Brief Announcement: Towards Optimal Communication Byzantine Reliable Broadcast Under a Message Adversary

Brief Announcement: Towards Optimal Communication Byzantine Reliable Broadcast Under a Message Adversary

Timothé Albouy
Univ Rennes, Inria, CNRS, IRISA, 35042 Rennes-cedex, France

Davide Frey
Univ Rennes, Inria, CNRS, IRISA, 35042 Rennes-cedex, France

Ran Gelles
Bar-Ilan University, Ramat Gan, Israel

Carmit Hazay
Bar-Ilan University, Ramat Gan, Israel

Michel Raynal
Univ Rennes, Inria, CNRS, IRISA, 35042 Rennes-cedex, France

Elad Michael Schiller
Chalmers University of Technology, Gothenburg, Sweden

François Taïani
Univ Rennes, Inria, CNRS, IRISA, 35042 Rennes-cedex, France

Vassilis Zikas
Purdue University, West Lafayette, IN, USA

Abstract

We address the problem of Reliable Broadcast in asynchronous message-passing systems with n nodes, of which up to t are malicious (faulty), in addition to a *message adversary* that can drop some of the messages sent by correct (non-faulty) nodes. We present a Message-Adversary-Tolerant Byzantine Reliable Broadcast (MBRB) algorithm that communicates an almost optimal amount of $\mathcal{O}(|m| + n^2\kappa)$ bits per node, where $|m|$ represents the length of the application message and $\kappa = \Omega(\log n)$ is a security parameter. This improves upon the state-of-the-art MBRB solution (Albouy, Frey, Raynal, and Taïani, TCS 2023), which incurs communication of $\mathcal{O}(n|m| + n^2\kappa)$ bits per node. Our solution sends at most $4n^2$ messages overall, which is asymptotically optimal. Reduced communication is achieved by employing coding techniques that replace the need for all nodes to (re-)broadcast the entire application message m. Instead, nodes forward authenticated fragments of the encoding of m using an erasure-correcting code. Under the cryptographic assumptions of PKI and collision-resistant hash, and assuming $n > 3t+2d$, where the adversary drops at most d messages per broadcast, our algorithm allows at least $\ell = n - t - (1+\epsilon)d$ (for any $\epsilon > 0$) correct nodes to reconstruct m, despite missing fragments caused by the malicious nodes and the message adversary.

2012 ACM Subject Classification Theory of computation → Distributed algorithms

Keywords and phrases Asynchronous message-passing, Byzantine fault-tolerance, Message adversary, Reliable Broadcast

Digital Object Identifier 10.4230/LIPIcs.DISC.2024.41

Related Version *Full Version*: `https://arxiv.org/abs/2312.16253` [1]

Funding This work was partially supported by the French ANR projects ByBloS (ANR-20-CE25-0002-01) and PriCLeSS (ANR-10-LABX-07-81), devoted to the modular design of building blocks for large-scale Byzantine-tolerant multi-users applications. Research supported in part by the United States-Israel Binational Science Foundation (BSF) through Grant No. 2020277.

Acknowledgements R. Gelles would like to thank Paderborn University and CISPA – Helmholtz Center for Information Security for hosting him while part of this research was done.

1 Introduction

Byzantine Reliable Broadcast (*BRB* for short) allows n asynchronous nodes to agree eventually on a message sent by a designated node, the *sender*, despite the possible malicious (Byzantine) behavior by some nodes and the transmission network [4]. Byzantine reliable broadcast plays a crucial role in several key applications, including consensus algorithms, replication, event notification, and distributed file systems, among others. These systems sometimes require broadcasting large messages or files (e.g., permissioned blockchains), and thus, reducing the communication overhead to a minimum is an important aspect of achieving scalability. In that vein, this work aims at providing *communication efficient* solutions for the task of reliable broadcast in the presence of node and link faults.

A significant challenge to reliable broadcast algorithms arises when the message-passing system is unreliable and possibly cooperates with the Byzantine nodes. Link faults [11, 12] give Byzantine nodes (potentially limited) control over certain network links, enabling them to omit or corrupt messages (an ability captured under the umbrella term *message adversary* [9]). This work focuses on a specific type of *message adversary* [9] that can only omit messages sent by correct nodes, but that cannot alter their content. This message adversary abstracts cases related to *silent churn*, where nodes may voluntarily or involuntarily disconnect from the network without explicitly notifying other nodes.

Problem overview. We assume n nodes over an asynchronous network, where a message can be delayed for an arbitrary yet finite amount of time (unless omitted by the message adversary). We assume the existence of t Byzantine nodes and a message adversary capable of omitting up to d messages per node's broadcast. To be more precise, a node communicates through a comm primitive (or a similar multicast/unicast primitive that targets a dynamically defined subset of processes), which results in the transmission of n messages, with each node being sent one message, including the sender. The message adversary can choose to omit messages in transit to a subset of at most d correct processes. The adversary is only limited by the size of that subset. For instance, between different comm invocations, the adversary has the freedom to modify the set of correct processes to which messages are omitted. Furthermore, a designated sender node holds a message m that it wishes to broadcast to all the nodes.

An algorithm that satisfies the requirements of reliable broadcast despite Byzantine nodes and a message adversary is called a *Message-adversary Byzantine Reliable Broadcast* (MBRB) algorithm. The detailed version of MBRB's requirements was formulated in [2], see Section 2.

Background. Albouy, Frey, Raynal, and Taïani [2] recently proposed a Message-adversary Byzantine Reliable Broadcast algorithm (which we denote AFRT for short) for asynchronous networks that withstands the presence of t Byzantine nodes and a message adversary capable of omitting up to d messages per node's broadcast. AFRT guarantees the reliable delivery of any message when $n > 3t+2d$. Moreover, they demonstrate the necessity of this bound on the number of Byzantine nodes and omitted messages, as no reliable broadcast algorithm exists otherwise.

One caveat of AFRT regards its communication efficiency. While it achieves an optimal number of $\mathcal{O}(n^2)$ messages, and an optimal delivery power $\ell = n - t - d$, each node's communication requires $\mathcal{O}(n \cdot (|m| + n\kappa))$ bits, where $|m|$ represents the number of bits in the broadcast message and κ is the length of the digital signatures used in their algorithm. In the current work, we design an algorithm that significantly reduces the communication cost per node while preserving the total number of messages communicated. Our solution features at most $4n$ messages per correct node (corresponding to $4n^2$ messages overall), and only $\mathcal{O}(|m| + n^2\kappa)$ bits per correct node. Overall, $\mathcal{O}(n|m| + n^3\kappa)$ bits are communicated by correct nodes. Reducing the second term to $(n^2 \log n)\kappa$ can be done by employing standard techniques of *threshold signatures*, which replace the need to communicate a quorum of signatures; see, e.g., [3]. Note that $\Omega(n|m| + n^2\kappa)$ is a straightforward lower bound on the overall communication for deterministic algorithms using signatures (up to the size of the signature), see [5, 8], as every correct node must receive the message m, and as the reliable broadcast of a single bit necessitates at least $\Omega(n^2)$ messages [6].

Contributions. This work is the first to present an MBRB algorithm tolerating an hybrid adversary combining t Byzantine nodes and a Message Adversary of power d, while providing optimal Byzantine resilience, near-optimal communication, and near-optimal delivery power ℓ.

2 Preliminaries

General notations and conventions. For a positive integer n, let $[n]$ denote the set $\{1, 2, \ldots, n\}$. A sequence of elements $(x_1, \ldots, x_n)$ is shorthanded as $(x_i)_{i\in[n]}$. We use the symbol '-' to indicate any possible value. That is, $(h, \text{-})$ means a tuple where the second index includes any arbitrary value which we do not care about. All logarithms are base 2.

Nodes and Network. We focus on asynchronous message-passing systems that have no guarantees of communication delay. Also, the algorithm cannot explicitly access the clock or use timeouts. The system consists of a set, $\mathcal{P} = \{p_1, \ldots, p_n\}$, of n fail-prone nodes (or processes). We identify party i with p_i.

Communication means. Any ordered pair of nodes $p_i, p_j \in \mathcal{P}$ has access to a communication channel, $channel_{i,j}$. Each node can send messages to all nodes (possibly by sending a different message to each node). That is, any node, $p_i \in \mathcal{P}$, can invoke the transmission macro, $\mathsf{comm}(m_1, \ldots, m_n)$, that communicates the message m_j to p_j over $channel_{i,j}$. The message m_j can also be empty, in which case nothing will be sent to p_j. However, in our algorithms, all messages sent in a single comm activation will have the same length. Furthermore, when a node sends the same message m to all nodes, we write $\mathsf{broadcast}(m) = \mathsf{comm}(m, m, \ldots, m)$ for shorthand. We call each message m_j transmitted by the protocol an *implementation message* (or simply, a *message*) to distinguish such messages from the *application*-level messages, i.e., the one the sender wishes to broadcast.

Byzantine nodes. Faulty nodes are called *Byzantine* and their adversarial behavior can deviate from the proposed algorithm in any manner. They might perform any arbitrary computation, and we assume their computing power is at least as strong as that of non-faulty nodes, yet not as strong as to undermine the security of the cryptographic signatures we use (see below). We assume that, at most, t nodes are faulty, where t is a value known to the nodes. Non-faulty nodes are called *correct nodes.* The set of correct nodes contains c nodes where $n - t \leq c \leq n$. The value of c is unknown.

Message adversary. This entity can remove implementation messages from the communication channels used by correct nodes when they invoke $\mathsf{comm}(\cdot)$. More precisely, during each $\mathsf{comm}(m_1, \ldots, m_n)$ call, the adversary has the discretion to eliminate up to d messages in the set $\{m_i\}$ from their corresponding communication channels where they were queued. Similar to [2], we assume $n > 3t+2d$.

Error Correction Codes. A central tool used in our algorithm is an error-correction code (ECC) [10]. Intuitively speaking, an ECC takes a message as input and adds redundancy to create a codeword from which the original message can be recovered even when parts of the codeword are corrupted. In this work, we focus on *erasures*, a corruption that replaces a symbol of the codeword with a special erasure mark $\perp$.

Cryptographic Primitives. Our algorithm relies on cryptographic assumptions. We assume that the Byzantine nodes are computationally bounded with respect to the security parameter, denoted by κ. That is, all cryptographic algorithms are polynomially bounded in the input 1^κ. We assume that $\kappa = \Omega(\log n)$.

Hash functions. A collision-resistant hash is a function[1] $\mathsf{hash} : \{0,1\}^* \to \{0,1\}^\kappa$ that satisfies the following *collision resistance* property: For any computationally bounded algorithm A and any $x \in \{0,1\}^*$, $\Pr[A(x) = x' \wedge \mathsf{hash}(x') = \mathsf{hash}(x)] < 2^{-\Omega(\kappa)}$. I.e., finding a pair x, x' with the same hash is infeasible, except with negligible probability in the security parameter.

Signature schemes. A *digital signature* scheme is a pair of possibly randomized algorithms $\mathsf{SIG} = (\mathsf{sign}, \mathsf{Verify})$. The signing algorithm executed by node p_i (denoted, sign_i) takes a message m and implicitly a private key. It then produces a signature $\sigma = \mathsf{sign}_i(m)$. The verifying algorithm takes a message, its corresponding signature, and the identity of the signer (and implicitly a public key), and outputs a single bit, $b = \mathsf{Verify}(m, \sigma, i)$, which indicates whether the signature is valid or not, $b \in \{\mathsf{valid}, \mathsf{invalid}\}$.

Merkle Trees [7]. These are means to commit to a message composed of several fragments so that one can prove, for each fragment independently, that it belongs to the committed message. This primitive is parameterized by a security parameter κ and consists of two functions: $\mathsf{MerkleTree}(\cdot)$ which generates the proofs for each fragment of the message, and $\mathsf{VerifyMerkle}(\cdot)$, which given a fragment along with its proof, verifies that the fragment indeed belongs to the committed message.

Specification of the MBRB primitive. The Objective of MBRB is to guarantee a reliable delivery of a message while upholding specific safety and liveness criteria, despite actions taken by Byzantine nodes and the message adversary An MBRB algorithm contains the MBRB-broadcast and MBRB-deliver operations.

Definition 1 specifies the safety and liveness properties. Safety ensures that messages are delivered correctly without spurious messages, duplication, or duplicity. The liveness guarantee that if a correct node broadcasts a message, it will eventually be delivered by at least one correct node (MBRB-Local-delivery), and that if a correct node delivers a message from any specific sender, that message will eventually be delivered by a sufficient number, ℓ, of correct nodes (MBRB-Global-delivery), where ℓ is a measure of the *delivery power* of the MBRB algorithm and might depend on the adversary's power, i.e., on t and d.

▶ **Definition 1.** *An MBRB is an algorithm that satisfies the following properties.*

- ***MBRB-Validity.*** *Suppose p_s is correct and a correct node, p_i, MBRB-delivers an application message m. Then, node p_s has MBRB-broadcast m (before that MBRB-delivery).*
- ***MBRB-No-duplication.*** *A correct node p_i MBRB-delivers at most one application message m.*
- ***MBRB-No-duplicity.*** *No two different correct nodes MBRB-deliver different application messages from node p_s.*
- ***MBRB-Local-delivery.*** *Suppose p_s is correct and MBRB-broadcasts an application message m. At least one correct node, p_j, eventually MBRB-delivers m from node p_s.*
- ***MBRB-Global-delivery.*** *Suppose a correct node, p_i, MBRB-delivers an application message m from p_s. Then, at least ℓ correct nodes MBRB-deliver m from p_s.*

[1] Formally speaking, a hash function must be chosen randomly from a family of possible hash functions. Otherwise, an adversarial algorithm A exists. We avoid a formal treatment of this issue in our paper. In practice, a fixed function is used (e.g., SHA2 or SHA3).

3 The Coded-MBRB algorithm

The proposed solution, named Coded MBRB (Algorithm 2), allows a distinguished sender p_s to disseminate one specific application message m. In the description below, we assume there is a single sender, p_s, and all nodes know its identity p_s. In the full version [1], we discuss how to extend this algorithm so that it implements a general MBRB algorithm, allowing any node to be the sender, as well as allowing multiple instances of the MBRB, either with the same or different senders, to run concurrently.

Algorithm description. MBRB-BROADCAST(m) (line 6) allows the sender to start disseminating the application message, m. It is designed to be executed by the sender process, p_s. The initial step of the sender (line 7) invokes COMPUTEFRAGMERKLETREE(m) (Algorithm 1), which encodes the message m using an error-correction code, divides it into n fragments and constructs a Merkle tree that includes the different fragments. The function returns several essential values: the Merkle root hash h, and the fragment details $(\tilde{m}_j, \pi_j, j)$, which contains the fragment data itself $\tilde{m}_j$ (the j-th part of the codeword $\mathsf{ECC}(m)$), a proof of inclusion π_j for that part, and the respective index j of each fragment.

The sender node, p_s, is responsible for signing the computed Merkle root hash h and generating a signature, denoted sig_s (line 8). Notably, this signature includes p_s's identifier. The sender then initiates m's propagation by employing the operation COMM (line 9), which sends to each process p_j the Merkle root hash h, the j-th fragment details $(\tilde{m}_j, \pi_j, j)$, and the signature sig_s (line 8). When this message (or later messages communicated in the algorithm) is received by some node p_i, it first verifies that all the signatures and the Merkle proofs that the message contains are valid, and that p_s's signature is included in the messages; otherwise, the message is ignored. This action is encapsulated by ISVALID() (lines 11, 17, and 33).

The rest of the algorithm progresses in two phases. The first phase is responsible for message dissemination, which forwards message fragments received by the sender. The other role of this phase is reaching a quorum of nodes that vouch for the same message. A node vouches for a single message by signing its hash value. Nodes collect and store signatures until it is evident that sufficiently many nodes agree on the same message. The subsequent phase focuses on disseminating the quorum of signatures so that it is observed by at least ℓ correct nodes, and on successfully terminating while ensuring the delivery of the reconstructed message.

Algorithm 1 The COMPUTEFRAGMERKLETREE(m) function.

```
1 Function COMPUTEFRAGMERKLETREE($m$) is
2     $\tilde{m} \leftarrow \mathsf{ECC}(m)$        ▷ Such that $m$ is recoverable from $k = \Omega(n)$ fragments
3     let $\tilde{m}_1, \dots, \tilde{m}_n$ be $n$ equal size fragments of $\tilde{m}$
4     $(h, \pi_1, \dots, \pi_n) \leftarrow \mathsf{MerkleTree}(\tilde{m}_1, \dots, \tilde{m}_n)$ ;
5     return $\big(h, (\tilde{m}_j, \pi_j, j)_{j \in [n]}\big)$
```

Analysis. The following theorem states that our algorithm is correct. Due to page limit, the complete proof and discussion on the assumptions appear in the full version [1].

▶ **Theorem 2** (Main). *Assume $n > 3t + 2d$, $k \leq (n - t - 2d)$ and $\varepsilon > 0$. Algorithm 2 implements an MBRB solution with $\ell > n - t - (1 + \varepsilon)d$. Any algorithm activation on the input message m communicates $4n^2$ messages, where each node communicates $O(|m| + n^2\kappa)$ bits overall.*

Algorithm 2 The Coded MBRB Algorithm (code for p_i, single-shot, single-sender).

```
6  Function MBRBBROADCAST$(m)$ is                    $\triangleright$ only executed by the sender, $p_s$
7      $\big(h, (\tilde{m}_j, \pi_j, j)_j\big) \leftarrow$ COMPUTEFRAGMERKLETREE$(m)$
8      $sig_s \leftarrow \big(\mathsf{sign}_s(h), s\big)$
9      comm$(v_1, \dots, v_n)$ where $v_j = \langle$SEND$, h, (\tilde{m}_j, \pi_j), sig_s\rangle$

   Phase I: Message dissemination

10 Upon $\langle$SEND$, h', (\tilde{m}_i, \pi_i, i), sig_s\rangle$ arrival from $p_s$ do
11     if $\neg$ISVALID$\big(h', \{(\tilde{m}_i, \pi_i, i)\}, \{sig_s\}\big)$ then return       $\triangleright$ discard invalid messages
12     if $p_i$ already executed l. 15 or signed a msg from $p_s$ with hash $h'' \neq h'$ then return
13     store $\tilde{m}_i$ and $sig_s$ for $h'$
14     $sig_i \leftarrow \big(\mathsf{sign}_i(h'), i\big)$ ; store $sig_i$ for $h'$
15     broadcast $\langle$FORWARD$, h', (\tilde{m}_i, \pi_i, i), \{sig_s, sig_i\}\rangle$

16 Upon $\langle$FORWARD$, h', fragtuple_j, sigs_j = \{sig_s, sig_j\}\rangle$ arrival from $p_j$ do
17     if $\neg$ISVALID$\big(h', \{fragtuple_j\}, sigs_j\big)$ then return       $\triangleright$ discard invalid messages
18     if $p_i$ already signed a message from $p_s$ with hash $h'' \neq h'$ then return
19     store $sigs_j$ for $h'$
20     if $fragtuple_j \neq \bot$ then
21         $(\tilde{m}_j, \pi_j, j) \leftarrow fragtuple_j$ ; store $\tilde{m}_j$ for $h'$
22     if no FORWARD message sent yet then
23         $sig_i \leftarrow \big(\mathsf{sign}_i(h'), i\big)$ ; store $sig_i$ for $h'$
24         broadcast $\langle$FORWARD$, h', \bot, \{sig_s, sig_i\}\rangle$

   Phase II: Reaching Quorum and Termination

25 When $\left\{ \begin{array}{l} \exists h' : \big|\{\text{stored signatures for } h'\}\big| > \frac{n+t}{2} \wedge \big|\{\text{stored } \tilde{m}_j \text{ for } h'\}\big| \geq k \\ \wedge \text{ no message has been MBRB-delivered yet} \end{array} \right\}$ do
26     $m_i \leftarrow \mathsf{ECC}^{-1}(\tilde{m}_1, \dots, \tilde{m}_n)$, $\begin{cases} \text{where } \tilde{m}_j \text{ are taken from line 25;} \\ \text{when a fragment is missing use } \bot. \end{cases}$
27     $\big(h, (\tilde{m}'_j, \pi'_j, j)_j\big) \leftarrow$ COMPUTEFRAGMERKLETREE$(m_i)$
28     if $h' = h$ then
29         $sigs_h \leftarrow \{\text{all stored signatures for } h\}$
30         comm$(v_1, \dots, v_n)$ where $v_j = \langle$BUNDLE$, h, (\tilde{m}'_i, \pi'_i, i), (\tilde{m}'_j, \pi'_j, j), sigs_h\rangle$
31         MBRBDELIVER$(m_i)$

32 Upon $\langle$BUNDLE$, h', (\tilde{m}'_j, \pi'_j, j), fragtuple'_i, sigs\rangle$ arrival from $p_j$ do
33     if $\neg$ISVALID$\big(h', \big\{(\tilde{m}'_j, \pi'_j, j), fragtuple'_i\big\}, sigs\big)$ then return  $\triangleright$ discard invalid msgs
34     if $|sigs| \leq \frac{n+t}{2}$ then return                   $\triangleright$ discard msgs with no quorum
35     store $(\tilde{m}'_j, \pi'_j, j)$ and $sigs$ for $h'$
36     if no BUNDLE message has been sent yet $\wedge$ $fragtuple'_i \neq \bot$ then
37         $(\tilde{m}'_i, \pi'_i, i) \leftarrow fragtuple'_i$
38         store $(\tilde{m}'_i, \pi'_i, i)$ for $h'$
39         broadcast $\langle$BUNDLE$, h', (\tilde{m}'_i, \pi'_i, i), \bot, sigs\rangle$
```

References

1 Timothé Albouy, Davide Frey, Ran Gelles, Carmit Hazay, Michel Raynal, Elad Michael Schiller, François Taïani, and Vassilis Zikas. Towards optimal communication Byzantine reliable broadcast under a message adversary. *CoRR*, abs/2312.16253, 2023. `doi:10.48550/arXiv.2312.16253`.

2 Timothé Albouy, Davide Frey, Michel Raynal, and François Taïani. Asynchronous Byzantine reliable broadcast with a message adversary. *Theor. Comput. Sci.*, 978:114110, 2023. `doi:10.1016/J.TCS.2023.114110`.

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 *Proceedings of the 2022 ACM Symposium on Principles of Distributed Computing*, PODC'22, pages 399–417, 2022. `doi:10.1145/3519270.3538475`.

4 Gabriel Bracha. Asynchronous byzantine agreement protocols. *Inf. Comput.*, 75(2):130–143, 1987. `doi:10.1016/0890-5401(87)90054-X`.

5 Sourav Das, Zhuolun Xiang, and Ling Ren. Asynchronous data dissemination and its applications. In *CCS*, pages 2705–2721. ACM, 2021. `doi:10.1145/3460120.3484808`.

6 Danny Dolev and Rüdiger Reischuk. Bounds on information exchange for Byzantine agreement. *J. ACM*, 32(1):191–204, January 1985. `doi:10.1145/2455.214112`.

7 Ralph C. Merkle. A certified digital signature. In Gilles Brassard, editor, *Advances in Cryptology — CRYPTO' 89 Proceedings*, pages 218–238, New York, NY, 1990. Springer New York. `doi:10.1007/0-387-34805-0_21`.

8 Kartik Nayak, Ling Ren, Elaine Shi, Nitin H. Vaidya, and Zhuolun Xiang. Improved extension protocols for Byzantine broadcast and agreement. In *34th International Symposium on Distributed Computing (DISC 2020)*, volume 179, pages 28:1–28:17, 2020. `doi:10.4230/LIPIcs.DISC.2020.28`.

9 Michel Raynal. Message adversaries. In *Encyclopedia of Algorithms*, pages 1272–1276. Springer, 2016. `doi:10.1007/978-1-4939-2864-4_609`.

10 Ron M. Roth. *Introduction to coding theory*. Cambridge University Press, 2006.

11 Nicola Santoro and Peter Widmayer. Time is not a healer. In *STACS 89*, pages 304–313, Berlin, Heidelberg, 1989. Springer Berlin Heidelberg. `doi:10.1007/BFb0028994`.

12 Nicola Santoro and Peter Widmayer. Agreement in synchronous networks with ubiquitous faults. *Theor. Comput. Sci.*, 384(2-3):232–249, 2007. `doi:10.1016/J.TCS.2007.04.036`.