specify such unique identifiers as we focus on a single execution.

The sender has a message m that it aims to disseminate to all nodes. The size of m in bits is denoted by $|m|$. In addition to the message itself, nodes can send other pieces of data such as hashes, signatures, and the aforementioned identifiers. The used hash functions and signature schemes are assumed to be *cryptographically strong* in that it is infeasible to find hash collisions or spoof signatures, except with negligible probability. Since hashes and signatures are considered secure for a certain minimum length, we introduce the security parameter κ, upper bounding the size of hashes and signatures by $O(\kappa)$.

State-of-the-art reliable broadcast algorithms make extensive use of *erasure codes*, which can handle missing data, or *error-correcting codes*, which can correct erroneous data. The novel algorithm introduced in §4.2 employs (n, k)-erasure codes with parameters $k < n$ defined in that section.

An *execution* $\mathcal{E}$ of a reliable broadcast algorithm is characterized by the send and receive events occurring at all nodes. If the execution results in a message being delivered, we call the execution *successful*. On the contrary, an execution where no message is delivered is called a *failed* execution. The validity property implies that all executions where the sender is honest must be successful. In other words, an execution may only fail if the sender is Byzantine. A Byzantine sender, together with other Byzantine nodes, may also prevent the delivery of a message for some time and then transmit more messages which will cause the honest nodes to deliver a message after all. Formally, we can define the *point of guaranteed eventual delivery* as the inflection point τ^* for which it holds that there must eventually be an honest node that delivers a message at some time $\tau \geq \tau^*$ but delivery is not guaranteed at any time $\tau < \tau^*$. If there is such a point in time τ^*, the agreement and totality property together imply that all honest nodes will eventually deliver the same message. Thus, a successful (failed) execution $\mathcal{E}$ is an execution for which it holds that $\tau^* < \infty$ ($\tau^* = \infty$). As argued before, the validity property implies that for any execution $\mathcal{E}$ with an honest sender starting at time τ_0 it must hold that $\tau_0 = \tau^*$.

The primary complexity measure for reliable broadcast algorithms is their *communication complexity*, which is the maximum number of bits exchanged by all honest nodes together over all possible executions. When defining $\mathcal{C}^{\mathcal{E}}(n, \ell)$ as the communication complexity in a network of n nodes, disseminating a message of size ℓ in execution $\mathcal{E}$ when running algorithm $\mathcal{A}$, the communication complexity of algorithm $\mathcal{A}$ is $\mathcal{C}(n, \ell) := \max_{\mathcal{E} \in \mathfrak{E}} \mathcal{C}^{\mathcal{E}}(n, \ell)$, where $\mathfrak{E}$ denotes the set of all possible executions. If a Byzantine sender happens to send messages with different sizes – or parts of messages with different sizes – in some execution, we define ℓ as the maximum size of all sent messages.

Since all nodes must receive the sender's message in a successful execution, at least $n\ell$ bits must be transmitted overall.[1] The overhead factor of a reliable broadcast algorithm is defined as $\mathcal{L}(n) \coloneqq \lim_{\ell\to\infty} \frac{\mathcal{C}(n,\ell)}{\ell n}$. The goal is to get $\mathcal{L}(n)$ as close as possible to 1. However, we distinguish between successful and failed executions: While $\mathcal{L}^{\mathcal{E}}(n) \approx 1$ is the best possible result for any successful execution $\mathcal{E}$, it should hold that $\mathcal{L}^{\bar{\mathcal{E}}}(n) = 0$ for any failed execution $\bar{\mathcal{E}}$ because no message is delivered at all. Rather than specifying $\mathcal{L}^{\mathcal{E}}(n)$ with respect to an execution $\mathcal{E}$ that may be successful or failed, we slightly abuse our notation and simply state the goal as minimizing $\mathcal{L}(n)$ for successful executions while ensuring that $\mathcal{L}(n) = 0$ for all failed executions. Note that for any reliable broadcast algorithm that meets these requirements it holds that $\mathcal{L}^{\mathcal{E}}(n) = 0$ at any time $\tau < \tau^*$ for any execution $\mathcal{E}$.

The second relevant complexity measure is an algorithm's *time complexity*, defined as the maximum duration of any execution, i.e., the time between the start of the execution and the time when the last honest node delivers the message, normalizing the maximum message delay to 1 and defining that all local computation takes 0 time. As an execution may never terminate given a Byzantine sender, we specify the time complexity for the good case with an honest sender. It is easy to see that honest nodes also engage in only a few rounds of communication for successful executions with a Byzantine sender for all considered algorithms.

3 Detectors

3.1 Definition

Ensuring that $\mathcal{L}^{\mathcal{E}}(n) = 0$ at any time $\tau < \tau^*$ in any execution $\mathcal{E}$ requires a mechanism to detect whether the point τ^* has been reached, in which case the nodes can start disseminating the message (or parts thereof). We introduce the notion of a reliable broadcast *detector* $\mathcal{D}$, which is itself a distributed algorithm. A detector $\mathcal{D}$ offers the following functionality:

- $\mathcal{D}$.`ready()` returns *true* if the point τ^* of guaranteed eventual delivery has been reached for a unique message m.
- $\mathcal{D}$.`validators()` returns the set of all nodes that claim to have successfully validated message m.

Any detector only signals readiness for a message from the designated sender. If $\mathcal{D}$.`ready()` returns *true*, we say that $\mathcal{D}$ *signals readiness for message* m to the reliable broadcast algorithm. Note that $\mathcal{D}$ may signal readiness even if only a *unique identifier* of message m is locally available. The constructions of detectors in the subsequent sections show that it is possible to deduce locally that the point of guaranteed eventual delivery has been reached for message m when receiving sufficiently many confirmations that a certain number of nodes have validated m. As shown in §4, the set $\mathcal{D}$.`validators()` of nodes that claim to have validated the sender's message is useful to reduce the communication complexity because it is not necessary to send any parts of the message to nodes in this set.

The following properties must hold in the asynchronous communication model with $t < n/3$ Byzantine nodes as defined in §2. The first property ensures that honest nodes eventually agree that the point of guaranteed eventual delivery has been reached.

▶ **Property 1.** *If $\mathcal{D}$ signals readiness at honest node v, then $\mathcal{D}$ eventually signals readiness at any honest node w.*

[1] Technically, $(n-1)\ell$ bits suffice, not counting the sender itself, but the relative difference tends to zero as $n \to \infty$.

It is further important that there is agreement as to which message may be broadcast in order to avoid wasteful transmissions.

▶ **Property 2.** *If $\mathcal{D}$ signals readiness at honest nodes v and v' for messages m and m', respectively, then $m = m'$.*

We say that a detector is *correct* if it satisfies both properties. Given the lower bound of $\Omega(n^2)$ on the communication complexity of reliable broadcast [13], it is desirable to guarantee an asymptotically tight upper bound on the communication complexity of detectors.

▶ **Property 3.** *Given the security parameter κ, the communication complexity of an* efficient *detector $\mathcal{D}$ is $C(n, \ell) \in O(\kappa n^2)$.*

Note that the communication complexity of an efficient detector is independent of the message size ℓ. Naturally, a detector $\mathcal{D}$ can only operate meaningfully in conjunction with a reliable broadcast algorithm $\mathcal{A}$ with certain properties. The following two properties specify the required interplay between the two components.

▶ **Property 4.** *If the honest sender executes algorithm $\mathcal{A}$, then detector $\mathcal{D}$ will eventually signal readiness at some honest node.*

Property 4 and Property 1 together imply that all honest nodes will eventually reach the same state. Since the honest sender first broadcasts m when executing the algorithms in §4, the following property implies Property 4 for the considered algorithms and detectors.

▶ **Property 5.** *If the honest sender executes algorithm $\mathcal{A}$, broadcasting m in the first step, then detector $\mathcal{D}$ will eventually signal readiness for message m at some honest node.*

The last property ties the actions of $\mathcal{A}$ to the output of $\mathcal{D}$ in the sense that readiness implies delivery and vice versa.

▶ **Property 6.** *Any honest node executing algorithm $\mathcal{A}$ eventually delivers m if and only if detector $\mathcal{D}$ signals readiness for message m.*

Under the assumption that these properties hold for algorithm $\mathcal{A}$ and detector $\mathcal{D}$, we get the following result.

▶ **Theorem 2.** *If algorithm $\mathcal{A}$ and detector $\mathcal{D}$ have Properties 1-6, then algorithm $\mathcal{A}$ implements reliable broadcast in the asynchronous communication model with $t < n/3$ Byzantine nodes.*

Proof. Each property of reliable broadcast is proved separately.

Validity: Property 4 implies that detector $\mathcal{D}$ will eventually signal readiness at the some honest node v. The detectors of all honest nodes will eventually signal readiness due to Property 1. Finally, Property 6 implies that all honest nodes will deliver the message.

Agreement: If an honest node v delivers m, its detector must have signaled readiness for m due to Property 6. Property 2 states that any other honest node w can only signal readiness for m as well, in which case w delivers m eventually due to Property 6.

Integrity: Since readiness can only be signaled for one message due to Property 2 and a message can only be delivered if readiness is signaled for this message according to Property 6, at most one message can be delivered.

Totality: If honest node v delivers m, its detector must have signaled readiness for m before due to Property 6. Property 1 then implies that detectors at every honest node will eventually signal readiness, causing the delivery of m due to Property 6. ◀

Thus, it suffices to show that algorithms and detectors have these properties to ensure that they implement reliable broadcast. Let a *detector-triggered algorithm* be an algorithm that only sends messages when the detector signals readiness, apart from the sender, which may send some data to other nodes to initiate the execution. The following result holds for such algorithms when paired with an efficient detector.

▶ **Lemma 3.** *For any* detector-triggered algorithm $\mathcal{A}$ *using an efficient detector it holds that* $\mathcal{L}(n) = 0$ *for failed executions.*

Proof. In any failed execution, the sender must be Byzantine and therefore its messages do not contribute to the communication complexity. By definition, no messages are sent when executing algorithm $\mathcal{A}$ as the detectors never signal readiness. Since the communication complexity of an efficient detector is $O(\kappa n^2)$ due to Property 3, the result follows. ◀

3.2 Basic Construction

There is a straightforward construction of a basic detector, denoted by $\mathcal{D}_b$, that shares similarities with Bracha's reliable broadcast algorithm itself [6]. The designated sender is supposed to broadcast message m. When receiving the first message m from the sender, the detector $\mathcal{D}_b$ verifies the validity of the message and then broadcasts an `attest` message containing $h = \mathtt{H}(m)$, a hash value of size $O(\kappa)$ derived using a hash function `H`. The reliable broadcast algorithm determines the concrete hash function to be used.

The first `attest(`h`)` message received from w is processed by adding w to the map $A : \mathcal{H} \to 2^V$, which stores the set of nodes from which `attest` messages have been received for a certain hash. The set $\mathcal{H}$ denotes the set of all possible hashes. Once at least $n-t$ `attest` messages for a certain hash h have been obtained, `confirm(`h`)` is broadcast, confirming that sufficiently many `attest` messages have been received. Information about such confirmations are collected in the map $C : \mathcal{H} \to 2^V$, storing the set of nodes from which `confirm` messages have been received for a certain hash. Again, only the first such message is processed from any node in order to bound the space complexity. A node v also broadcasts `confirm(`h`)` after receiving at least $n - 2t \geq t + 1$ `confirm` messages for h, i.e., v broadcasts a confirmation when assured that at least one honest node must have broadcast a `confirm` message.

Let h_{max} denote the hash for which the most confirmations have been obtained, i.e., $h_{max} \coloneqq \arg\max_{h \in \mathcal{H}} |C(h)|$, breaking ties arbitrarily. The detector $\mathcal{D}_b$ signals readiness when it has received at least $n - 2t$ `attest` messages and at least $n - t$ `confirm` messages for h_{max}. The set of validators returned when calling $\mathcal{D}_b$`.validators()` is simply the set $A(h_{max})$. The entire algorithm of detector $\mathcal{D}_b$ is given in Algorithm 1. We now proceed to show the correctness of $\mathcal{D}_b$.

▶ **Lemma 4.** *All* `confirm` *messages from honest nodes contain the same hash* h*.*

Proof. Let v be the first honest node to broadcast a `confirm` message for some hash h. Since it is the first such message, v must have received $n - t$ `attest` messages containing h. Let v' be the first honest node to broadcast a `confirm` message for some hash $h' \neq h$. As v' is the first such node, it cannot have sent it because it received $n - 2t \geq t + 1$ `confirm` messages containing h'. Hence it follows that it must have received $n - t$ `attest` messages containing h'. However, this implies that an honest node must have sent two (conflicting) `attest` messages, a contradiction. ◀

Note that the lemma implies that any honest node sends at most one `confirm` message. The following result states that detector $\mathcal{D}_b$ is both correct and efficient.

Algorithm 1 Detector $\mathcal{D}_b$ at node v given hash function $\mathtt{H}$. Initially, $A = C = \{\}$. $h_{max} \coloneqq \arg\max_{h \in \mathcal{H}} |C(h)|$.

if first m received from *sender* **and** `is_valid(`m`)` **then**
  broadcast `attest(`$\mathtt{H}(m)$`)`

if first `attest(`h`)` received from w **then**
  $A(h) \coloneqq A(h) \cup \{w\}$

if first `confirm(`h`)` received from w **then**
  $C(h) \coloneqq C(h) \cup \{w\}$

if $(|A(h_{max})| \geq n - t$ **or** $|C(h_{max})| \geq n - 2t)$ **and not** sent `confirm(`h_{max}`)` **then**
  broadcast `confirm(`h_{max}`)`

function READY()
  return $|A(h_{max})| \geq n - 2t$ **and** $|C(h_{max})| \geq n - t$

function VALIDATORS()
  return $A(h_{max})$

▶ **Lemma 5.** *Detector $\mathcal{D}_b$ is a correct and efficient detector.*

Proof. The two correctness properties and the efficiency property are proved separately.

Property 1: Assume that $\mathcal{D}_b$ signals readiness at node v for some message m with hash $h = \mathtt{H}(m)$. Let w be the first honest node to send a `confirm` message containing h. As it is the first such message, w must have received $n - t$ `attest(`h`)` messages. At least $n - 2t$ of these messages must have come from honest nodes having broadcast `attest(`h`)`. Thus, $|A(h)| \geq n - 2t$ will eventually hold at all honest nodes. Node v received at least $n - t$ `confirm` messages, which implies that all honest nodes will eventually receive at least $n - 2t$ `confirm` messages from honest nodes. Since honest nodes only send one `confirm` message, all containing h due to Lemma 4, it must eventually hold that $|C(h)| \geq n - 2t$. Hence it follows that all honest nodes broadcast `confirm(`h`)` and it eventually holds that $|C(h)| \geq n - t$ at all honest nodes.

Property 2: If readiness is signaled at node v and v' for messages m and $m' \neq m$, respectively, it must hold that $|C(\mathtt{H}(m))| \geq n - t$ at node v and $|C(\mathtt{H}(m'))| \geq n - t$ at node v'. However, these inequalities imply that there must be an honest node that has sent two different `confirm` messages, a contradiction to Lemma 4.

Property 3: Each honest node broadcasts at most one `attest` message by definition and at most one `confirm` message due to Lemma 4, each of size $O(\kappa)$, i.e., $O(\kappa n^2)$ bits are sent in total. ◀

The next lemma states that detector $\mathcal{D}_b$ eventually signals readiness given an honest sender that initially broadcasts the message.

▶ **Lemma 6.** *Detector $\mathcal{D}_b$ satisfies Property 5.*

Proof. Since the honest sender broadcasts m, all honest nodes will eventually broadcast `attest(`h`)` for $h \coloneqq \mathtt{H}(m)$. Consequently, all honest nodes will eventually receive at least $n - t$ `attest(`h`)` messages and broadcast `confirm(`h`)`. When $n - t$ `confirm(`h`)` messages have been received, the readiness condition is reached. ◀

As far as time complexity is concerned, if the honest sender broadcasts the message at the start, it takes 3 rounds of communication until readiness is signaled at all honest nodes.

▶ **Theorem 7.** *If the honest sender broadcasts m, the time complexity until $\mathcal{D}_b$ signals readiness for message m is* 3*.*

Proof. All honest nodes receive m after at most 1 time unit, causing them to broadcast the corresponding `attest` message. After at most 2 time units, $|A(\mathtt{H}(m))| \geq n - t$ holds at all honest nodes, triggering the transmission of `confirm` messages. Thus, after at most 3 time units, $|C(\mathtt{H}(m))| \geq n - t$ holds at all honest nodes and $\mathcal{D}_b$ signals readiness. ◀

3.3 Construction with Threshold Signatures

In this section, an alternative construction based on threshold cryptography is presented. Every node must be equipped with a *private key share* that enables it to *threshold-sign* a message, i.e., it can generate a *signature share* that can be combined with sufficiently many other signature shares to obtain a signature that is valid under some globally known public key. Specifically, our setup requires $n - t$ signature shares to construct a signature using the function `compute_signature`. It is possible for the nodes to verify the validity of both signatures and signature shares. The key idea is for each node to attach a signature share to its `attest` message. If a node receives a signature, derived from $n - t$ valid signature shares, the signature acts as proof that at least $n - 2t \geq t + 1$ honest nodes have validated the message. As a result, there is no need for `confirm` messages, which effectively serve to convince the nodes that at least one honest node attests to the validity of the message.

The detector based on threshold signatures, denoted by $\mathcal{D}_t$, is given in Algorithm 2. Every node v locally stores the map $A : \mathcal{H} \to 2^{\mathcal{S}}$, where $\mathcal{S}$ is the set of all possible signatures, plus a signature σ and a hash h_σ, both uninitialized (i.e., set to $\perp$) at first.

After validating the sender's message m, node v computes $h \coloneqq \mathtt{H}(m)$, which it then threshold-signs to produce a signature share σ_v. The hash h is broadcast together with the threshold signature σ_v in the `attest` message. When receiving an `attest`(h, σ_w) message, the threshold signature σ_w is validated and then added to the set $A(h)$. Given $n - t$ signature shares for some hash h, σ and h_σ are set to the computed signature and the corresponding hash, respectively. The signature σ, once set, is broadcast in a `sig` message, which contains h_σ as well. Alternatively, σ can be set when a `sig` message with a valid signature is received.

The function `ready()` is adapted to return *true* if and only if $|A(h_\sigma)| \geq n - 2t$ and a signature has been set, i.e., $\sigma \neq \perp$. It is important to note that $h_{max} \coloneqq \arg\max_{h \in \mathcal{H}} |A(h)|$ cannot be used because an honest node may receive more `assert` messages for a hash $h \neq h_\sigma$ when the Byzantine sender and other Byzantine nodes equivocate. For the same reason, the function `validators()` returns the set $A(h_\sigma)$, guaranteeing that the validators are returned from which `assert` messages containing h_σ have been received.

It remains to prove that detector $\mathcal{D}_t$ has the necessary properties as well. Lemma 8 is useful in those proofs, stating that the hash for which a signature is derived is unique.

▶ **Lemma 8.** *All broadcast* `sig` *messages contain a signature for the same hash.*

Proof. Every honest node only broadcasts a signature share for one message from the sender. Since $n - t$ signature shares are required to generate a signature, there cannot be enough signature shares for two distinct signatures as any two sets of $n - t$ signature shares intersect in at least $n - 2t \geq t + 1$ nodes. ◀

The lemma is used to prove the following result.

Algorithm 2 Detector $\mathcal{D}_t$ at node v given hash function $\mathtt{H}$. Initially, $A = \{\}$, $\sigma = h_\sigma = \bot$. $h_{max} \coloneqq \arg\max_{h \in \mathcal{H}} |A(h)|$.

if first m received from *sender* **and** `is_valid`(m) **then**
  $h \coloneqq \mathtt{H}(m)$
  $\sigma_v \coloneqq$ `threshold_sign`(h)
  broadcast `attest`(h, σ_v)

if first `attest`(h, σ_w) received from w **and** `is_valid`(h, σ_w) **then**
  $A(h) \coloneqq A(h) \cup \{\sigma_w\}$

if $|A(h_{max})| \geq n - t$ **and** $\sigma = \bot$ **then**
  $\sigma \coloneqq$ `compute_signature`$(A(h_{max}))$
  $h_\sigma \coloneqq h_{max}$

if `sig`(h', σ') received **and** $\sigma = \bot$ **and** `is_valid`(h', σ') **then**
  $\sigma \coloneqq \sigma'$
  $h_\sigma \coloneqq h'$

if $\sigma \neq \bot$ not sent **then**
  broadcast `sig`(h_σ, σ)

function READY()
  return $|A(h_\sigma)| \geq n - 2t$ **and** $\sigma \neq \bot$

function VALIDATORS()
  return $A(h_\sigma)$

▶ **Lemma 9.** *Detector $\mathcal{D}_t$ is a correct and efficient detector.*

Proof. As before, each property is proved separately.

Property 1: Assume that $\mathcal{D}_t$ signals readiness at honest node v, i.e., $|A| \geq n - 2t$ and $\sigma \neq \bot$. Since v broadcasts σ, it eventually holds at all nodes that $\sigma \neq \bot$. Due to Lemma 8, there can only be one such signature. Moreover, $\sigma \neq \bot$ implies that some node must have received $n - t$ valid signature shares to compute σ. Since at least $n - 2t$ of these signature shares must have come from honest nodes, which all broadcast `attest` messages, it eventually holds that $|A| \geq n - 2t$ at all honest nodes.

Property 2: The property follows from Lemma 8.

Property 3: Each honest node broadcasts at most one `attest` and `sig` message of size $O(\kappa)$ for a total of $O(\kappa n^2)$ bits. ◀

If the honest sender starts the execution by broadcasting the message, detector $\mathcal{D}_t$ signals readiness eventually.

▶ **Lemma 10.** *Detector $\mathcal{D}_t$ satisfies Property 5.*

Proof. All honest nodes broadcast an `attest` message and thus eventually receive at least $n - t$ signature shares. Subsequently, they compute the signature unless it is received in a `sig` message. In either case, the readiness condition is reached. ◀

Lastly, if we assume again that the honest sender initiates the execution by broadcasting the message, then 2 rounds of communication suffice for all honest nodes to signal readiness.

▶ **Theorem 11.** *If the honest sender broadcasts m, the time complexity until $\mathcal{D}_t$ signals readiness for message m is* 2*.*

Proof. Every honest node receives m after 1 time unit. Subsequently, every honest node broadcasts its `attest` message, which is received after 2 time units. Since all honest nodes have at least $n - t$ signature shares, they compute a signature σ, unless they received a signature earlier, which must be a signature for the same hash due to Lemma 8. Thus, after 2 time units, $|A(h_\sigma)| \geq n - t$ and $\sigma \neq \perp$ holds at all honest nodes. ◀

The primary advantage of $\mathcal{D}_t$ compared to $\mathcal{D}_b$ is its lower time complexity at the expense of a more complex setup with private key shares securely deployed on all nodes.

4 Algorithms

4.1 Simple Algorithm

In Bracha's reliable broadcast algorithm [6], every node broadcasts the message m (at most) twice in a successful execution. Including the sender's initial broadcast to all other nodes, the communication complexity is $(n-1)|m| + 2n(n-1)|m| \approx 2n^2|m|$.

There are *failed executions* where the communication complexity is nearly as large, even if only the sender is Byzantine: Consider the scenario where the Byzantine sender sends message m to $2t$ nodes and message $m' \neq m$ to the remaining t nodes. The honest nodes broadcast an `echo` message containing the message received from the sender. However, the Byzantine sender merely sends an `echo` message containing m to t other nodes, i.e., there are t nodes that receive $2t + 1$ `echo` messages for m and hence broadcast `ready` messages containing m. Since honest nodes solely receive t `ready` messages, none of them delivers m and the execution does not terminate. Assuming that $|m| = |m'|$, the communication complexity of this execution is $(n-1)^2|m| + t(n-1)|m| \approx \frac{4}{3}n^2|m|$.

Armed with a detector such as $\mathcal{D}_b$ and $\mathcal{D}_t$ introduced in §3, a simple reliable broadcast algorithm, denoted by $\mathcal{A}_s$, can be constructed easily, illustrating the effectiveness of the detector abstraction. In this algorithm, the sender first broadcasts m. If $\mathcal{D}$ signals readiness for m and m is locally available, m is forwarded to the nodes not in $\mathcal{D}$.`validators()` before delivering m. The nodes cache received messages until the execution terminates. The space complexity can be bounded by only caching at most one message per node.[2] The detector can use any cryptographically strong hash function `H`. Algorithm $\mathcal{A}_s$, given in Algorithm 3, is a correct reliable broadcast algorithm with the desired properties.

▶ **Theorem 12.** *Algorithm $\mathcal{A}_s$ using detector $\mathcal{D}_b$ or $\mathcal{D}_t$ implements reliable broadcast in the asynchronous communication model with $t < n/3$ Byzantine nodes.*

Proof. Since an honest sender broadcasts m as the first step, Property 5 holds when pairing $\mathcal{A}_s$ with either detector. Clearly, m is only delivered when readiness is signaled. On the other hand, readiness implies that some honest nodes have message m, which they transmit to all nodes not in $\mathcal{D}$.`validators()` when ready, leading to the eventual delivery of m at all nodes. Thus, Property 6 holds as well. ◀

[2] In practice, a message may only be cached if it carries the dedicated sender's signature. In this case, equivocation can be detected and penalized.

Algorithm 3 Algorithm $\mathcal{A}_s$ using detector $\mathcal{D}$ executed at node v.

if `reliable_broadcast(`m`)` invoked **and** $v = sender$ **then**
  broadcast m

▷ Any received message is cached.

if $\mathcal{D}$`.ready()` **and** (the corresponding) $m \neq \perp$ **and not** delivered m **then**
  for $w \in V \setminus \mathcal{D}$`.validators()` **do**
    send m to w
  `deliver(`m`)`

In comparison to Bracha's algorithm, the communication complexity of $\mathcal{A}_s$ is lower by a factor of 3. Moreover, $\mathcal{A}_s$ is clearly superior in the case of failed executions.

▶ **Theorem 13** (Communication complexity). *It holds for algorithm $\mathcal{A}_s$ using detector $\mathcal{D}_b$ or $\mathcal{D}_t$ that $\mathcal{L}(n) < \frac{2}{3}n$ for successful executions and $\mathcal{L}(n) = 0$ for failed executions.*

Proof. If the execution is successful, an honest sender's initial broadcast of m contributes $(n-1)|m|$ bits. Every honest node sends m to all nodes not in the set $\mathcal{D}$`.validators()` of cardinality at least $n - 2t$, resulting in a communication complexity of

$$(n-1)|m| + 2t(n-1)|m| + O(\kappa n^2) < \frac{2}{3}n^2|m| + O(\kappa n^2)$$

and thus $L(n) < \frac{2}{3}n$.

Since algorithm $\mathcal{A}_s$ is a detector-triggered algorithm and both $\mathcal{D}_b$ or $\mathcal{D}_t$ are efficient, $\mathcal{L}(n) = 0$ holds for failed executions due to Lemma 3. ◀

Lastly, $\mathcal{A}_s$ paired with $\mathcal{D}_t$ achieves a lower time complexity than Bracha's algorithm.

▶ **Theorem 14** (Time complexity). *If the sender is honest, then the time complexity of $\mathcal{A}_s$ corresponds to the time complexity of the used detector $\mathcal{D}_b$ or $\mathcal{D}_t$.*

Proof. All honest nodes receive m after at most 1 time unit if the sender is honest. Since the time complexity of both detectors is greater than 1 and $\mathcal{A}_s$ delivers m when m is available and readiness is signaled without waiting for any further messages, the claim follows. ◀

4.2 Coding-Based Algorithm

Algorithm $\mathcal{A}_s$ has a communication complexity that grows quadratically with the number of nodes, which is far from optimal. The goal is to devise a reliable broadcast algorithm that has a communication complexity in the order of $n|m|$ for successful executions but wastes little bandwidth in the case of failed executions.

All known reliable broadcast algorithms with a communication complexity of $O(n|m|)$ make use of erasure (or error-correcting) codes, disseminating *fragments* of m of size inversely proportional to n. While these algorithms are more efficient, it is usually trivial for a Byzantine sender to trigger executions that are bound to fail but cause as many bits to be sent as in successful executions. The problem is that, given a fragment alone, it cannot be determined if it is part of a valid encoding. Merkle proofs can be used, e.g., to efficiently check if a received fragment is consistent with some root hash but that does not guarantee that the fragments themselves have been constructed using an erasure (or error-correcting)

code. A Byzantine sender can exploit this fact, creating a valid Merkle tree over random pieces of data instead of using a coding procedure and trigger the execution. Every node that receives enough fragments can then deduce that the fragments have not been constructed correctly and abort the execution. However, this check typically happens *at the end* of the algorithm, i.e., as many bits as in a successful execution have already been transmitted.

Detectors can remedy this situation as the following sketch of an algorithm using an $(n, n-2t)$-erasure code illustrates: When readiness is signaled at a node that has received m from the sender, it constructs all n fragments and broadcasts its own fragment. As readiness implies that there must be at least $n-2t$ honest nodes that have received m from the sender when using either $\mathcal{D}_b$ or $\mathcal{D}_t$, each node receives at least $n-2t$ valid fragments and is thus able to reconstruct m. Given m, each node can then derive and broadcast its own fragment before delivering m. Since $t < n/3$, the communication complexity of this algorithm is

$$(n-1)|m| + n(n-1)\frac{|m|}{n-2t} + O(\kappa n^2 \log(n)) < 4n|m| + O(\kappa n^2 \log(n)).$$

In the following, a more complex algorithm, denoted by $\mathcal{A}_c$, is introduced that achieves a lower communication complexity. Algorithm $\mathcal{A}_c$ uses an $(n, n-t)$-erasure code, encoding fragments $f_1, \ldots, f_n$ of the message m in such a way that any $n-t$ out of n fragments can be used to recover the message m. The size of a fragment is

$$|f| \coloneqq \frac{|m|}{n-t} + O(\log(n)), \tag{1}$$

where the $O(\log(n))$ stems from the fact that erasure codes work with a field of cardinality at least n. Additionally, algorithm $\mathcal{A}_c$ uses an $(n, n-2t)$-erasure code, encoding fragments of the fragments, called *mini-fragments* [23]. A fragment can be recovered using any $n-2t$ out of n mini-fragments of size

$$|\mu| \coloneqq \frac{|m|}{(n-t)(n-2t)} + O(\log(n)). \tag{2}$$

As algorithm $\mathcal{A}_c$ works with (mini-)fragments, the hash function `H` used by the detector is evaluated over the (mini-)fragments instead of the message m itself. For any $i, j \in \{1, \ldots, n\}$, let π_i and π_{ij} denote the (integrity) proof of fragment f_i and the j^{th} mini-fragment of f_i, respectively. Merkle proofs are commonly used to prove the integrity of (mini-)fragments, with a size of $|\pi| \in O(\kappa \log(n))$. In this case, the hash function `H` returns the Merkle root hash. An alternative would be to use a commitment scheme such as KZG [18], reducing the size to $|\pi| \in O(\kappa)$ for an improved communication complexity. The disadvantage is that it requires a trusted setup and introduces an additional cryptographic assumption [4]. Since $\mathcal{A}_c$ is also a detector-triggered algorithm, there is only one valid hash h for a specific message m that must be considered. Thus, we assume that received fragments and mini-fragments for any other hash are silently discarded. Each node v_i stores any received fragment f_j together with π_j in the set F. Similarly, received mini-fragments μ_{ij} of its own fragment f_i are stored in the set M alongside π_{ij} and π_i.

Algorithm $\mathcal{A}_c$ utilizes several functions: Given message m and index $i \in \{1, \ldots, n\}$, the function `get_fragment` returns the fragment-proof pair (f_i, π_i). In a similar manner, given m and indices i and j, the function `get_mini_fragment` returns the tuple $(\mu_{ij}, \pi_{ij}, \pi_i)$. Once at least $n-t$ fragments are collected in the set F, the message m is derived from the fragments using the function `recover_message`, whereas node v_i's fragment f_i can be recovered using the function `recover_fragment` after collecting at least $n-2t$ mini-fragments in the set M.

Algorithm 4 Algorithm $\mathcal{A}_c$ at node v_i using detector $\mathcal{D}$. initially, $F = M = \{\}$, and $m = \perp$.

if `reliable_broadcast(`m`)` invoked **and** $v_i = sender$ **then**
  broadcast m

▷ Received and verified tuples (f_j, π_j) and $(\mu_{ij}, \pi_{ij}, \pi_j)$ are stored in F and M, respectively.

if $\mathcal{D}$`.ready()` **then**
  if $m \neq \perp$ **and not** sent mini-fragments before **then**
    for $v_j \in V \setminus \mathcal{D}$`.validators()` **do**
      $(\mu_{ji}, \pi_{ji}, \pi_j) \coloneqq$ `get_mini_fragment(`m, j, i`)`
      send `mini-fragment(`$\mu_{ji}, \pi_{ji}, \pi_j$`)` to v_j

  if $(f_i, \pi_i) \in F$ **and not** sent (f_i, π_i) before **then**
    for $v_j \in V \setminus \mathcal{D}$`.validators()` **do**
      send `fragment(`f_i, π_i`)` to v_j

  if $(f_i, \pi_i) \notin F$ **and** $m \neq \perp$ **then**
    $(f_i, \pi_i) \coloneqq$ `get_fragment(`m, i`)`
    $F \coloneqq F \cup \{(f_i, \pi_i)\}$

  if $(f_i, \pi_i) \notin F$ **and** $|M| \geq n - 2t$ **then**
    $(f_i, \pi_i) \coloneqq$ `recover_fragment(`M`)`
    $F \coloneqq F \cup \{(f_i, \pi_i)\}$

  if $m = \perp$ **and** $|F| \geq n - t$ **then**
    $m \coloneqq$ `recover_message(`F`)`

  if sent f_i **and** sent mini-fragments **and** $m \neq \perp$ **and not** delivered m **then**
    `deliver(`m`)`

The sender starts the execution by broadcasting m. Once readiness is signaled, the following actions may occur. An honest node v_i sends messages in two situations: First, if m is available locally, v_i sends `mini-fragment(`$\mu_{ji}, \pi_{ji}, \pi_j$`)` to v_j for all j where $v_j \notin \mathcal{D}$`.validators()`. Second, if f_i is available, it sends fragment(f_i, π_i) to all nodes not in the set $\mathcal{D}$`.validators()` as well. Node v_i can obtain fragment f_i by invoking the function `get_fragment` when m is available. Alternatively, f_i can be recovered from $n - 2t$ mini-fragments in the set M using the function `recover_fragment`. The message itself can be recovered after collecting $n - t$ fragments in F. Lastly, once m is available and the fragment and mini-fragments are sent as stated before, m is delivered. Algorithm $\mathcal{A}_c$, given in Algorithm 4, is a correct reliable broadcast algorithm.

▶ **Theorem 15.** *Algorithm $\mathcal{A}_c$ using detector $\mathcal{D}_b$ or $\mathcal{D}_t$ implements reliable broadcast in the asynchronous communication model with $t < n/3$ Byzantine nodes.*

Proof. Property 5 holds when using either detector because an honest sender initially broadcasts the message. It follows from the specification of the algorithm that a message is only delivered when readiness was signaled beforehand. Thus, it remains to show that readiness implies delivery to show that Property 6 holds.

Assume that readiness is signaled at some node. Due to Property 1, the detectors at all honest nodes eventually signal readiness. Since there must be an honest node that received at least $n-t$ `attest` messages if either $\mathcal{D}_b$ or $\mathcal{D}_t$ is used, there are $n-2t$ honest nodes that have received m. As soon as these $n-2t$ honest nodes receive m and reach the readiness condition, they will send mini-fragments as defined in Algorithm 4. For any honest node v that did not receive m, it holds that v is not in the `validators()` set of any honest node because it never broadcasts an `attest` message. Hence it follows that every honest node receives at least $n-2t$ mini-fragments and is able to recover and send its fragment. As a result, every honest node that did not get m will eventually obtain at least $n-t$ fragments and reconstruct the message. Thus, $m \neq \perp$ holds at all honest nodes eventually, which leads to the delivery of m after sending the fragments and mini-fragments. ◀

Algorithm $\mathcal{A}_c$ indeed achieves a lower communication complexity than the coding-based algorithm sketched before.

▶ **Theorem 16** (Communication complexity). *It holds for algorithm $\mathcal{A}_c$ that $\mathcal{L}(n) < 2 + \frac{1}{2n}$ for successful executions and $\mathcal{L}(n) = 0$ for failed executions.*

Proof. In addition to the sender's initial broadcast, each honest node sends one fragment message and mini-fragment message each to every node in the set $V \setminus \mathcal{D}.\texttt{validators()}$ of cardinality at most $2t \leq n-t-1$ for a communication complexity of

$$\begin{aligned}
\mathcal{C}(n,|m|) &= (n-1)|m| + n(n-t-1)(|\mu| + |f| + O(|\pi|)) \\
&\overset{(1),(2)}{\leq} (n-1)|m| + n(n-t-1)\left(\frac{|m|}{n-t} + \frac{|m|}{(n-t)(n-2t)}\right) + O\left(n^2(|\pi| + \log(n))\right) \\
&< (n-1)|m| + n(n-t)\left(\frac{|m|}{n-t} + \frac{|m|}{(n-t)(n-2t)}\right) \\
&\quad - \frac{n}{n-t}|m| + O\left(n^2(|\pi| + \log(n))\right) \\
&= 2n|m| + \left(\frac{n}{n-2t} - \frac{n}{n-t} - 1\right)|m| + O\left(n^2(|\pi| + \log(n))\right) \\
&\overset{n>3t}{<} 2n|m| + \frac{|m|}{2} + O\left(n^2(|\pi| + \log(n))\right),
\end{aligned}$$

which implies that $\mathcal{L}(n) < 2 + \frac{1}{2n}$.

The result $\mathcal{L}(n) = 0$ for failed executions follows from the fact that $\mathcal{D}_b$ and $\mathcal{D}_t$ are efficient and $\mathcal{A}_c$ is a detector-triggered algorithm. ◀

Regarding the time complexity of $\mathcal{A}_c$, the same result as for algorithm $\mathcal{A}_s$ holds.

▶ **Theorem 17** (Time complexity). *If the sender is honest, then the time complexity of $\mathcal{A}_c$ corresponds to the time complexity of the used detector $\mathcal{D}_b$ or $\mathcal{D}_t$.*

Proof. If the sender is honest, all nodes obtain m after 1 time unit. The time complexity of both detectors is greater than 1. As soon as readiness is signaled, every honest node can compute and send its fragment and mini-fragments before delivering m without delay. ◀

5 Related Work

As discussed in §1, Bracha presented the first reliable broadcast algorithm with a communication complexity of $O(n^2|m|)$ [6]. While the communication complexity is high, the algorithm is *error-free*, i.e., it does not depend on any cryptographic assumptions. The algorithm

based on erasure coding by Cachin and Tessaro improved the communication complexity to $O(n|m| + \kappa n^2 \log(n))$, which is asymptotically optimal for $|m| \in \Omega(\kappa n \log(n))$ [8]. However, it is not error-free as it uses cryptographically strong hash functions.

With respect to error-free reliable broadcast, the time complexity has been reduced successively from $O(n|m| + n^4 \log(n))$ [27] to $O(n|m| + n^3 \log(n))$ [26] and then to $O(n|m| + n^2 \log(n))$ [3]. It has further been shown how to achieve the bound $O(n|m| + n^2 \log(n^3/\varepsilon))$ using a probabilistic algorithm, guaranteeing validity but agreement and totality only hold with probability $1 - \varepsilon$ [1].

As far as algorithms are concerned that make use of cryptographic primitives, an algorithm has been presented with a communication complexity of $O(n|m| + \kappa n^2)$ using only collision-resistant hash functions [10], improving upon the algorithm by Cachin and Tessaro. The downside of this algorithm is that it has a higher computational cost and is not balanced. As stated in §4.2, when using a commitment scheme such as KZG [18] instead of Merkle proofs in the algorithm by Cachin and Tessaro, the same communication complexity can be attained without sacrificing the balanced nature of the algorithm and without inducing a higher computational cost. The downside is that a trusted setup and an additional cryptographic assumption is required. Alternatively, the same bound can be achieved using a public key infrastructure and cryptographic accumulators [26]. Threshold signatures can be used to obtain an improved bound on the communication complexity of $O(n|m| + \kappa n + n^2)$ [3].

The constant in the $O(n|m|)$ term is at least 3 in all of these algorithms. In an effort to improve efficiency for large messages, algorithms have been proposed to bring this constant down, first to 2 [21] and then to 3/2 [22, 23], which is conjectured to be optimal. Thus, the best known algorithms come close to the lower bound of $\Omega(n|m| + n^2)$ [13].

The reliable broadcast problem has been studied in a variety of models. It has been shown how stochastic sampling can be used to obtain a probabilistic algorithm violating each property with a fixed probability [16]. Moreover, reliable broadcast has also been considered in a model with dynamic membership [15]. The efficacy when combining Bracha's algorithm with Dolev's reliable communication protocol [12] to broadcast messages reliably in partially connected networks has been analyzed using simulations [5]. Lastly, the problem that composing n parallel reliable broadcast instances can lead to non-termination has also been addressed, proposing *quit-resistant reliable broadcast* enabling parties to quit early [25].

6 Conclusion

Existing reliable broadcast algorithms are not efficient in failed executions in the worst case. In this paper, the concept of detectors is introduced, supporting reliable broadcast algorithms in keeping the communication complexity low in the failure case. Detectors further have powerful properties that facilitate the design of reliable broadcast algorithms. Moreover, they can reduce or even eliminate redundant transmissions in the absence of faults, which may prove to be relevant in practice. A reliable broadcast algorithm has been presented that makes use of a detector, achieving an overhead factor of $2 + O(1/n)$, which is nearly optimal when considering algorithms where the sender broadcasts the message first. However, there is still a gap between the achieved bound and the lower bound of 5/3 that can be explored. Another open question is whether a lower overhead factor can be obtained when the sender disperses the message to a strict subset of the nodes. Since fewer nodes obtain the message in this step, this modification increases the onus on the honest nodes to ensure that every node obtains the message eventually. These questions may be addressed in future research.

References

1 Ittai Abraham and Gilad Asharov. Gradecast in Synchrony and Reliable Broadcast in Asynchrony with Optimal Resilience, Efficiency, and Unconditional Security. In *Proc. 43rd ACM Symposium on Principles of Distributed Computing (PODC)*, pages 392–398, 2022. doi:10.1145/3519270.3538451.

2 Ittai Abraham, Philipp Jovanovic, Mary Maller, Sarah Meiklejohn, Gilad Stern, and Alin Tomescu. Reaching Consensus for Asynchronous Key Generation. In *Proc. 42nd ACM Symposium on Principles of Distributed Computing (PODC)*, pages 363–373, 2021.

3 Nicolas Alhaddad, Sourav Das, Sisi Duan, Ling Ren, Mayank Varia, Zhuolun Xiang, and Haibin Zhang. Balanced Byzantine Reliable Broadcast with Near-Optimal Communication and Improved Computation. In *Proc. 43rd ACM Symposium on Principles of Distributed Computing (PODC)*, pages 399–417, 2022. doi:10.1145/3519270.3538475.

4 Dan Boneh and Xavier Boyen. Short Signatures Without Random Oracles. In *Proc. 23rd Annual International Conference on the Theory and Applications of Cryptographic Techniques (Eurocrypt)*, pages 56–73, 2004. doi:10.1007/978-3-540-24676-3_4.

5 Silvia Bonomi, Jérémie Decouchant, Giovanni Farina, Vincent Rahli, and Sébastien Tixeuil. Practical Byzantine Reliable Broadcast on Partially Connected Networks. In *Proc. 41st International Conference on Distributed Computing Systems (ICDCS)*, pages 506–516, 2021. doi:10.1109/ICDCS51616.2021.00055.

6 Gabriel Bracha. Asynchronous Byzantine Agreement Protocols. *Information and Computation*, 75(2):130–143, 1987. doi:10.1016/0890-5401(87)90054-X.

7 Christian Cachin and Jonathan A Poritz. Secure Intrusion-tolerant Replication on the Internet. In *Proc. International Conference on Dependable Systems and Networks (DSN)*, pages 167–176, 2002. doi:10.1109/DSN.2002.1028897.

8 Christian Cachin and Stefano Tessaro. Asynchronous Verifiable Information Dispersal. In *Proc. 24th IEEE Symposium on Reliable Distributed Systems (SRDS)*, pages 191–201, 2005. doi:10.1109/RELDIS.2005.9.

9 George Danezis, Lefteris Kokoris-Kogias, Alberto Sonnino, and Alexander Spiegelman. Narwhal and Tusk: A DAG-based Mempool and Efficient BFT Consensus. In *Proc. 17th European Conference on Computer Systems (EuroSys)*, pages 34–50, 2022. doi:10.1145/3492321.3519594.

10 Sourav Das, Zhuolun Xiang, and Ling Ren. Asynchronous Data Dissemination and its Applications. In *Proc. 28th ACM Conference on Computer and Communications Security (CCS)*, pages 2705–2721, 2021. doi:10.1145/3460120.3484808.

11 Sourav Das, Thomas Yurek, Zhuolun Xiang, Andrew Miller, Lefteris Kokoris-Kogias, and Ling Ren. Practical Asynchronous Distributed Key Generation. In *Proc. IEEE Symposium on Security and Privacy (S&P)*, pages 2518–2534, 2022. doi:10.1109/SP46214.2022.9833584.

12 Danny Dolev. Unanimity in an Unknown and Unreliable Environment. In *Proc. 22nd Annual Symposium on Foundations of Computer Science (FOCS)*, pages 159–168, 1981. doi:10.1109/SFCS.1981.53.

13 Danny Dolev and Rüdiger Reischuk. Bounds on Information Exchange for Byzantine Agreement. *Journal of the ACM (JACM)*, 32(1):191–204, 1985. doi:10.1145/2455.214112.

14 Sisi Duan, Michael K. Reiter, and Haibin Zhang. BEAT: Asynchronous BFT Made Practical. In *Proc. 25th ACM Conference on Computer and Communications Security (CCS)*, pages 2028–2041, 2018. doi:10.1145/3243734.3243812.

15 Rachid Guerraoui, Jovan Komatovic, Petr Kuznetsov, Yvonne-Anne Pignolet, Dragos-Adrian Seredinschi, and Andrei Tonkikh. Dynamic Byzantine Reliable Broadcast. In *Proc. 24th International Conference on Principles of Distributed Systems (OPODIS)*, pages 23:1–23:18, 2020. doi:10.4230/LIPICS.OPODIS.2020.23.

16 Rachid Guerraoui, Petr Kuznetsov, Matteo Monti, Matej Pavlovic, Dragos-Adrian Seredinschi, and Yann Vonlanthen. Scalable Byzantine Reliable Broadcast. In *Proc. 33rd International Symposium on Distributed Computing (DISC)*, 2019.

17 Bingyong Guo, Zhenliang Lu, Qiang Tang, Jing Xu, and Zhenfeng Zhang. Dumbo: Faster Asynchronous BFT Protocols. In *Proc. 27th ACM Conference on Computer and Communications Security (CCS)*, pages 803–818, 2020. doi:10.1145/3372297.3417262.

18 Aniket Kate, Gregory M Zaverucha, and Ian Goldberg. Constant-Size Commitments to Polynomials and Their Applications. In *Proc. 16th International Conference on the Theory and Application of Cryptology and Information Security (Asiacrypt)*, pages 177–194, 2010. doi:10.1007/978-3-642-17373-8_11.

19 Idit Keidar, Eleftherios Kokoris-Kogias, Oded Naor, and Alexander Spiegelman. All You Need is DAG. In *Proc. 42nd ACM Symposium on Principles of Distributed Computing (PODC)*, pages 165–175, 2021. doi:10.1145/3465084.3467905.

20 Eleftherios Kokoris-Kogias, Dahlia Malkhi, and Alexander Spiegelman. Asynchronous Distributed Key Generation for Computationally-Secure Randomness, Consensus, and Threshold Signatures. In *Proc. 27th ACM Conference on Computer and Communications Security (CCS)*, pages 1751–1767, 2020. doi:10.1145/3372297.3423364.

21 Thomas Locher. Byzantine Reliable Broadcast with Low Communication and Time Complexity. In *Proc. 28th International Conference on Principles of Distributed Systems (OPODIS)*, volume 324, pages 16:1–16:17, 2024. doi:10.4230/LIPICS.OPODIS.2024.16.

22 Thomas Locher and Victor Shoup. Improving the Round Complexity of MiniCast. Cryptology ePrint Archive, Paper 2025/779, 2025. URL: https://eprint.iacr.org/2025/779.

23 Thomas Locher and Victor Shoup. MiniCast: Minimizing the Communication Complexity of Reliable Broadcast. In *Proc. 44th Annual International Conference on the Theory and Applications of Cryptographic Techniques (Eurocrypt)*, 2025.

24 Andrew Miller, Yu Xia, Kyle Croman, Elaine Shi, and Dawn Song. The Honey Badger of BFT Protocols. In *Proc. 23rd ACM Conference on Computer and Communications Security (CCS)*, pages 31–42, 2016. doi:10.1145/2976749.2978399.

25 Mose Mizrahi Erbes and Roger Wattenhofer. Quit-Resistant Reliable Broadcast and Efficient Terminating Gather. In *Proc. 28th International Conference on Principles of Distributed Systems (OPODIS)*, volume 324, pages 15:1–15:22, 2024. doi:10.4230/LIPICS.OPODIS.2024.15.

26 Kartik Nayak, Ling Ren, Elaine Shi, Nitin H Vaidya, and Zhuolun Xiang. Improved Extension Protocols for Byzantine Broadcast and Agreement. In *Proc. 34th International Symposium on Distributed Computing (DISC)*, 2020.

27 Arpita Patra. Error-free Multi-valued Broadcast and Byzantine Agreement with Optimal Communication Complexity. In *Proc. 15th International Conference on Principles of Distributed Systems (OPODIS)*, pages 34–49, 2011. doi:10.1007/978-3-642-25873-2_4.

28 Victor Shoup. Sing a song of Simplex. In *Proc. 38th International Symposium on Distributed Computing (DISC)*, pages 37:1–37:22, 2024. doi:10.4230/LIPICS.DISC.2024.37.

29 Alexander Spiegelman, Neil Giridharan, Alberto Sonnino, and Lefteris Kokoris-Kogias. Bullshark: DAG BFT Protocols Made Practical. In *Proc. 29th ACM Conference on Computer and Communications Security (CCS)*, pages 2705–2718, 2022. doi:10.1145/3548606.3559361.

30 Thomas Yurek, Licheng Luo, Jaiden Fairoze, Aniket Kate, and Andrew Miller. hbACSS: How to Robustly Share Many Secrets. In *Proc. 29th Annual Network and Distributed System Security Symposium (NDSS)*, 2022.

31 Haibin Zhang and Sisi Duan. PACE: Fully Parallelizable BFT from Reproposable Byzantine Agreement. In *Proc. 29th ACM Conference on Computer and Communications Security (CCS)*, pages 3151–3164, 2022. doi:10.1145/3548606.3559348.

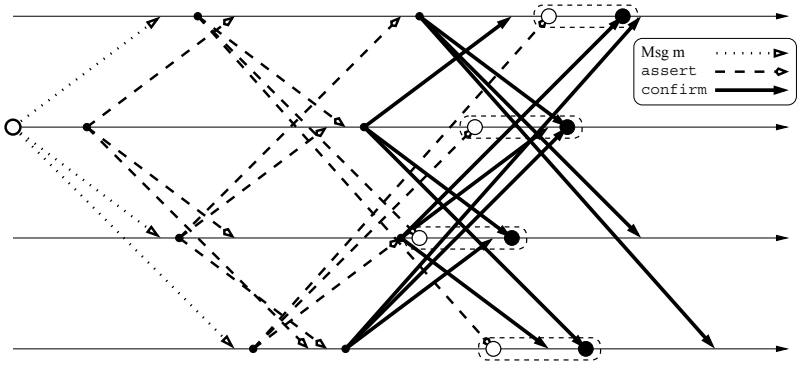

Figure 1 Fault-free execution of $\mathcal{D}_b$, each node receiving n `assert` messages before receiving $n - t$ `confirm` messages.

A Appendix

A.1 Discussion and Practical Considerations

It is worth noting that the presented algorithms are not only of theoretical interest. Under benign network conditions, the delay until the detector signals readiness can be sufficient for all nodes to report that they received the sender's message in a fault-free execution. As an example, Figure 1 depicts an execution of detector $\mathcal{D}_b$ where every node receives not only m but also all $n = 4$ `assert` messages (marked with a white circle) before receiving $n - t = 3$ `confirm` messages (marked with a black circle). In this execution, $|A(h_{max})| = V$ holds at every node, and thus neither $\mathcal{A}_s$ nor $\mathcal{A}_c$ sends any additional messages, implying that $\mathcal{L}(n) = 1$, i.e., there is no overhead with respect to the message distribution itself.

If each node sends the same amount of data up to a constant factor, the algorithm is called *balanced* [3]. Formally, let b_{min} and b_{max} denote the minimum and maximum number of bits sent by honest nodes in any (successful or failed) execution. If $b_{max} \leq c \cdot b_{min}$ for some constant $c \geq 1$, then the algorithm is called *balanced*. The analysis of algorithm $\mathcal{A}_c$ in §4.2 shows that the sender effectively sends as many bits as all other nodes combined, i.e., $\mathcal{A}_c$ is not balanced, as opposed to algorithm $\mathcal{A}_s$. The lack of balance may not be an issue in practice, in particular when all nodes frequently initiate instances of reliable broadcast at nearly the same time. Naturally, having a balanced algorithm is preferable when there are only a few instances at any time and the latency must be kept small when broadcasting large messages. An obvious question is whether a balanced algorithm exists that guarantees that $\mathcal{L}(n) = 0$ for failed executions. Unfortunately, this is only possible if every node sends many bits as the following argument shows. Assuming that the entire message m must be known in order to determine its validity, an honest node will only send m or parts of it if m is available locally or it received sufficient assurance that m is a valid message. Obviously, t Byzantine nodes may erroneously report that m is valid, which implies that an honest node must receive an assurance from at least $t + 1$ nodes in the absence of m. Hence, an honest sender must send m to at least $t + 1$ honest nodes, i.e., the sender must send at least $\Omega(n|m|)$ bits. A consequence of this result is that any algorithm that guarantees that $\mathcal{L}(n) = 0$ for failed executions where some nodes send $o(n|m|)$ bits cannot be balanced. It further implies that algorithms that guarantee that $\mathcal{L}(n) \in o(n)$ cannot be balanced or they cannot guarantee that $\mathcal{L}(n) = 0$ for failed executions.

A.2 Lower Bound

In this section, we consider the class $\mathfrak{A}$ of algorithms where the sender sends m to all other nodes in the first step and prove the following result in the asynchronous communication model.

▶ **Theorem 18.** *If communication is asynchronous and there are $t < n/3$ Byzantine nodes, it holds that $L(n) \geq \frac{5}{3}$ for every reliable broadcast algorithm $\mathcal{A} \in \mathfrak{A}$.*

Proof. We consider the following three scenarios, each having a specific set $\mathcal{V}$, $|\mathcal{V}| = t$, excluding the sender denoted by v_s:

1. The sender v_s is honest but the nodes in $\mathcal{V}$ are Byzantine.
2. The sender v_s is Byzantine and sends m to the nodes in $V \setminus \mathcal{V}$. All nodes other than the sender are honest.
3. The sender v_s is Byzantine and sends m to the honest nodes in $V \setminus \mathcal{V}$. There are t Byzantine nodes in $V \setminus \mathcal{V}$.

The nodes in $\mathcal{V}$ may send messages indicating that they did not receive m from the sender to all other nodes except the sender but any other message from these nodes is scheduled to arrive after the delivery of m. Note that a message must be delivered in Scenario (1) due to the validity condition.

Let $\bar{\mathcal{V}}$ denote the complement of $\mathcal{V}$ excluding the sender, i.e., $\bar{\mathcal{V}} \coloneqq V \setminus \mathcal{V} \setminus \{v_s\}$. Scenario (1) and Scenario (2) are indistinguishable for any node $v \in \bar{\mathcal{V}}$ because it is impossible to differentiate between the sender or the nodes in $\mathcal{V}$ being dishonest.

Moreover, Scenario (2) and Scenario (3) are indistinguishable for any node $v \in \bar{\mathcal{V}}$ that is honest in both executions if the Byzantine nodes behave correctly toward the honest nodes in $\bar{\mathcal{V}}$. indistinguishability holds even if the Byzantine nodes exhibit faulty behavior toward the nodes in $\mathcal{V}$ because no such information is received from nodes in $\mathcal{V}$. Consequently, Scenario (1) and Scenario (3) are also indistinguishable for nodes in $\bar{\mathcal{V}}$ that are honest in both executions.

Since the nodes in $\bar{\mathcal{V}}$ deliver m in Scenario (1) and Scenario (2) is indistinguishable for these nodes, they deliver m in Scenario (2) as well. However, the totality condition implies that the nodes in $\mathcal{V}$ must also deliver m. Let b_i denote the number of bits that $v_i \in V \setminus \mathcal{V}$ sends to nodes in $\mathcal{V}$ before delivering m, excluding the sender's initial message, in Scenario (1).

Assume that there is a subset $\mathcal{S} \subset V \setminus \mathcal{V}$, $v_s \in \mathcal{S}$ and $|\mathcal{S}| = n-2t$, such that $\sum_{v_i \in \mathcal{S}} b_i < t|m|$. In this case, we define that the nodes in $\bar{\mathcal{V}} \setminus \mathcal{S}$ are the $(n-t-1)-(n-2t-1) = t$ Byzantine nodes in Scenario (3). Since Scenario (1) and Scenario (3) are indistinguishable for the honest nodes in $V \setminus \mathcal{V}$, they send the same messages to nodes in $\mathcal{V}$ as in Scenario (1). However, the nodes in $\mathcal{V}$ receive fewer than $t|m|$ bits if the Byzantine nodes refuse to send any bits. Thus, there must be an honest node in $\mathcal{V}$ that does not deliver m, violating the totality condition. It follows that we must have that

$$\forall \mathcal{S} \subset V \setminus \mathcal{V}, v_s \in \mathcal{S} \text{ and } |\mathcal{S}| = n - 2t : \sum_{v_i \in \mathcal{S}} b_i \geq t|m|,$$

and therefore the average number of bits sent must be at least

$$\bar{b} \coloneqq \frac{1}{|V \setminus \mathcal{V}|} \sum_{v_i \in V \setminus \mathcal{V}} b_i \geq \frac{t|m|}{n-2t}. \tag{3}$$

Thus, for $n = 3t + 1$, it must hold that

$$\sum_{v_i \in V \setminus \mathcal{V}} b_i = \bar{b}|V \setminus \mathcal{V}| \overset{(3)}{\geq} \frac{t|m|}{n-2t}|V \setminus \mathcal{V}| = \frac{|m|t(n-t)}{n-2t} \overset{n\to\infty}{=} \frac{2}{3}n|m|.$$

Since Scenario (3) is indistinguishable from Scenario (1) irrespective of how the t Byzantine nodes are chosen among the nodes in $\bar{\mathcal{V}}$, at least $\frac{2}{3}n|m|$ bits are sent in Scenario (1) as well. As the honest sender broadcasts m first, the total communication complexity must be at least $n|m| + \frac{2}{3}n|m| = \frac{5}{3}n|m|$. ◀

This theorem shows that algorithm $\mathcal{A}_c$ is nearly optimal for algorithms in class $\mathfrak{A}$. Moreover, the result is interesting in that the lower bound exceeds the upper bound of $L(n) \leq \frac{3}{2}$ of the best known balanced algorithm that does not aim to minimize the communication complexity for failed executions. It is an open question whether any algorithm where the sender sends $\alpha n|m|$ bits, for some $\alpha \in (0, 1)$, can achieve a better communication complexity. We conjecture that little can be gained by choosing $\alpha < 1$ if $\mathcal{L}(n) = 0$ must hold for all failed executions.