

# Trading off $t$ -Resilience for Efficiency in Asynchronous Byzantine Reliable Broadcast

Damien Imbs<sup>†</sup> and Michel Raynal<sup>\*,‡</sup>

<sup>†</sup> LIF, Université d'Aix-Marseille, 13288 Marseille, France

<sup>\*</sup> Institut Universitaire de France

<sup>‡</sup> IRISA, Université de Rennes, 35042 Rennes Cedex, France

damien.imbs@lif.univ-mrs.fr

raynal@irisa.fr

## Abstract

This paper presents a simple and efficient reliable broadcast algorithm for asynchronous message-passing systems made up of  $n$  processes, among which up to  $t < n/5$  may behave arbitrarily (Byzantine processes). This algorithm requires two communication steps and  $n^2 - 1$  messages. When compared to Bracha's algorithm, which is resilience optimal ( $t < n/3$ ) and requires three communication steps and  $2n^2 - n - 1$  messages, the proposed algorithm shows an interesting tradeoff between communication efficiency and  $t$ -resilience.

**Keywords:** Algorithm, Asynchronous system, Byzantine process, Distributed computing, Fault-tolerance, Message-passing, Reliable broadcast.

## 1 Introduction

**On reliable broadcast** *Reliable broadcast* (RB) is a communication abstraction central to fault-tolerant distributed systems. It allows each process to broadcast messages to all processes despite failures. More precisely, it guarantees that the non-faulty processes deliver the same set of messages and this set includes at least all the messages they broadcast. It can also contain messages broadcast by faulty processes.

The fundamental property of reliable broadcast lies in the fact that no two correct processes deliver different sets of messages. This communication abstraction is a basic building block used to build a *total order reliable broadcast* abstraction (sometimes called “atomic broadcast”), which adds the total order property on message delivery (see e.g., [1, 2, 5, 6, 7, 9, 11]). In turn, total order broadcast is a basic building block for state machine replication, which is a fundamental paradigm in fault-tolerance.

**Reliable broadcast in the presence of Byzantine processes** Reliable broadcast has been studied in the context of Byzantine failures since the eighties. A process commits a Byzantine failure if it behaves arbitrarily [8, 10]. Such a behavior can be intentional (also called malicious) or the result of a transient fault which altered the content of local variables of a process, thereby modifying its intended behavior in an unpredictable way.

An elegant algorithm, due to G. Bracha, implementing the reliable broadcast abstraction in an asynchronous system of  $n$  processes which communicate by message-passing, and where up to  $t < n/3$  processes may be Byzantine is described in [3]. This algorithm is signature-free. It is shown in [3, 4] that  $t < n/3$  is an upper bound on the number of Byzantine processes that can be tolerated. Hence, Bracha's algorithm is  $t$ -resilience optimal. This algorithm is based on a “double echo” mechanism of

the value broadcast by the sender process. It uses three types of messages, requires three consecutive communication steps, and  $(n - 1) + 2n(n - 1) = 2n^2 - n - 1$  underlying messages.

**Content of the paper** This paper presents a new signature-free Byzantine-tolerant reliable broadcast algorithm, which uses only two message types, two consecutive communication steps, and  $(n - 1) + n(n - 1) = n^2 - 1$  underlying messages. This gain, with respect Bracha’s algorithm, in both the time and the number of messages, is obtained with a weaker  $t$ -resilience requirement, namely  $t < n/5$  instead of  $t < n/3$ . This shows an interesting tradeoff between communication cost (number of communication steps<sup>1</sup> and the number of messages) on one side, and fault-resilience on the other side (see Table 1).

	fault resilience	communication steps message types	number of messages
Bracha’s algorithm [3]	$n > 3t$	3	$2n^2 - n - 1$
This paper	$n > 5t$	2	$n^2 - 1$

Table 1: Bracha’s algorithm with respect to the proposed algorithm

## 2 Computation Model

**Asynchronous processes** The system is made up of a finite set  $\Pi$  of  $n > 1$  asynchronous sequential processes, namely  $\Pi = \{p_1, \dots, p_n\}$ . “Asynchronous” means that each process proceeds at its own speed, which can vary arbitrarily with time, and always remains unknown to the other processes.

**Communication network** The processes communicate by exchanging messages through an asynchronous reliable point-to-point network. “Asynchronous” means that a message that has been sent is eventually received by its destination process, i.e., there is no bound on message transfer delays. “Reliable” means that the network does not lose, duplicate, modify, or create messages. “Point-to-point” means that there is a bi-directional communication channel between each pair of processes.

A process  $p_i$  sends a message to a process  $p_j$  by invoking the primitive “send TAG( $m$ ) to  $p_j$ ”, where TAG is the type of the message and  $m$  its content. To simplify the presentation, it is assumed that a process can send messages to itself. A process receives a message by executing the primitive “receive()”. The macro-operation “broadcast TAG( $m$ )” is a shortcut for “**for**  $j \in \{1, \dots, n\}$  **do** send TAG( $m$ ) to  $p_j$  **end for**”.

**Failure model** Up to  $t$  processes can exhibit a *Byzantine* behavior. A Byzantine process is a process that behaves arbitrarily: it can crash, fail to send or receive messages, send arbitrary messages, start in an arbitrary state, perform arbitrary state transitions, etc. As a simple example, a Byzantine process, which is assumed to send a message  $m$  to all the processes, can send a message  $m_1$  to some processes, a different message  $m_2$  to another subset of processes, and no message at all to the other processes. Moreover, Byzantine processes can collude to “pollute” the computation. They can also control the network in the sense that they can re-order message deliveries at correct processes. It is however assumed that a Byzantine process cannot send an infinite number of messages.

Let us notice that, as each pair of processes is connected by a channel, a process can identify the sender of each message it receives. Hence, no Byzantine process can impersonate another process. As in Bracha’s algorithm, this allows the proposed algorithm to be signature-free.

---

<sup>1</sup>The number of different message types is always the same as the number of communication steps. This is needed to associate the appropriate processing to each message.

A process that exhibits a Byzantine behavior is also called *faulty*. Otherwise, it is *correct* or *non-faulty*.

### 3 Reliable Broadcast

The reliable broadcast (denoted RB-broadcast) communication abstraction provides each process with two operations, denoted `RB_broadcast()` and `RB_deliver()`. As in [6], we use the following terminology: when a process invokes `RB_broadcast()`, we say that it “RB-broadcasts a message”, and when it executes `RB_deliver()`, we say that it “RB-delivers a message”. RB-broadcast is defined by the following properties.

- **RB-Validity.** If a correct process RB-delivers the message  $\text{MSG}(v)$  from a correct process  $p_i$ , then  $p_i$  RB-broadcast  $\text{MSG}(v)$ .
- **RB-Integrity.** A correct process RB-delivers at most one message from any process  $p_i$ .
- **RB-Agreement.** No two correct processes RB-deliver distinct messages from the same process.
- **RB-Termination-1.** If a correct process RB-broadcast a message, all correct processes eventually RB-deliver this message.
- **RB-Termination-2.** If a correct process RB-delivers a message  $m$  from  $p_i$  (possibly faulty) then all correct processes eventually RB-deliver  $m$  from  $p_i$ .

**On the safety properties’ side** RB-validity relates the output (messages RB-delivered) to the inputs (messages RB-broadcast). RB-integrity states that there is no duplication. RB-agreement states that there is no duplicity: be the sender correct or not, it is not possible for a correct process to RB-deliver  $m$  while another correct process RB-delivers  $m' \neq m$ .

**On the liveness properties’ side** The RB-Termination properties state the guarantees on message RB-delivery. RB-Termination-1 states that a message RB-broadcast by a correct process is RB-delivered by all correct processes. RB-Termination-2 gives its name to reliable broadcast. Be the sender correct or not, every message RB-delivered by a correct process is RB-delivered by all correct processes.

It follows that all correct processes RB-deliver the same set of messages, and this set contains at least all the messages RB-broadcast by the correct processes.

**RB-broadcasting a sequence of messages** The previous definition considers that each correct process RB-broadcasts at most one message. It is easily possible to extend it to the case where a correct process RB-broadcasts a sequence of messages. In the algorithm that follows, the identity  $j$  of the sender  $p_j$  must then be replaced by a pair  $\langle j, sn \rangle$ , where  $sn$  is the sequence number associated by  $p_j$  with the message.

### 4 A Communication-Efficient Reliable Broadcast Algorithm for $t < n/5$

**The algorithm** Algorithm 1, which implements the reliable broadcast abstraction, consists of a client side and a server side. On the client side, when a (correct) process wants to RB-broadcast an application message  $\text{MSG}(v_i)$ , it simply broadcasts the algorithm message  $\text{INIT}(i, v_i)$ .

On the server side, a process can receive two types of messages.

- When it receives a message  $\text{INIT}(j, v)$  (necessarily from process  $p_j$  as the processes are connected by bidirectional channels), a process  $p_i$  broadcasts the message  $\text{WITNESS}(j, v)$  (line 2) if (a) this message is the first message  $\text{INIT}()$   $p_i$  receives from  $p_j$ , and (b)  $p_i$  has not yet broadcast a message  $\text{WITNESS}(j, -)$ .

- When a process  $p_i$  receives a message  $\text{WITNESS}(j, v)$  (from any process), it does the following.
  - If  $p_i$  has received the same message from “enough-1” processes (where “enough-1” is  $(n - 2t)$ , i.e., at least  $n - 3t \geq 2t + 1$  correct processes sent this message, and  $p_i$  has not yet broadcast the same message  $\text{WITNESS}(j, v)$ , it forwards it to all processes. This concludes the “forwarding phase” of  $p_i$  as far as a message of  $p_j$  is concerned.
  - If  $p_i$  received the same message from “enough-2” processes (where “enough-2” means “at least  $(n - t)$  processes”, i.e., the message was received from at least  $n - 2t \geq 3t + 1$  correct processes,  $p_i$  locally RB-delivers  $\text{MSG}(j, v)$  if not yet done. This concludes the “RB-delivering phase” of a message from  $p_j$ , as far as  $p_i$  is concerned.

**operation** RB\_broadcast  $\text{MSG}(v_i)$  **is**

(1) broadcast  $\text{INIT}(i, v_i)$ .

**when**  $\text{INIT}(j, v)$  **is received from**  $p_j$  **do**

(2) **if** (first reception of  $\text{INIT}(j, -)$  and  $\text{WITNESS}(j, -)$  not yet broadcast) **then** broadcast  $\text{WITNESS}(j, v)$  **end if**.

**when**  $\text{WITNESS}(j, v)$  **is received do**

(3) **if** ( $\text{WITNESS}(j, v)$  received from  $(n - 2t)$  different processes and  $\text{WITNESS}(j, v)$  not yet broadcast)

(4) **then** broadcast  $\text{WITNESS}(j, v)$

(5) **end if**;

(6) **if** ( $\text{WITNESS}(j, v)$  received from  $(n - t)$  different processes and  $\text{MSG}(j, -)$  not yet RB\_delivered)

(7) **then** RB\_deliver  $\text{MSG}(j, v)$

(8) **end if**.

Algorithm 1: Communication-efficient Byzantine reliable broadcast algorithm for  $t < n/5$

**Cost of the algorithm** Only two types of messages are used (INIT and WITNESS). It is easy to see that the broadcast of a message by a correct process requires two consecutive communication steps (broadcast of an INIT message whose receptions entail at most  $n$  broadcasts of WITNESS messages). Not counting the messages that a process sends to itself, a reliable broadcast by a correct process costs  $(n - 1)$  messages INIT and at most  $n(n - 1)$  messages WITNESS (counting only the messages under the control of the algorithm).

## 5 Proof of the Algorithm

The proof assumes  $t < n/5$ .

**Lemma 1.** *Let  $\text{INIT}(i, v)$  be a message that is never broadcast by a correct process  $p_i$ . If Byzantine processes broadcast the message  $\text{WITNESS}(i, v)$ , no correct process will forward this message at line 4.*

**Proof** Let us consider the worst case where  $t$  processes are Byzantine and each of them broadcasts the same message  $\text{WITNESS}(i, v)$ . For a correct process  $p_j$  to forward this message at line 4, the forwarding predicate of line 3 must be satisfied. But, in order for this predicate to be true at a correct process  $p_j$ , this process must receive the message  $\text{WITNESS}(i, v)$  from  $n - 2t$  different processes. As  $n - 2t > t$ , this cannot occur.  $\square_{\text{Lemma 1}}$

**Theorem 1.** *Algorithm 1 implements the reliable broadcast abstraction in  $n$ -process asynchronous message-passing systems in which up to  $t < n/5$  processes may commit Byzantine failures.*

## Proof

Proof of the RB-Validity property.

Let  $p_i$  be a correct process that invokes  $\text{RB\_broadcast MSG}(v)$  and consequently broadcasts the message  $\text{INIT}(i, v)$  at line 1. The fact that no correct process RB-delivers a message different from  $\text{MSG}(i, v)$  comes from the following observation. To RB-deliver a message  $\text{MSG}(i, v')$ , a correct process must receive the message  $\text{WITNESS}(i, v')$  from more than  $(n - t)$  different processes (line 6). But if the (at most)  $t$  Byzantine processes forge a fake message  $\text{WITNESS}(i, v')$ , with  $v \neq v'$ , this message will never be forwarded by correct processes (Lemma 1). As  $n - t > t$ , it follows from the predicate of line 6 that the content of the message RB-delivered by any correct process cannot be different from  $(i, v)$ .

Proof of the RB-Integrity property.

This property follows directly from the RB-delivery predicate of line 6, namely, at most one message  $\text{MSG}(j, v)$  can be delivered by any correct process  $p_i$ .

Proof of the RB-Agreement property.

Let  $p_k$  be a process that sends at least one message  $\text{INIT}(k, -)$ . If  $p_k$  is correct, it sends at most one such message. If it is Byzantine, it may send more. Hence, let us assume that  $p_k$  sends  $\text{INIT}(k, v_1)$ ,  $\text{INIT}(k, v_2)$ , etc.,  $\text{INIT}(k, v_m)$ , where  $m \geq 1$ . For any  $x \in [1..m]$ , let  $Q_x$  be the set of correct processes that receive the message  $\text{INIT}(k, v_x)$ , and these receptions directed them to broadcast the message  $\text{WITNESS}(k, v_x)$  at line 2. Due to the fact that only  $p_k$  can send messages  $\text{INIT}(k, -)$ , it follows from the reception predicate of line 2 that a correct process can belong to at most one set  $Q_x$ . Hence, we have:  $(x \neq y) \Rightarrow Q_x \cap Q_y = \emptyset$ . We consider two cases according to the size of the sets  $Q_x$ .

- Let us first consider a set  $Q_x$  such that  $|Q_x| < n - 3t$ . Let  $p_j$  be any correct process that does not belong to  $Q_x$  (hence  $p_j$  does not process the message  $\text{INIT}(k, v_x)$  at line 2 if it receives it). As  $n - t > n - 3t$ ,  $p_j$  does exist. Process  $p_j$  can receive the message  $\text{WITNESS}(k, v_x)$  (a) from each process of  $Q_x$ , and (b) from each of the  $t$  Byzantine processes. It follows that  $p_j$  can receive  $\text{WITNESS}(k, v_x)$  from at most  $t + |Q_x|$  different processes. As  $t + |Q_x| < n - 2t$ , the predicate of line 3 cannot be satisfied at  $p_j$ , and consequently,  $p_j$  (i.e., any correct process  $\notin Q_x$ ) will never send the message  $\text{WITNESS}(k, v_x)$ . Hence the number of messages  $\text{WITNESS}(k, v_x)$  received by any correct process can never attain  $(n - t)$ , from which we conclude that no correct process RB-delivers  $\text{MSG}(k, v_x)$ . It follows that, if there is a single set (of correct processes)  $Q_z$  (i.e.,  $z = m = 1$ ), and this set is such that  $|Q_z| \geq n - 3t$ , at most one message  $\text{MSG}(k, -)$  may be RB-delivered by a correct process, and this message is then  $\text{MSG}(k, v_z)$ .
- Let us now consider the case where there are at least two different sets of correct processes  $Q_x$  and  $Q_y$ , each of size at least  $(n - 3t)$ . Let us remind that, in the worst case, each of the  $t$  Byzantine processes can systematically play double game by sending both  $\text{WITNESS}(k, v_x)$  and  $\text{WITNESS}(k, v_y)$  to each correct process without having received the associated message  $\text{INIT}(k, -)$ . Moreover, in the worst case, we have exactly  $(n - t)$  correct processes. (If, in a given execution, strictly less than  $t$  processes are Byzantine, we consider the equivalent execution in which exactly  $t$  processes are Byzantine, and some of them behave like correct processes.) As both  $Q_x$  and  $Q_y$  contain only correct processes, and  $Q_x \cap Q_y = \emptyset$ , it follows that  $|Q_x| + |Q_y| + t \leq n$ , which implies  $2n - 6t + t \leq |Q_x| + |Q_y| + t \leq n$ , from which we obtain  $5t \geq n$ , which contradicts the assumption on  $t$  (namely,  $n > 5t$ ). Consequently, at least one of  $Q_x$  and  $Q_y$  is composed of less than  $(n - 3t)$  correct processes. It follows from the previous paragraph that the corresponding message  $\text{MSG}(k, -)$  cannot be RB-delivered by a correct process. As this is true for any pair of sets  $Q_x$  and  $Q_y$ , it follows that, if  $p_k$  sends several messages  $\text{INIT}(k, v_1)$ ,  $\text{INIT}(k, v_2)$ , etc.,  $\text{INIT}(k, v_m)$ , at most one of them can give rise to a set  $Q_x$  such that  $|Q_x| \geq n - 3t$ , and, consequently, at most one message  $\text{MSG}(k, v_x)$  can be RB-delivered by any correct process.

Proof of the RB-Termination-1 property.

Let  $p_i$  be a correct process that invokes  $\text{RB\_broadcast MSG}(v)$  and consequently broadcasts the message  $\text{INIT}(i, v_i)$  at line 1. It follows that any correct process  $p_j$  receives this message. Let us remember that, due to the network connectivity assumption, there is a channel connecting  $p_i$  to  $p_j$  and consequently the message  $\text{INIT}(i, v)$  cannot be a fake message forged by a Byzantine process. Moreover, due to Lemma 1, no message  $\text{WITNESS}(i, v')$ , with  $v' \neq v$ , forged by Byzantine processes, can be forwarded by a correct process at lines 3-4. Hence, when  $p_j$  receives  $\text{INIT}(i, v)$ , it broadcasts the message  $\text{WITNESS}(i, v)$  at line 2. It follows that every correct process eventually receives this message from  $(n - t)$  different processes and consequently locally RB-delivers the message  $\text{MSG}(i, v)$  at lines 6-8, which proves the property.

Proof of the RB-Termination-2 property.

Let  $p_i$  be a correct process that RB-delivers the message  $\text{MSG}(k, v)$ . It follows that the RB-delivery predicate of line 6 is true at  $p_i$ , and consequently,  $p_i$  received the message  $\text{WITNESS}(k, v)$  from at least  $(n - t)$  different processes, i.e., from at least  $n - 2t > t$  correct processes.

It follows that at least  $(n - 2t)$  correct processes broadcast  $\text{WITNESS}(k, v)$ , and consequently the predicate of line 3 is eventually true at each correct process. Hence, every correct process eventually broadcasts the message  $\text{WITNESS}(k, v)$  at line 4, if not yet done before (at line 2 or line 4). As there are at least  $(n - t)$  correct processes, each of them eventually receives  $\text{WITNESS}(k, v)$  from  $(n - t)$  different processes, and consequently RB-delivers  $\text{MSG}(k, v)$  at line 7, which proves the property.

□<sub>Theorem 1</sub>

## Acknowledgments

The authors want to thank the reviewers for their constructive comments. This work has been partially supported by the Franco-German DFG-ANR Project 40300781 DISCMAT devoted to connections between mathematics and distributed computing, and the French ANR project DESCARTES devoted to distributed software engineering.

## References

- [1] Attiya H. and Welch J., *Distributed computing: fundamentals, simulations and advanced topics*, (2d Edition), Wiley-Interscience, 414 pages, 2004.
- [2] Birman K. P., *Guide to Reliable Distributed Systems*. Texts in Computer Science, Springer, 730 pages, 2012 (ISBN 978-1-4471-2415-3).
- [3] Bracha G., Asynchronous Byzantine agreement protocols. *Information & Computation*, 75(2):130-143, 1987.
- [4] Bracha G. and Toueg S., Asynchronous consensus and broadcast protocols. *Journal of the ACM*, 32(4):824-840, 1985.
- [5] Cachin Ch., Guerraoui R., and Rodrigues L., *Reliable and secure distributed programming*, Springer, 367 pages, 2011 (ISBN 978-3-642-15259-7).
- [6] Hadzilacos V. and Toueg S., A Modular Approach to Fault-Tolerant Broadcasts and Related Problems. *Tech Report 94-1425*, 83 pages, Cornell University, Ithaca (USA), 1994.
- [7] Lamport L., The Part-time Parliament. *ACM Transactions on Computer Systems*, 16(2):133-169, 1998.



- [8] Lamport L., Shostack R., and Pease M., The Byzantine generals problem. *ACM Transactions on Programming Languages and Systems*, 4(3)-382-401, 1982.
- [9] Lynch N. A., *Distributed algorithms*. Morgan Kaufmann Pub., San Francisco (CA), 872 pages, 1996 (ISBN 1-55860-384-4).
- [10] Pease M., Shostak R., and Lamport L., Reaching agreement in the presence of faults. *Journal of the ACM*, 27:228-234, 1980.
- [11] Raynal M., *Communication and agreement abstractions for fault-tolerant asynchronous distributed systems*. Morgan & Claypool, 251 pages, 2010 (ISBN 978-1-60845-293-4).