A Modular Approach to Construct Signature-Free BRB Algorithms Under a Message Adversary

Timothé Albouy
Univ Rennes, Inria, CNRS, IRISA, France

Davide Frey
Univ Rennes, Inria, CNRS, IRISA, France

Michel Raynal
Univ Rennes, Inria, CNRS, IRISA, France

François Taïani
Univ Rennes, Inria, CNRS, IRISA, France

Abstract

This paper explores how reliable broadcast can be implemented without signatures when facing a dual adversary that can both corrupt processes and remove messages. More precisely, we consider an asynchronous n-process message-passing system in which up to t processes are Byzantine and where, at the network level, for each message broadcast by a correct process, an adversary can prevent up to d processes from receiving it (the integer d defines the power of the message adversary). So, unlike previous works, this work considers that not only can computing entities be faulty (Byzantine processes), but, in addition, that the network can also lose messages. To this end, the paper adopts a modular strategy and first introduces a new basic communication abstraction denoted $k2\ell$-cast, which simplifies quorum engineering, and studies its properties in this new adversarial context. Then, the paper deconstructs existing signature-free Byzantine-tolerant asynchronous broadcast algorithms and, with the help of the $k2\ell$-cast communication abstraction, reconstructs versions of them that tolerate both Byzantine processes and message adversaries. Interestingly, these reconstructed algorithms are also more efficient than the Byzantine-tolerant-only algorithms from which they originate.

2012 ACM Subject Classification Theory of computation → Distributed algorithms

Keywords and phrases Asynchronous system, Byzantine processes, Communication abstraction, Delivery predicate, Fault-Tolerance, Forwarding predicate, Message adversary, Message loss, Modularity, Quorum, Reliable broadcast, Signature-free algorithm, Two-phase commit

Digital Object Identifier 10.4230/LIPIcs.OPODIS.2022.26

Related Version *Full Version*: `https://arxiv.org/abs/2204.13388`

Funding This work has been partially supported by the French ANR projects ByBloS (ANR-20-CE25-0002-01) and PriCLeSS (ANR-10-LABX-07-81) devoted to the design of modular distributed computing building blocks.

1 Introduction

Context: reliable broadcast and message adversaries. Reliable broadcast (RB) is a fundamental abstraction in distributed computing that lies at the core of many higher-level constructions (including distributed memories, distributed agreement, and state machine replication). Essentially, RB requires that non-faulty (i.e., correct) processes agree on the set of messages they deliver so that this set includes at least all the messages that correct processes have broadcast.

In a failure-free system, implementing reliable broadcast on top of an asynchronous network is relatively straightforward [27]. If processes may fail, and in particular if failed processes may behave arbitrarily (a failure known as Byzantine [21, 26]), implementing reliable broadcast becomes far from trivial as Byzantine processes may collude to fool correct processes [29]. An algorithm that solves reliable broadcast in the presence of Byzantine processes is known as implementing BRB (Byzantine reliable broadcast).

BRB in asynchronous networks (in which no bound is known over message delays) has been extensively studied over the last forty years [1, 2, 7, 11, 12, 18, 20, 23, 22, 25, 29]. Existing BRB algorithms typically assume they execute over a *reliable* point-to-point network, i.e., a network in which sent messages are eventually received. This is a reasonable assumption as most unreliable networks can be made reliable using re-transmissions and acknowledgments (e.g. a timeout-free version of the TCP protocol).

This work takes a drastic turn away from this usual assumption and explores how BRB might be provided when processes execute on an *unreliable* network that might lose point-to-point messages. Our motivation is threefold: First, in volatile networks (e.g., mobile networks or networks under attack), processes might remain disconnected over long periods (e.g., weeks or months), leading in practice to considerable delays (a.k.a. tail latencies) when using re-transmissions. Because most asynchronous Byzantine-tolerant algorithms exploit intersecting quorums, these tail latencies can potentially limit the performance of BRB algorithms drastically, a well-known phenomenon in systems research [14, 15, 34]. Second, re-transmissions require that correct processes be eventually able to receive messages and cannot, therefore, model the permanent disconnection of correct processes. Finally, this question is interesting in its own right, as it opens up novel trade-offs between algorithm guarantees and network requirements, with potential application to the design of reactive distributed algorithms tolerant to both processes and network failures.

The impact of network faults on distributed algorithms has been studied in several works, in particular using the concept of message adversaries (MA). Message adversaries were initially introduced by N. Santoro and P. Widmayer in [31, 32][1], and then used (sometimes implicitly) in many works (e.g., [4, 3, 13, 17, 30, 28, 32, 33]). Initially proposed for synchronous networks, an MA may suppress point-to-point network messages according to rules that define its power. For instance, a tree MA in a synchronous network might suppress any message except those transiting on an (unknown) spanning tree of the network, with this spanning tree possibly changing in each round.

The message losses that an MA causes differ fundamentally from Byzantine faults. This is because an MA can affect the messages sent by any correct process, and can change the processes it targets during an execution, in contrast to Byzantine corruptions that are tied to a set of fixed processes (which is why MA faults are sometimes dubbed *transient* or *mobile*). For instance, it may be tempting to think that Byzantine-fault-tolerant (BFT) algorithms inherently tolerate message losses from correct processes because they can only afford to wait for at most $n - t$ messages (where n is the total number of processes, and t the upper bound on Byzantine processes). In an asynchronous network, a BFT algorithm could therefore miss up to t messages from correct processes, if those are delayed by the scheduler. This scenario only applies, however, in the particular circumstance where the t faulty processes

[1] Where the terminology *communication failure model* and *ubiquitous faults* is used instead of MA. While we consider only message losses, the work of Santoro and Widmayer also considers message additions and corruptions.

send messages that are received and accepted as valid by correct recipients. This caveat is fundamental. If the faulty processes remain silent or send contradicting messages (if they are Byzantine), then a BFT algorithm cannot afford to lose t messages from correct processes.

Content of the paper. This paper combines a Message Adversary with Byzantine processes, and studies the signature-free implementation of Byzantine Reliable Broadcast (BRB) in an asynchronous, fully connected network subject to this MA and to at most t Byzantine faults. The MA models lossy connections by preventing up to d point-to-point messages from reaching their recipient every time a correct process seeks to communicate with the rest of the network.[2]

To limit as much as possible our working assumptions, we further assume that the computability power of the adversary is unbounded (except for the cryptography-based algorithm presented in Section 6), which precludes the use of signatures. (We do assume, however, that each point-to-point communication channel is authenticated.)[3]

This represents a particularly challenging environment, as the MA may target different correct processes every time the network is used or focus indefinitely on the same (correct) victims. Further, the Byzantine processes may collude with the MA for maximal impact.

For clarity, in the remainder of the paper, we simply call *messages* the point-to-point network messages used internally by a BRB algorithm. (The MA may suppress these messages.) We distinguish these messages from the messages the BRB algorithm seeks to disseminate, which we call "*application messages*" (*app-messages* for short). In such a context, the paper presents the following results.

- It first introduces a new modular abstraction, named $k2\ell$-cast, which appears to be a foundational building block to implement BRB abstractions (with or without the presence of an MA). This communication abstraction systematically dissociates the predicate used to forward (network) messages from the predicate that triggers the delivery of an app-message, and lies at the heart of the work presented in the paper. When proving the $k2\ell$-cast communication abstraction, the paper presents an in-depth analysis of the power of an adversary that controls at most t Byzantine processes and an MA of power d.
- Then, the paper deconstructs two signature-free BRB algorithms (Bracha's [11] and Imbs and Raynal's [20] algorithms) and reconstructs versions of them that tolerate *both* Byzantine processes and MA. Interestingly, when considering Byzantine failures only, these deconstructed versions use smaller quorum sizes and are, therefore, more efficient than their initial counterparts.

So, this paper is not only the first to present signature-free BRB algorithms in the context of asynchrony and MA but also the first to propose an intermediary communication abstraction that allows us to obtain efficient BRB algorithms. For clarity, we give in Table 1 the list of acronyms and notations used in this paper.

Roadmap. The paper is composed of 7 sections and one appendix. Section 2 describes the underlying computing model. Section 3 presents the $k2\ell$-cast abstraction and its properties. Section 4 defines the MA-tolerant BRB communication abstraction. Section 5 shows that

[2] A close but different notion was introduced by Dolev in [16], which considers static κ-connected networks. If the adversary selects statically, for each correct sender, d correct processes that do not receive any of this sender's messages, the proposed model includes Dolev's model where $\kappa = n - d$.

[3] Let us mention that the problem of designing an MA-tolerant BRB has been solved in [4] by leveraging digital signatures within a monolithic algorithm. Finding a signature-free counterpart remained, however, an open question, which we answer positively in this paper using a modular strategy.

Table 1 Acronyms and notations.

Acronyms	Meaning
BRB	Byzantine-tolerant reliable broadcast
MA	Message adversary
MBRB	Message adversary- and Byzantine-tolerant reliable broadcast
Notations	**Meaning**
n	total nb of processes in the network
t	upper bound on the nb of Byzantine processes
d	power of the message adversary
c	effective nb of correct processes in a run ($n - t \leq c \leq n$)
k	minimal nb of correct processes that $k2\ell$-cast a message
ℓ	minimal nb of correct processes that $k2\ell$-deliver a message
k'	minimal nb of correct $k2\ell$-casts if there is a correct $k2\ell$-delivery
δ	`true` iff no-duplicity is guaranteed, `false` otherwise
q_d	size of the $k2\ell$-delivery quorum
q_f	size of the forwarding quorum
single	`true` iff only a single message can be endorsed, `false` otherwise

thanks to the $k2\ell$-cast abstraction, existing BRB algorithms can give rise to MA-tolerant BRB algorithms which, when $d = 0$, are more efficient than the BRB algorithms they originate from. Section 6 presents a signature-based implementation of $k2\ell$-cast that possesses optimal guarantees. Finally, Section 7 concludes the paper. Due to page limitations, some proofs and a numerical evaluation of the $k2\ell$-cast abstraction are presented in appendices of this paper and its extended version [5].

2 Computing Model

Process model. The system is composed of n asynchronous sequential processes denoted $p_1, ..., p_n$. Each process p_i has a distinct identity, known to other processes. For simplicity and without loss of generality, we assume that i is the identity of p_i.

In terms of faults, up to $t \geq 0$ processes can be Byzantine, where a Byzantine process is a process whose behavior does not follow the code specified by its algorithm [21, 26]. Byzantine processes may collude to fool non-Byzantine processes (also called correct processes). In this model, the premature stop (crash) of a process is a Byzantine failure. In the following, given an execution, c denotes the effective number of processes that behave correctly in that execution. We always have $n - t \leq c \leq n$. While this number remains unknown to correct processes, it is used to analyze and characterize (more precisely than using its worse value $n - t$) the guarantees provided by the proposed algorithms.

Finally, the processes have no access to random numbers, and their computability power is unbounded. Hence, the algorithms presented in the paper are deterministic and signature-free (except the signature-based algorithm presented in Section 6).

Communication model. Processes communicate by exchanging messages through a fully connected asynchronous point-to-point network, assumed to be reliable in the sense it neither corrupts, duplicates, nor creates messages. As far as messages losses are concerned, the network is under the control of an adversary (see below) that can suppress messages.

Let MSG be a message type and v the associated value. A process can invoke the best-effort broadcast macro-operation denoted ur_broadcast(MSG(v)), which is a shorthand for "**for all** $j \in \{1, \cdots, n\}$ **do** send MSG(v) **to** p_j **end for**". Correct processes are assumed to invoke

ur_broadcast to send messages. When they do, we say that the messages are *ur-broadcast* and *received*. The operation ur_broadcast(MSG(v)) is not reliable. For example, if the invoking process crashes during its invocation, an arbitrary subset of processes receive the message MSG(v). Moreover, due to its very nature, a Byzantine process can send fake messages without using the macro-operation ur_broadcast.

Message adversary. Let d be an integer constant such that $0 \leq d < n - t$. The communication network is controlled by an MA (as defined in Section 1), which eliminates messages ur-broadcast by correct processes, so these messages are lost. More precisely, when a correct process invokes ur_broadcast(MSG(v)), the MA is allowed to arbitrarily suppress up to d copies of the message MSG(v) intended to correct processes[4]. This means that, although the sender is correct, up to d correct processes may miss the message MSG(v). The extreme case $d = 0$ corresponds to the case where no message is lost.

As an example, let us consider a set D of correct processes, where $1 \leq |D| \leq d$, such that during some period of time, the MA suppresses all the messages sent to them. It follows that, during this period of time, this set of processes appears as being input-disconnected from the other correct processes. Note that the size and the content of D can vary with time and are never known by the correct processes.

3 $k2\ell$-Cast Abstraction

Signature-free BRB algorithms [9, 11, 20] often rely on successive waves of internal messages (e.g. the ECHO or READY messages of Bracha's algorithm [11]) to provide safety and liveness. Each wave is characterized by a threshold-based predicate that triggers the algorithm's next phase when fulfilled (e.g. enough ECHO messages for the same app-message m).

In this section, we introduce, implement, and prove a new modular abstraction, called $k2\ell$-cast, that encapsulates a wave/thresholding mechanism that is both Byzantine- and MA-tolerant. As previously announced, we then use this abstraction to reconstruct MA-tolerant BRB algorithms in Section 5 from two existing BRB algorithms [11, 20].

3.1 Definition

$k2\ell$-cast (for k-to-ℓ-cast) is a many-to-many communication abstraction[5]. Intuitively, it relates the number k of correct processes that send a message m (we say that these processes *$k2\ell$-cast* m) with the number ℓ of correct processes that deliver m (we say that they *$k2\ell$-deliver* m). Both k and ℓ are subject to thresholding constraints: enough correct processes must $k2\ell$-cast a message for it to be $k2\ell$-delivered at least once; and as soon as one (correct) $k2\ell$-delivery occurs, some minimal number of correct processes are guaranteed to $k2\ell$-deliver as well.

[4] Note that this message adversary is not limited to algorithms that use the ur_broadcast macro-operation. The same adversary can be equivalently defined for an operation ur_multicast that sends a message to a dynamically defined subset of processes (be it multiple recipients or only one in the case of unicast), by stipulating that the MA can still suppress up to d copies of this message. In this case, the most robust way for correct processes to disseminate a message is to send it to all processes, i.e. to fall back on a ur_broadcast operation.

[5] An example of this family is the binary reliable broadcast introduced in [24], which is defined by specific delivery properties – not including MA-tolerance – allowing binary consensus to be solved efficiently with the help of a common coin.

More formally, $k2\ell$-cast is a multi-shot abstraction, i.e. each app-message m that is $k2\ell$-cast or $k2\ell$-delivered is associated with an identity id. (Typically, such an identity is a pair consisting of a process identity and a sequence number.) It provides two operations, $k2\ell$_cast and $k2\ell$_deliver, whose behavior is defined by the values of four parameters: three integers k', k, ℓ, and a Boolean δ. This behavior is captured by the following six properties:

- Safety:
  - $k2\ell$-Validity. If a correct process p_i $k2\ell$-delivers an app-message m with identity id, then at least k' correct processes $k2\ell$-cast m with identity id.
  - $k2\ell$-No-duplication. A correct process $k2\ell$-delivers at most one app-message m with identity id.
  - $k2\ell$-Conditional-no-duplicity. If the Boolean δ is `true`, then no two different correct processes $k2\ell$-deliver different app-messages with the same identity id.
- Liveness[6]:
  - $k2\ell$-Local-delivery. If at least k correct processes $k2\ell$-cast an app-message m with identity id and no correct process $k2\ell$-casts an app-message $m' \neq m$ with identity id, then at least one correct process $k2\ell$-delivers the app-message m with identity id.
  - $k2\ell$-Weak-Global-delivery. If a correct process $k2\ell$-delivers an app-message m with identity id, then at least ℓ correct processes $k2\ell$-deliver an app-message m' with identity id (each of them possibly different from m).
  - $k2\ell$-Strong-Global-delivery. If a correct process $k2\ell$-delivers an app-message m with identity id, and no correct process $k2\ell$-casts an app-message $m' \neq m$ with identity id, then at least ℓ correct processes $k2\ell$-deliver the app-message m with identity id.

This specification is *parameterized* in the sense that each tuple (k', k, ℓ, δ) defines a specific communication abstraction with different guarantees. This versatility explains why the $k2\ell$-cast abstraction can be used to produce highly compact reconstructions of existing BRB algorithms, rendering them MA-tolerant in the process (using four and three lines of pseudo-code respectively, see Section 5). Despite this versatility, however, we will see in Section 3.2 that $k2\ell$-cast can be implemented using a single (parameterized) algorithm, underscoring the fundamental commonalities of MA-tolerant BRB algorithms.

Intuitively, the parameters k', k, and ℓ hobble the disruption power of the Byzantine/MA adversary by setting limits on the number of correct processes that are either required or guaranteed to be involved in one communication "wave" (corresponding to one identity id). k' sets the minimal number of correct processes that must $k2\ell$-cast for any $k2\ell$-delivery to occur: it thus limits the ability of the Byzantine/MA adversary to trigger spurious $k2\ell$-deliveries. The role of k is symmetrical. It guarantees that some $k2\ell$-delivery will necessarily occur if k correct processes $k2\ell$-cast some message. It thus prevents the adversary from silencing correct processes as soon as some critical mass of them participates. Finally, ℓ captures a "quite-a-few-or-nothing" guarantee that mirrors the traditional "all-or-nothing" delivery guarantee of traditional BRB. As soon as one correct $k2\ell$-delivery occurs (for some identity id), then ℓ correct processes must also $k2\ell$-deliver (with the same identity).

The fourth parameter, δ, is a flag that when `true` enforces agreement between $k2\ell$-deliveries. When $\delta =$ `true`, the $k2\ell$-Conditional-no-duplicity property implies that all the app-messages m' involved in the $k2\ell$-Weak-Global-delivery property are equal to m.

[6] The liveness properties comprise a *local* delivery property that provides a necessary condition for the $k2\ell$-delivery of an app-message by at least *one* correct process, and two *global* delivery properties that consider the collective behavior of correct processes.

Algorithm 1 Signature-free $k2\ell$-cast (code for p_i).

```
object SigFreeK2LCast(q_d, q_f, single) is
(1)  operation k2ℓ_cast(m, id) is
(2)    if (ENDORSE(−, id) not already ur-broadcast)
(3)      then ur_broadcast(ENDORSE(m, id))
(4)    end if.
(5)  when ENDORSE(m, id) is received do
       % ——————— forwarding step ———————
(6)    if (ENDORSE(m, id) received from at least q_f processes
         ∧ ((¬single ∧ ENDORSE(m, id) not already ur-broadcast)
           ∨ ENDORSE(−, id) not already ur-broadcast))
(7)      then ur_broadcast(ENDORSE(m, id))
(8)    end if;
       % ——————— delivery step ———————
(9)    if (ENDORSE(m, id) received from at least q_d processes
         ∧ (−, id) not already k2ℓ-delivered)
(10)     then k2ℓ_deliver(m, id)
(11)   end if.
end object.
```

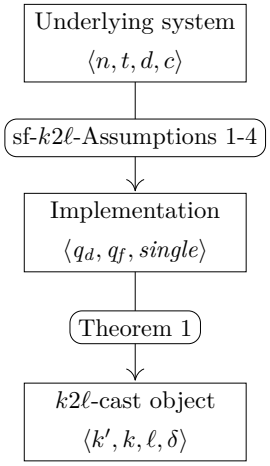


Figure 1 From the system parameters to a $k2\ell$-cast implementation.

3.2 A Signature-Free Implementation of $k2\ell$-Cast

Among the many possible ways of implementing $k2\ell$-cast, this section presents a quorum-based[7] signature-free implementation[8] of the abstraction. To overcome the disruption caused by Byzantine processes and message losses from the MA, our algorithm uses the ur-broadcast primitive (cf. our communication model in Sec. 2) to accumulate and forward ENDORSE messages before deciding whether to deliver. Forwarding and delivery are triggered by *two thresholds* (a pattern also found, for instance, in Bracha's BRB algorithm [11]):

- A first threshold, q_d, triggers the delivery of an app-message m when enough ENDORSE messages supporting m have been received.
- A second threshold, q_f, which is lower than q_d, controls how ENDORSE messages are forwarded during the algorithm's execution.

Forwarding, which is controlled by q_f, amplifies how correct processes react to ENDORSE messages, and is instrumental to ensure the algorithm's liveness. As soon as some critical "mass" of agreeing ENDORSE messages accumulates within the system, forwarding triggers a chain reaction which guarantees that a minimum number of correct processes eventually $k2\ell$-deliver the corresponding app-message.

More concretely, our algorithm provides an object (SigFreeK2LCast, Alg. 1), instantiated using the function SigFreeK2LCast(q_d, q_f, *single*), using three input parameters:

- q_d: the number of matching ENDORSE messages that must be received from distinct processes in order to $k2\ell$-deliver an app-message.

[7] In this paper, a quorum is a set of processes that (at the implementation level) ur-broadcast the same message. This definition takes quorums in their ordinary sense. In a deliberative assembly, a quorum is the minimum number of members that must vote the same way for an irrevocable decision to be taken. Let us notice that this definition does not require quorum intersection. However, if quorums have a size greater than $\frac{n+t}{2}$, the intersection of any two quorums contains, despite Byzantine processes, at least one correct process [11, 29].

[8] Another $k2\ell$-cast implementation, which uses digital signatures and allows to reach optimal values for k and ℓ, is presented in Section 6.

- q_f: the number of matching ENDORSE messages that must be received from distinct processes for the local p_i to endorse the corresponding app-message (if it has not yet).
- *single*: a Boolean that controls whether a given correct process can endorse different app-messages for the same identity *id* (*single* = `false`), or not (*single* = `true`).

The algorithm provides the operations $k2\ell$_cast and $k2\ell$_deliver. Given an app-message m with identity *id*, the operation $k2\ell$_cast(m, id) ur-broadcasts ENDORSE(m, id) provided p_i has not yet endorsed any different app-message for the same identity *id* (lines 2-4). When p_i receives a message ENDORSE(m, id), its executes two steps. If the forwarding quorum q_f has been reached, p_i first retransmits ENDORSE(m, id) (Forwarding step, lines 6-8). Then, if the $k2\ell$-delivery quorum q_d is attained, p_i $k2\ell$-delivers m (Delivery step, lines 9-11).

For brevity, we define $\alpha = n + q_f - t - d - 1$. Given an execution defined by the system parameters n, t, d, and c, Alg. 1 requires the following assumptions to hold for the input parameters q_f and q_d of a $k2\ell$-cast instance (a global picture linking all parameters is presented in Fig. 1). The prefix "sf" stands for signature-free.

- sf-$k2\ell$-Assumption 1: $c - d \geq q_d \geq q_f + t \geq 2t + 1$,
- sf-$k2\ell$-Assumption 2: $\alpha^2 - 4(q_f - 1)(n - t) \geq 0$,
- sf-$k2\ell$-Assumption 3: $\alpha(q_d - 1) - (q_f - 1)(n - t) - (q_d - 1)^2 > 0$,
- sf-$k2\ell$-Assumption 4: $\alpha(q_d - 1 - t) - (q_f - 1)(n - t) - (q_d - 1 - t)^2 \geq 0$.

In particular, the safety of Alg. 1 algorithm relies solely on sf-$k2\ell$-Assumption 1, while its liveness relies on all four of them. sf-$k2\ell$-Assumption 2 through 4 constrain the solutions of a second-degree inequality resulting from the combined action of the MA, the Byzantine processes, and the message-forwarding behavior of Alg. 1. We show in the extended version that, in practical cases, these assumptions can be satisfied by a bound of the form $n > \lambda t + \xi d + f(t, d)$, where $\lambda, \xi \in \mathbb{N}$ and $f(t, 0) = f(0, d) = 0$. Together, the assumptions allow Alg. 1 to provide a $k2\ell$-cast abstraction (with values of the parameters k', k, ℓ, and δ defining a specific $k2\ell$-cast instance) as stated by the following theorem.

▶ **Theorem 1** ($k2\ell$-CORRECTNESS)**.** *If sf-$k2\ell$-Assumptions* 1–4 *are verified, Alg.* 1 *implements $k2\ell$-cast with the following guarantees:*

- $k2\ell$-VALIDITY *with* $k' = q_f - n + c$,
- $k2\ell$-NO-DUPLICATION,
- $k2\ell$-CONDITIONAL-NO-DUPLICITY *with* $\delta = \left(q_f > \dfrac{n+t}{2}\right) \vee \left(single \wedge q_d > \dfrac{n+t}{2}\right)$,
- $k2\ell$-LOCAL-DELIVERY *with* $k = \left\lfloor \frac{c(q_f-1)}{c-d-q_d+q_f} \right\rfloor + 1$,
- $\begin{cases} \text{if } single = \texttt{false}, & k2\ell\text{-WEAK-GLOBAL-DELIVERY} \\ \text{if } single = \texttt{true}, & k2\ell\text{-STRONG-GLOBAL-DELIVERY} \end{cases}$ *with* $\ell = \left\lceil c\left(1 - \frac{d}{c-q_d+1}\right)\right\rceil$.

3.3 Proof of Algorithm 1

The proofs of the $k2\ell$-cast safety properties stated in Theorem 1 ($k2\ell$-VALIDITY, $k2\ell$-NO-DUPLICATION, and $k2\ell$-CONDITIONAL-NO-DUPLICITY) are fairly straightforward. To save space, these proofs are provided in the extended version.

The proofs of the $k2\ell$-cast liveness properties ($k2\ell$-LOCAL-DELIVERY, $k2\ell$-WEAK-GLOBAL-DELIVERY, $k2\ell$-STRONG-GLOBAL-DELIVERY) are sketched informally below (Lemmas 2-10). Their full development can be found in Appendix A.

When seeking to violate the liveness properties of $k2\ell$-cast, the attacker can use the MA to control in part how many ENDORSE messages are received by each correct process, thus interfering with the quorum mechanisms defined by q_d and q_f. To analyze the joint effect of this interference with Byzantine faults, our proofs consider seven well-chosen subsets of correct processes (A, B, C, U, F, NF, and NB, depicted in Fig. 2a).

These subsets are defined for an execution of Alg. 1 in which k_I correct processes $k2\ell$-cast (m, id) (the I in k_I is for "Initial"), and ℓ_e correct processes receive at least q_d message ENDORSE(m, id). The first three subsets, A, B, and C, partition correct processes based on the number of ENDORSE(m, id) messages they receive.

- A contains the ℓ_e correct processes that receive at least q_d ENDORSE(m, id) messages (be it from correct or from Byzantine processes), and thus $k2\ell$-deliver some message.[9]
- B contains the correct processes that receive at least q_f but less than q_d ENDORSE(m, id) messages and thus do not $k2\ell$-deliver (m, id).
- C contains the remaining correct processes that receive less than q_f ENDORSE(m, id) messages. They neither forward nor deliver any message for identity id (since $q_f \leq q_d$).

In our proofs, we count how many messages ENDORSE(m, id) ur-broadcast by correct processes are received by the processes of A (resp. B and C). We note these quantities w_A^c, w_B^c, and w_C^c, and use them to bootstrap our proofs using bounds on messages (see below).

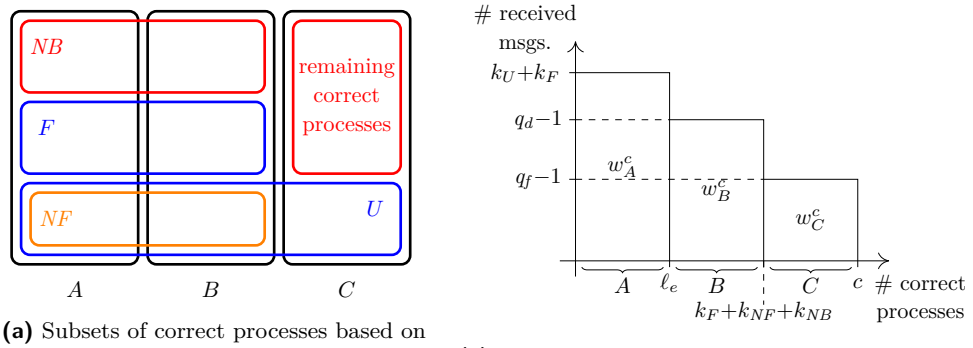
The last four subsets intersect with A, B and C, and distinguish correct processes based on the ur-broadcast operations they perform.

- U consists of the correct processes that ur-broadcast ENDORSE(m, id) at line 3.
- F denotes the correct processes of $A \cup B$ that ur-broadcast ENDORSE(m, id) at line 7 (i.e., they perform forwarding).
- NF denotes the correct processes of $A \cup B$ that ur-broadcast ENDORSE(m, id) at line 3.
- NB denotes the correct processes of $A \cup B$ that never ur-broadcast ENDORSE(m, id), be it at line 3 or at line 7. These processes have received at least q_f messages ENDORSE(m, id), but do not forward ENDORSE(m, id), because they have already ur-broadcast ENDORSE(m', id) at line 3 or at line 7 for an app-message $m' \neq m$.

Proof strategy. We note $k_U = |U|$, $k_F = |F|$, $k_{NF} = |NF|$, $k_{NB} = |NB|$. Observe that $k_U \leq k_I$ and $k_{NF} \leq k_U$, since all (correct) processes in U and NF invoke $k2\ell$_cast. Also, $(k_U + k_F)$ represents the total number of correct processes that ur-broadcast a message ENDORSE(m, id). Fig. 2b illustrates how these quantities constrain the distribution of ENDORSE messages across A, B and C. Our core proof strategy consists in bounding the areas shown in Fig. 2b. (For instance, observe that $w_A^c \leq |A| \times (k_U + k_F)$, since each of the ℓ_e correct processes in A can receive at most one ENDORSE message from each of the $(k_U + k_F)$ correct processes that send them.) This reasoning on bounds yields a polynomial involving $\ell_e = |A|$, k_I, and k_U, whose roots can then be constrained to yield the liveness guarantees required by the $k2\ell$-cast specification.

Observation. In the same way we have bounded w_A^c, we can also bound w_B^c by observing that there are $(k_{NF} + k_{NB} + k_F - \ell_e)$ processes in B and that each can receive at most $q_d - 1$ ENDORSE messages. Similarly, we can bound w_C^c by observing that the $(c - k_{NF} - k_{NB} - k_F)$ processes of C can receive at most $q_f - 1$ ENDORSE messages. Thus:

[9] Because of the condition at line 9, these processes do not necessarily $k2\ell$-deliver (m, id), but all do $k2\ell$-deliver an app-message for identity id.



(a) Subsets of correct processes based on the number of received ENDORSE messages (A, B and C) and based on their ur-broadcast actions (U, F, NF, and NB).

(b) Distribution of ENDORSE messages among correct processes of A, B, and C, sorted by decreasing numbers of ENDORSE messages received.

Figure 2 Subsets of correct processes and distribution of ENDORSE messages among them.

$$w_A^c \leq (k_U + k_F)\ell_e, \tag{1}$$
$$w_B^c \leq (q_d - 1)(k_{NF} + k_{NB} + k_F - \ell_e), \tag{2}$$
$$w_C^c \leq (q_f - 1)(c - k_{NF} - k_{NB} - k_F). \tag{3}$$

Moreover, the MA cannot suppress more than d copies of each individual ENDORSE message ur-broadcast to the c correct processes. Thus, the total number of ENDORSE messages received by correct processes ($w_A^c + w_B^c + w_C^c$) is such that:

$$w_A^c + w_B^c + w_C^c \geq (k_U + k_F)(c - d). \tag{4}$$

▶ **Lemma 2.** $\ell_e \times (k_U + k_F - q_d + 1) \geq (k_U + k_F)(c - d - q_d + q_f) - c(q_f - 1) - k_{NB}(q_d - q_f)$.

Proof sketch. We get this result by combining (1), (2), (3) and (4), and using sf-$k2\ell$-Assumption 1 with the fact that $k_{NF} \leq k_U$. (Full derivations in Appendix A.) ◀

▶ **Lemma 3.** *If no correct process $k2\ell$-casts (m', id) with $m' \neq m$, then no correct process forwards ENDORSE(m', id) at line 7 (and then $k_{NB} = 0$).* (Proof in Appendix A.)

▶ **Lemma 4** ($k2\ell$-LOCAL-DELIVERY)**.** *If at least $k = \left\lfloor \frac{c(q_f-1)}{c-d-q_d+q_f} \right\rfloor + 1$ correct processes $k2\ell$-cast an app-message m with identity id and no correct process $k2\ell$-casts any app-message m' with identity id such that $m \neq m'$, then at least one correct process p_i $k2\ell$-delivers m with identity id.*

Proof sketch. From the hypotheses, Lemma 3 helps us determine that $k_{NB} = 0$. Then, the property is proved by contraposition, by assuming that no correct process $k2\ell$-delivers (m, id), which leads us to $\ell_e = 0$. Using prior information and sf-$k2\ell$-Assumption 1, we can rewrite the inequality of Lemma 2 to get the threshold of $k2\ell$-casts above which there is at least one $k2\ell$-delivery. (Full derivations in Appendix A.) ◀

▶ **Lemma 5.** $(single = \texttt{false}) \implies (k_{NB} = 0)$. (Proof in Appendix A.)

▶ **Lemma 6.** *If at least one correct process $k2\ell$-delivers (m, id) and $x = k_U + k_F$ (the number of correct processes that ur-broadcast ENDORSE(m, id) at line 3 or 7), then $x \geq q_d - t$ and $x^2 - x(c - d + q_f - 1 - k_{NB}) \geq -(c - k_{NB})(q_f - 1)$.*

Proof sketch. We prove this lemma by counting the total number of messages (sent by Byzantine or correct processes) that are received by the processes of A, and by using (1), (3) (4), and sf-$k2\ell$-Assumption 1. (Full derivations in Appendix A.) ◀

▶ **Lemma 7.** *If $k_{NB} = 0$, and at least one correct process $k2\ell$-delivers (m, id), then $k_U + k_F \geq q_d$.*

Proof sketch. Given that $k_{NB} = 0$, we can rewrite the inequality of Lemma 6, which gives us a second-degree polynomial (where $x = k_U + k_F$ is the unknown variable). We compute its roots and show that the smaller one contradicts Lemma 6, and that the larger one is greater than or equal to q_d. The fact that x must be greater than or equal to the larger root to satisfy Lemma 6 proves the lemma. (Full derivations in Appendix A.) ◀

▶ **Lemma 8.** *If $k_{NB} = 0$ and $k_U + k_F \geq q_d$, then at least $\left\lceil c\left(1 - \frac{d}{c-q_d+1}\right)\right\rceil$ correct processes $k2\ell$-deliver some app-message with identity id (not necessarily m).*

Proof sketch. From the hypotheses, we can rewrite the inequality of Lemma 2 to get a lower bound on ℓ_e. Using sf-$k2\ell$-Assumption 3, we can determine that this lower bound is decreasing with the number of ur-broadcasts by correct processes ($x = k_U + k_F$). Hence, this lower bound is minimum when x is maximum, that is, when $x = c$. This gives us the minimum number of correct processes that $k2\ell$-deliver under the given hypotheses. (Full derivations in Appendix A.) ◀

▶ **Lemma 9** ($k2\ell$-Weak-Global-delivery)**.** *If single* = `false`*, and a correct process $k2\ell$-delivers an app-message m with identity id, then at least $\ell = \left\lceil c\left(1 - \frac{d}{c-q_d+1}\right)\right\rceil$ correct processes $k2\ell$-deliver an app-message m' with identity id (each possibly different from m).*

Proof sketch. As *single* = `false` and one correct process $k2\ell$-delivers (m, id), Lemmas 5 and 7 apply, and we have $k_{NB} = 0$ and $k_U + k_F \geq q_d$. This provides the prerequisites for Lemma 8, which concludes the proof. (Full derivations in Appendix A.) ◀

▶ **Lemma 10** ($k2\ell$-Strong-Global-delivery)**.** *If single* = `true`*, and a correct process $k2\ell$-delivers an app-message m with identity id, and no correct process $k2\ell$-casts an app-message $m' \neq m$ with identity id, then at least $\ell = \left\lceil c\left(1 - \frac{d}{c-q_d+1}\right)\right\rceil$ correct processes $k2\ell$-deliver m with identity id.*

Proof sketch. As *single* = `true`, Lemma 3 holds and implies that $k_{NB} = 0$. As above, Lemma 7 and Lemma 8 hold, yielding the lemma. (Full derivations in Appendix A.) ◀

4 BRB in the Presence of Message Adversary (MBRB): Definition

Before using the $k2\ell$-cast abstraction to reconstruct MA-tolerant BRB algorithms, we first specify what a Byzantine- and MA-tolerant broadcast should precisely achieve. We call such a broadcast an MBR-broadcast (for Message-adversarial Byzantine Reliable Broadcast), or MBRB for short. The MBRB abstraction provides two matching operations, mbrb_broadcast and mbrb_deliver. It is a multishot abstraction, i.e, it associates an identity $\langle sn, i\rangle$ (sequence number, sender identity) with each app-message, and assumes that correct processes never reuse the same sequence number for different mbrb_broadcast invocations.

When, at the application level, a process p_i invokes mbrb_broadcast(m, sn), where m is the app-message, we say it "mbrb-broadcasts (m, sn)". Similarly, when the invocation of mbrb_deliver by p_i returns the tuple (m, sn, j) to the client application (where p_j is the sender process), we say it "mbrb-delivers (m, sn, j)". So, the app-message are *mbrb-broadcast* and *mbrb-delivered*. Because of the MA, we cannot always guarantee that an app-message mbrb-delivered by a correct process is eventually received by all correct processes. Hence, in the MBR-broadcast specification, we introduce a variable ℓ_{MBRB} (reminiscent of the ℓ of $k2\ell$-cast) which indicates the strength of the global delivery guarantee of the primitive: if one correct process mbrb-delivers an app-message, then ℓ_{MBRB} correct processes eventually mbrb-deliver this app-message[10]. MBRB is defined by the following properties:

- Safety:
  - MBRB-Validity. If a correct process p_i mbrb-delivers an app-message m from a correct process p_j with sequence number sn, then p_j mbrb-broadcast m with sequence number sn.
  - MBRB-No-duplication. A correct process p_i mbrb-delivers at most one app-message from a process p_j with sequence number sn.
  - MBRB-No-duplicity. No two distinct correct processes mbrb-deliver different app-messages from a process p_i with the same sequence number sn.
- Liveness:
  - MBRB-Local-delivery. If a correct process p_i mbrb-broadcasts an app-message m with sequence number sn, then at least one correct process p_j eventually mbrb-delivers m from p_i with sequence number sn.
  - MBRB-Global-delivery. If a correct process p_i mbrb-delivers an app-message m from a process p_j with sequence number sn, then at least ℓ_{MBRB} correct processes mbrb-deliver m from p_j with sequence number sn.

It is implicitly assumed that a correct process does not use the same sequence number twice. Let us observe that, as at the implementation level, the MA can always suppress all the messages sent to a fixed set D of d processes, these mbrb-delivery properties are the strongest that can be implemented. More generally, the best-guaranteed value for ℓ_{MBRB} is $c - d$. So, the previous specification boils down to Bracha's specification [11] for $\ell_{MBRB} = c$.

5 $k2\ell$-Cast in Action: From Classical BRB to MA-Tolerant BRB (MBRB) Algorithms

This section uses $k2\ell$-cast to reconstruct two signature-free BRB algorithms [11, 20] initially introduced in a pure Byzantine context (i.e., without any MA). This reconstruction produces Byzantine-MA-tolerant versions of the initial algorithms that implement the MBRB specification of Section 4. Moreover, when $d = 0$, our two reconstructed BRB algorithms are strictly more efficient than the original algorithms that gave rise to them (they terminate earlier).

More precisely, the original and reconstructed versions of Bracha's BRB are identical in terms of communication cost, time complexity, and t-resilience (when $d = 0$). The same comparison holds for the original and reconstructed versions of Imbs and Raynal's BRB. However, both reconstructed BRB algorithms use smaller quorums than their original versions, and therefore require fewer messages to progress. In an actual network, this means a lower latency in practice, as practical networks typically exhibit a long tail distribution of latencies (a phenomenon well-studied by system and networking researchers [14, 15, 34]).

[10] If there is no MA (i.e. $d = 0$), we should have $\ell_{MBRB} = c \geq n - t$.

Algorithm 2 $k2\ell$-cast-based reconstruction of Bracha's BRB algorithm (code of p_i).

init: $obj_{\text{E}} \leftarrow$ SigFreeK2LCast(q_d=$\lfloor\frac{n+t}{2}\rfloor$+1, q_f=t+1, *single*=`true`);
$obj_{\text{R}} \leftarrow$ SigFreeK2LCast(q_d=$2t$+d+1, q_f=t+1, *single*=`true`).

(1) **operation** mbrb_broadcast(m, sn) **is** ur_broadcast(INIT(m, sn)).

(2) **when** INIT(m, sn) **is** received **from** p_j **do** $obj_{\text{E}}.k2\ell$_cast(ECHO(m), (sn, j)).

(3) **when** (ECHO(m), (sn, j)) **is** $obj_{\text{E}}.k2\ell$_delivered **do** $obj_{\text{R}}.k2\ell$_cast(READY(m), (sn, i)).

(4) **when** (READY(m), (sn, j)) **is** $obj_{\text{R}}.k2\ell$_delivered **do** mbrb_deliver(m, sn, j).

To help readers familiar with the initial algorithms, we use the same message types (INIT, ECHO, READY, WITNESS) as in the original publications. It has been shown in [4] that the MBRB problem can be solved if and only if $n > 3t + 2d$.

5.1 Bracha's BRB algorithm reconstructed

Reconstructed version. Bracha's BRB algorithm comprises three phases. When a process invokes brb_broadcast(m, sn), it disseminates the app-message m an INIT message (first phase). The reception of this message by a correct process triggers its participation in a second phase implemented by the exchange of messages tagged ECHO. Finally, when a process has received ECHO messages from "enough" processes, it enters the third phase, in which READY messages are exchanged, at the end of which it brb-delivers the app-message m. Alg. 2 is a reconstructed version of the Bracha's BRB, which assumes $n > 3t + 2d + 2\sqrt{td}$.

The algorithm requires two instances of $k2\ell$-cast, denoted obj_{E} and obj_{R}, associated with the ECHO messages and the READY messages, respectively. For both these objects, the Boolean *single* is set to `true`. For the quorums, we have the following:

- obj_{E}: $q_f = t + 1$ and $q_d = \lfloor\frac{n+t}{2}\rfloor + 1$,
- obj_{R}: $q_f = t + 1$ and $q_d = 2t + d + 1$.

The integer sn is the sequence number of the app-message m mbrb-broadcast by p_i. The identity of m is consequently the pair $\langle sn, i\rangle$.

Alg. 2 provides $\ell_{MBRB} = \left\lceil c\left(1 - \frac{d}{c-2t-d}\right)\right\rceil$ under:

- B87-Assumption (for Bracha 1987): $n > 3t + 2d + 2\sqrt{td}$;

its proof of correctness can be found in the extended version.

Comparison (Table 2). When $d = 0$, both Bracha's algorithm and its reconstruction use the same quorum size for the READY phase. The quorums of the ECHO phase are however different (Table 2). As the algorithm requires $n > 3t$, we define $\Delta = n - 3t$ as the slack between the lower bound on n and the actual value of n. When considering the forwarding threshold q_f, we have $\lfloor\frac{n+t}{2}\rfloor + 1 = 2t + \lfloor\frac{\Delta}{2}\rfloor + 1 > t + 1$. As a result, the reconstruction of Bracha's algorithm always uses a lower forwarding threshold for ECHO messages than the original. It therefore forwards messages more rapidly and reaches the delivery quorum faster.

Table 2 Bracha's original version vs. $k2\ell$-cast-based reconstruction when $d = 0$.

Threshold	Original version (ECHO phase)	$k2\ell$-cast-based version (obj_{E})
Forwarding q_f	$\left\lfloor\frac{n+t}{2}\right\rfloor + 1$	$t + 1$
Delivery q_d	$\left\lfloor\frac{n+t}{2}\right\rfloor + 1$	$\left\lfloor\frac{n+t}{2}\right\rfloor + 1$

Algorithm 3 $k2\ell$-cast-based reconstruction of Imbs and Raynal's BRB algorithm (code of p_i).

init: $obj_{\text{W}} \leftarrow$ SigFreeK2LCast($q_d = \lfloor \frac{n+3t}{2} \rfloor + 3d + 1$, $q_f = \lfloor \frac{n+t}{2} \rfloor + 1$, *single*=`false`).

(1) **operation** mbrb_broadcast(m, sn) **is** ur_broadcast(INIT(m, sn)).

(2) **when** INIT(m, sn) **is** received **from** p_j **do** $obj_{\text{W}}.k2\ell$_cast(WITNESS(m), (sn, j)).

(3) **when** (WITNESS(m), (sn, j)) **is** $obj_{\text{W}}.k2\ell$_delivered **do** mbrb_deliver(m, sn, j).

5.2 Imbs and Raynal's BRB algorithm reconstructed

Reconstructed version. Imbs and Raynal's BRB is another BRB implementation, which achieves an optimal good-case latency (only two communication steps) at the cost of a non-optimal t-resilience. Its reconstructed version requires $n > 5t + 12d + \frac{2td}{t+2d}$.

The algorithm requires a single $k2\ell$-cast object, denoted obj_{W}, associated with the WITNESS message, and which is instantiated with $q_f = \lfloor \frac{n+t}{2} \rfloor + 1$ and $q_d = \lfloor \frac{n+3t}{2} \rfloor + 3d + 1$, and the Boolean *single* = `false`. Similarly to Bracha's reconstructed BRB, an identity of app-message in this algorithm is a pair $\langle sn, i \rangle$ containing a sequence number sn and a process identity i. Alg. 3 provides $\ell_{MBRB} = \left\lceil c \left(1 - \frac{d}{c - \lfloor \frac{n+3t}{2} \rfloor - 3d} \right) \right\rceil$ under:

- IR16-Assumption (for Imbs-Raynal 2016): $n > 5t + 12d + \frac{2td}{t+2d}$; (where $t + d > 0$)

its proof of correctness can be found in the extended version.

Comparison (Table 3). Table 3 compares Imbs and Raynal's original algorithm against its $k2\ell$-cast reconstruction for $d = 0$. Recall that this algorithm saves one communication step with respect to Bracha's at the cost of a weaker t-tolerance, i.e., it requires $n > 5t$. As for Bracha, let us define the slack between n and its minimum as $\Delta = n - 5t$, we have $\Delta \geq 1$.

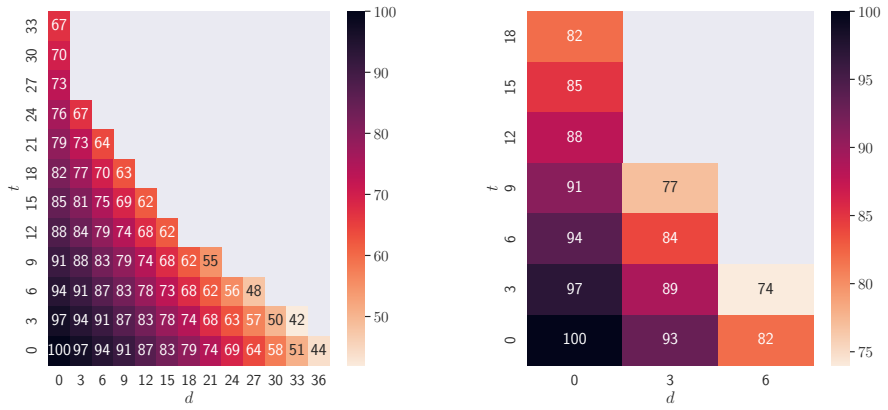
- Let us first consider the size of the forwarding quorum (first line of the table). We have $n - 2t = 3t + \Delta$ and $\lfloor \frac{n+t}{2} \rfloor + 1 = 3t + \lfloor \frac{\Delta}{2} \rfloor + 1$. When $\Delta > 2$, we always have $\Delta > \lfloor \frac{\Delta}{2} \rfloor + 1$, it follows that the forwarding predicate of the reconstructed version is equal or weaker than the one of the original version.
- The same occurs for the size of the delivery quorum (second line of the table). We have $n - t = 4t + \Delta$ and $\lfloor \frac{n+3t}{2} \rfloor + 1 = 4t + \lfloor \frac{\Delta}{2} \rfloor + 1$. So both reconstructed quorums are lower than those of the original version when $\Delta > 2$, making the reconstructed algorithm quicker as soon as $n \geq 5t + 3$. The two versions behave identically for $5t + 3 \geq n \geq 5t + 2$ ($\Delta \in \{1, 2\}$).

Table 3 Imbs and Raynal's original version vs. $k2\ell$-cast-based reconstruction when $d = 0$.

Threshold	**Original version** (WITNESS phase)	$k2\ell$**-cast-based version** (obj_{W})
Forwarding q_f	$n - 2t$	$\left\lfloor \frac{n+t}{2} \right\rfloor + 1$
Delivery q_d	$n - t$	$\left\lfloor \frac{n+3t}{2} \right\rfloor + 1$

5.3 Numerical evaluation of the MBRB algorithms

Fig. 3 provides a numerical evaluation of the delivery guarantees of both $k2\ell$-cast-based MBRB algorithms (Algs. 2 and 3) in the presence of Byzantine processes and an MA. Results were obtained for $n = 100$ and $c = n - t$, and show the values of ℓ_{MBRB} for different values of



(a) Reconstructed Bracha MBRB (Alg. 2). **(b)** Reconstructed Imbs-Raynal MBRB (Alg. 3).

Figure 3 Values of ℓ_{MBRB} for the reconstructed BRB algorithms when varying t and d ($n = 100$ and $c = n - t$) within the ranges that satisfy B87-Assumption and IR16-Assumption.

t and d. For instance, Fig. 3a shows that with 6 Byzantine processes and an MA suppressing up to 9 ur-broadcast messages, Alg. 2 ensures the MBRB-GLOBAL-DELIVERY property with $\ell_{MBRB} = 83$. The figures illustrate that the reconstructed Bracha algorithm performs in a broader range of parameter values, mirroring the bounds on n, t, and d captured by B87-Assumption and IR16-Assumption. Nonetheless, both algorithms exhibit values of ℓ_{MBRB} that can support real-world applications in the presence of an MA.

6 A Signature-Based Implementation of $k2\ell$-Cast

This section presents an implementation of $k2\ell$-cast based on digital signatures. The underlying model is the same as that of Section 2 (page 4), except that the computing power of the attacker is now bounded, which allows us to leverage asymmetric cryptography.

6.1 Algorithm

The signature-based algorithm is described in Alg. 4. It uses an asymmetric cryptosystem to sign messages and verify their authenticity. Every process has a public/private key pair. Public keys are known to everyone, but private keys are only known to their owner. (Byzantine processes may exchange their private keys.) Each process also knows the mapping between process indexes and associated public keys, and each process can produce a unique, valid signature for a given message, and check if a signature is valid.

It is a simple algorithm that ensures that an app-message must be $k2\ell$-cast by at least k correct processes to be $k2\ell$-delivered by at least ℓ correct processes. For the sake of simplicity, we say that a correct process p_i "ur-broadcasts a set of signatures" if it ur-broadcasts a BUNDLE$(m, id, sigs_i)$ in which $sigs_i$ contains the signatures at hand. A correct process p_i ur-broadcasts an app-message m with identity id at line 5 or line 11.

- If this occurs at line 5, p_i includes in the message it ur-broadcasts all the signatures it has already received for (m, id) plus its own signature.
- If this occurs at line 11, p_i has just received a message containing a set of signatures $sigs$ for the pair (m, id). The process p_i then aggregates in $sigs_i$ the valid signatures it just received with the ones it did know about beforehand (line 10).

This algorithm simply assumes: (the prefix "sb" stands for signature-based)

- sb-$k2\ell$-Assumption 1: $c > 2d$,
- sb-$k2\ell$-Assumption 2: $c - d \geq q_d \geq t + 1$.

Algorithm 4 $k2\ell$-cast implementation with signatures (code for p_i).

object SigBasedK2LCast(q_d) **is**

(1) **operation** $k2\ell$_cast(m, id) **is**
(2) **if** $\big((-, id)$ not already signed by $p_i\big)$ **then**
(3) $sig_i \leftarrow$ signature of (m, id) by p_i;
(4) $sigs_i \leftarrow \{$all valid signatures for (m, id) ur-broadcast by $p_i\} \cup \{sig_i\}$;
(5) ur_broadcast(BUNDLE$(m, id, sigs_i)$);
(6) check_delivery()
(7) **end if**.

(8) **when** BUNDLE$(m, id, sigs)$ **is** received **do**
(9) **if** $\big(sigs$ contains valid signatures for (m, id) not already ur-broadcast by $p_i\big)$ **then**
(10) $sigs_i \leftarrow \{$all valid signatures for (m, id) ur-broadcast by $p_i\}$
$\cup \{$all valid signatures for (m, id) in $sigs\}$;
(11) ur_broadcast(BUNDLE$(m, id, sigs_i)$);
(12) check_delivery()
(13) **end if**.

(14) **internal operation** check_delivery() **is**
(15) **if** $\big(p_i$ ur-broadcast at least q_d valid signatures for (m, id)
$\wedge\ (-, id)$ not already $k2\ell$-delivered$\big)$
(16) **then** $k2\ell$_deliver(m, id)
(17) **end if**.

end object.

Thanks to digital signatures, processes can relay the messages of other processes in Alg. 4. The algorithm, however, does not use forwarding in the same way Alg. 1 did: there is no equivalent of q_f here, that is, the only way to "endorse" an app-message (which, in this case, is equivalent to signing this app-message) is to invoke the $k2\ell$_cast operation. Furthermore, only one app-message can be endorsed by a correct process for a given identity (which is the equivalent of *single* = `true` in the signature-free version).

Although this implementation of $k2\ell$-cast provides better guarantees than Alg. 1, using it to reconstruct signature-free BRB algorithms would be counter-productive. This is because signatures allow for MA-tolerant BRB algorithms that are more efficient in terms of round and message complexity than those that can be constructed using $k2\ell$-cast [4].

However, a signature-based $k2\ell$-cast does make sense in contexts in which many-to-many communication patterns are required [9], and, we believe, opens the path to novel ways to handle local state resynchronization resilient to Byzantine failures and message adversaries. For instance, we are using the following algorithm in our own work to design churn-tolerant money transfer systems tolerating Byzantine failures and temporary disconnections.

6.2 Guarantees

The proof of the following theorem can be found in the extended version.

▶ **Theorem 11** ($k2\ell$-Correctness). *If sb-$k2\ell$-Assumption* 1 *and* 2 *are verified, Alg.* 4 *implements $k2\ell$-cast with the following guarantees: (i) $k' = q_d - n + c$, (ii) $k = q_d$, (iii) $\ell = c - d$, and (iv) $\delta = q_d > \frac{n+t}{2}$.*

7 Conclusion

This paper discussed reliable broadcast in asynchronous systems where an adversary can control some Byzantine processes and can suppress messages. Its starting point was the design of generic reliable broadcast abstractions suited to applications that do not require total order on the delivery of application messages (distributed money transfers are such applications [8, 10, 19]). However, the ability to thwart an adversary controlling Byzantine processes and a message adversary is new. This approach can be applied to the design of a wide range of quorum-based distributed algorithms other than reliable broadcast. For instance, we conjecture that $k2\ell$-cast could benefit self-stabilizing and self-healing distributed systems [6], where a critical mass of messages from other processes is needed in order to re-synchronize the local state of a given process.

References

1 I. Abraham, K. Nayak, L. Ren, and Z. Xiang. Good-case latency of Byzantine broadcast: a complete categorization. In *Proc. 40th ACM Symposium on Principles of Distributed Computing (PODC'21)*, pages 331–341. ACM Press, 2021.

2 I. Abraham, L. Ren, and Z. Xiang. Good-case and bad-case latency of unauthenticated Byzantine broadcast: A complete categorization. In *Proc. 25th Int'l Conference on Principles of Distributed Systems (OPODIS'21)*, pages 5:1–5:20. LIPIcs, 2021.

3 Y. Afek and E. Gafni. Asynchrony from synchrony. In *Proc. 14th Int'l Conference on Distributed Computing and Networking (ICDCN'13)*, pages 225–239. Springer, 2021.

4 T. Albouy, D. Frey, M. Raynal, and F. Taïani. Byzantine-tolerant reliable broadcast in the presence of silent churn. In *Proc. 23th Int'l Symposium on Stabilization, Safety, and Security of Distributed Systems (SSS'21)*, pages 21–33. Springer, 2021. Extended version: `arXiv:2205.09992`.

5 T. Albouy, D. Frey, M. Raynal, and F. Taïani. A modular approach to construct signature-free BRB algorithms under a message adversary, 2022. `arXiv:2204.13388`.

6 K. Altisen, S. Devismes, S. Dubois, and F. Petit. *Introduction to distributed self-stabilizing algorithms*. Morgan & Claypool, 2019.

7 H. Attiya and J. Welch. *Distributed computing: fundamentals, simulations and advanced topics*. Wiley-Interscience, 2004.

8 A. Auvolat, D. Frey, M. Raynal, and F. Taïani. Money transfer made simple: a specification, a generic algorithm, and its proof. *Bulletin of EATCS (European Association of Theoretical Computer Science)*, 132:22–43, 2020.

9 A. Auvolat, M. Raynal, and F. Taïani. Byzantine-tolerant set-constrained delivery broadcast. In *Proc. 23rd Int'l Conference on Principles of Distributed Systems (OPODIS'19)*, pages 6:1–6:23. LIPIcs, 2019.

10 M. Baudet, G. Danezis, and A. Sonnino. Fastpay: high-performance Byzantine fault tolerant settlement. In *Proc. 2nd ACM Conference on Advances in Financial Technologies (AFT'20)*, pages 163–177. ACM Press, 2020.

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

12 C. Cachin, R. Guerraoui, and L. Rodrigues. *Reliable and secure distributed programming*. Springer, 2011.

13 B. Charron-Bost and A. Schiper. The heard-of model: computing in distributed systems with benign faults. *Distributed Computing*, 22(1):49–71, 2009.

14 X. Chen, H. Song, J. Jiang, C. Ruan, C. Li, S. Wang, G. Zhang, R. Cheng, and H. Cui. Achieving low tail-latency and high scalability for serializable transactions in edge computing. In *Proc. 16th European Conference on Computer Systems (EuroSys'21)*, pages 210–227. ACM Press, 2021.

15 D. Didona and W. Zwaenepoel. Size-aware sharding for improving tail latencies in in-memory key-value stores. In *Proc. 16th USENIX Symposium on Networked Systems Design and Implementation (NSDI'19)*, pages 79–94. USENIX Association, 2019.

16 D. Dolev. The Byzantine generals strike again. *Journal of Algorithms*, 3:14–20, 1982.

17 C. Dwork, D. Peleg, N. Pippenger, and E. Upfal. Fault tolerance in networks of bounded degree. *SIAM Journal of Computing*, 17(5):975–988, 1988.

18 R. Guerraoui, J. Komatovic, P. Kuznetsov, Y.A. Pignolet, D.A. Seredinschi, and A. Tonkikh. Dynamic Byzantine reliable broadcast. In *Proc. 24th Int'l Conference on Principles of Distributed Systems (OPODIS'20)*, pages 23:1–23:18. LIPIcs, 2020.

19 R. Guerraoui, P. Kuznetsov, M. Monti, M. Pavlovic, and D.A. Seredinschi. The consensus number of a cryptocurrency. In *Proc. 38th ACM Symposium on Principles of Distributed Computing (PODC'19)*, pages 307–316. ACM Press, 2019.

20 D. Imbs and M. Raynal. Trading t-resilience for efficiency in asynchronous Byzantine reliable broadcast. *Parallel Processing Letters*, 26(4):1650017:1–1650017:8, 2016.

21 L. Lamport, R. Shostak, and M. Pease. The Byzantine generals problem. *ACM Transactions on Programming Languages and Systems*, 4(3):382–401, 1982.

22 D. Malkhi and M.K. Reiter. Byzantine quorum systems. *Distributed Computing*, 11(4):203–213, 1998.

23 A. Maurer, X. Défago, and S. Tixeuil. Communicating reliably in multi-hop dynamic networks despite Byzantine failures. In *Proc. 34th Symposium on Reliable Distributed Systems (SRDS'15)*, pages 238–245. IEEE Press, 2015.

24 A. Mostéfaoui, H. Moumen, and M. Raynal. Signature-free asynchronous byzantine consensus with $t < n/3$ and $O(n^2)$ messages. In *Proc. 33th ACM Symposium on Principles of Distributed Computing (PODC'14)*, pages 2–9. ACM Press, 2014.

25 K. Nayak, L. Ren, E. Shi, N.H. Vaidya, and Z. Xiang. Improved extension protocols for Byzantine broadcast and agreement. In *Proc. 34rd Int'l Symposium on Distributed Computing (DISC'20)*, pages 28:1–28:17. LIPIcs, 2020.

26 M. Pease, R. Shostak, and L. Lamport. Reaching agreement in the presence of faults. *Journal of the ACM*, 27:228–234, 1980.

27 M. Raynal. *Distributed algorithms for message-passing systems*. Springer, 2013.

28 M. Raynal. Message adversaries. In *Encyclopedia of Algorithms*, pages 1272–1276. Springer, 2016.

29 M. Raynal. *Fault-tolerant message-passing distributed systems: an algorithmic approach*. Springer, 2018.

30 M. Raynal and J. Stainer. Synchrony weakened by message adversaries vs asynchrony restricted by failure detectors. In *Proc. 32nd ACM Symposium on Principles of Distributed Computing (PODC'13)*, pages 166–175. ACM Press, 2013.

31 N. Santoro and P. Widmayer. Time is not a healer. In *Proc. 6th Annual Symposium on Theoretical Aspects of Computer Science (STACS'89)*, pages 304–316. Springer, 1989.

32 N. Santoro and P. Widmayer. Agreement in synchronous networks with ubiquitous faults. *Theoretical Computer Science*, 384(2-3):232–249, 2007.

33 L. Tseng, Q. Zhang, S. Kumar, and Y. Zhang. Exact consensus under global asymmetric Byzantine links. In *Proc. 40th IEEE Int'l Conference on Distributed Computing Systems (ICDCS 2020)*, pages 721–731. IEEE Press, 2020.

34 L. Yang, S.J. Park, M. Alizadeh, S. Kannan, and D. Tse. DispersedLedger: high-throughput Byzantine consensus on variable bandwidth networks. In *Prof. 19th USENIX Symposium on Networked Systems Design and Implementation (NSDI'22)*, pages 493–512. USENIX Association, 2022.

A Liveness Proof of the Signature-Free $k2\ell$-cast Implementation (Algorithm 1)

▶ **Lemma 2.** $\ell_e \times (k_U + k_F - q_d + 1) \geq (k_U + k_F)(c - d - q_d + q_f) - c(q_f - 1) - k_{NB}(q_d - q_f)$.

Proof. Combining (1), (2), (3) and (4) yields:

$$\begin{aligned}(k_U + k_F)\ell_e + (q_d - 1)(k_{NF} + k_{NB} + k_F - \ell_e) + \\ (q_f - 1)(c - k_{NF} - k_{NB} - k_F) \geq (k_U + k_F)(c - d),\\ \ell_e \times (k_U + k_F - q_d + 1) \geq (k_U + k_F)(c - d) - (q_d - 1)(k_{NF} + k_{NB} + k_F) - \\ (q_f - 1)(c - k_{NF} - k_{NB} - k_F),\\ \geq (k_U + k_F)(c - d) - (q_d - q_f)(k_{NF} + k_{NB} + k_F) - c(q_f - 1).\end{aligned}$$

Using sf-$k2\ell$-Assumption 1, we have $q_d - q_f \geq 0$. By definition, we also have $k_{NF} \leq k_U$, which yields:

$$\begin{aligned}\ell_e \times (k_U + k_F - q_d + 1) &\geq (k_U + k_F)(c - d) - (q_d - q_f)(k_U + k_F + k_{NB}) - c(q_f - 1),\\ &\geq (k_U + k_F)(c - d - q_d + q_f) - c(q_f - 1) - k_{NB}(q_d - q_f).\end{aligned}$$ ◀

▶ **Lemma 3.** *If no correct process $k2\ell$-casts (m', id) with $m' \neq m$, then no correct process forwards* ENDORSE(m', id) *at line 7 (and then $k_{NB} = 0$).*

Proof. Assume there is a correct process that ur-broadcasts ENDORSE(m', id) at line 7 with $m' \neq m$. Let us consider the first such process p_i. To execute line 7, p_i must first receive q_f messages ENDORSE(m', id) from distinct processes. Since $q_f > t$ (sf-$k2\ell$-Assumption 1), at least one of these processes, p_j, is correct. Since p_i is the first correct process to forward ENDORSE(m', id) at line 7, the ENDORSE(m', id) message of p_j must come from line 3, and p_j must have $k2\ell$-cast (m', id). We have assumed that no correct process $k2\ell$-cast $m' \neq m$, therefore $m' = m$. Contradiction.

We conclude that, under these assumptions, no correct process ur-broadcasts ENDORSE(m', id) with $m' \neq m$, be it at line 3 (by assumption) or at line 7 (shown by this proof). As a result, $k_{NB} = 0$. ◀

▶ **Lemma 4** ($k2\ell$-LOCAL-DELIVERY)**.** *If at least $k = \left\lfloor \frac{c(q_f-1)}{c-d-q_d+q_f} \right\rfloor + 1$ correct processes $k2\ell$-cast an app-message m with identity id and no correct process $k2\ell$-casts any app-message m' with identity id such that $m \neq m'$, then at least one correct process p_i $k2\ell$-delivers m with identity id.*

Proof. Let us assume that no correct process $k2\ell$-casts (m', id) with $m' \neq m$. No correct process therefore ur-broadcasts ENDORSE(m', id) with $m' \neq m$ at line 3. Lemma 3 also applies and no correct process forwards ENDORSE(m', id) with $m' \neq m$ at line 7 either, so $k_{NB} = 0$. Because no correct process ur-broadcasts ENDORSE(m', id) with $m' \neq m$ whether at line 3 or 7, a correct process receives at most t messages ENDORSE(m', id) (all coming from Byzantine processes). As by sf-$k2\ell$-Assumption 1, $t < q_d$, no correct process $k2\ell$-delivers (m', id) with $m' \neq m$ at line 10.

We now prove the contraposition of the Lemma. Let us assume no correct process $k2\ell$-delivers (m, id). Since, by our earlier observations, no correct process $k2\ell$-delivers (m', id) with $m' \neq m$ either, the condition at line 9 implies that no correct process ever receives at least q_d ENDORSE(m, id), and therefore $\ell_e = 0$. By Lemma 2 we have $c(q_f - 1) \geq (k_U + k_F)(c - d - q_d + q_f)$. sf-$k2\ell$-Assumption 1 implies that $c - d - q_d \geq 0 \iff c - d - q_d + q_f > 0$

(as $q_f \geq t+1 \geq 1$), leading to $k_U + k_F \leq \frac{c(q_f-1)}{c-d-q_d+q_f}$. Because of the condition at line 2, a correct process p_j that has $k2\ell$-cast (m, id) but has not ur-broadcast ENDORSE(m, id) at line 3 has necessarily ur-broadcast ENDORSE(m, id) at line 7. We therefore have $k_I \leq k_U + k_F$, which gives $k_I \leq \frac{c(q_f-1)}{c-d-q_d+q_f}$. By contraposition, if $k_I > \frac{c(q_f-1)}{c-d-q_d+q_f}$, then at least one correct process must $k2\ell$-deliver (m, id). Hence, we have $k = \left\lfloor \frac{c(q_f-1)}{c-d-q_d+q_f} \right\rfloor + 1$. ◀

▶ **Lemma 5.** $(single = \texttt{false}) \implies (k_{NB} = 0)$.

Proof. Let us consider a correct process $p_i \in A \cup B$. If we assume $p_i \notin F$, p_i never executes line 7 by definition. Because $p_i \in A \cup B$, p_i has received at least q_f messages ENDORSE(m, id), and therefore did not fulfill the condition at line 6 when it received its $q_f{}^{\text{th}}$ message ENDORSE(m, id). As $single = \texttt{false}$ by Lemma assumption, to falsify this condition, p_i must have had already ur-broadcast ENDORSE(m, id) when this happened. Because p_i never executes line 7, this implies that p_i ur-broadcasts ENDORSE(m, id) at line 3, and therefore $p_i \in NF$. This reasoning proves that $A \cup B \setminus F \subseteq NF$. As the sets F, NF and NB partition $A \cup B$, this shows that $NB = \emptyset$, and $k_{NB} = |\emptyset| = 0$. ◀

▶ **Lemma 6.** *If at least one correct process $k2\ell$-delivers (m, id) and $x = k_U + k_F$ (the number of correct processes that ur-broadcast ENDORSE(m, id) at line 3 or 7), then $x \geq q_d - t$ and $x^2 - x(c - d + q_f - 1 - k_{NB}) \geq -(c - k_{NB})(q_f - 1)$.*

Proof. Let us write w_A^b the total number of ENDORSE(m, id) messages from Byzantine processes received by the processes of A, and $w_A = w_A^c + w_A^b$ the total of number ENDORSE(m, id) messages received by the processes of A, whether these ENDORSE messages originated from correct or Byzantine senders. By definition, $w_A^b \leq t\ell_e$ and $w_A \geq q_d\ell_e$. By combining these two inequalities with (1) on w_A^c we obtain:

$$\begin{aligned} q_d\ell_e \leq w_A = w_A^c + w_A^b &\leq (k_U + k_F)\ell_e + t\ell_e = (t + k_U + k_F)\ell_e, \\ q_d &\leq t + k_U + k_F, && \text{(as } \ell_e > 0\text{)} \\ q_d - t &\leq k_U + k_F = x. && (5) \end{aligned}$$

This proves the first inequality of the lemma. The processes in $A \cup B$ each receive at most $k_U + k_F$ distinct ENDORSE(m, id) messages from correct processes, so we have $w_A^c + w_B^c \leq (k_{NF} + k_F + k_{NB})(k_U + k_F)$. Combined with the inequalities (3) on w_C^c and (4) on $w_A^c + w_B^c + w_C^c$ that remain valid in this case, we now have:

$$\begin{aligned} &(k_{NF} + k_F + k_{NB})(k_U + k_F) + (q_f - 1)(c - k_{NF} - k_{NB} - k_F) \geq (k_U + k_F)(c - d), \\ &(k_{NF} + k_F + k_{NB})(k_U + k_F - q_f + 1) \geq (k_U + k_F)(c - d) - c(q_f - 1). && (6) \end{aligned}$$

Let us determine the sign of $(k_U + k_F - q_f + 1)$. We derive from (5):

$$\begin{aligned} k_U + k_F - q_f + 1 &\geq q_d - t - q_f + 1 \\ &\geq 1 > 0. && \text{(as } q_d - q_f \geq t \text{ by sf-}k2\ell\text{-Assumption 1)} \end{aligned}$$

As $(k_U + k_F - q_f + 1)$ is positive and we have $k_U \geq k_{NF}$ by definition, we can transform (6) into:

$$\begin{aligned} (k_U + k_F + k_{NB})(k_U + k_F - q_f + 1) &\geq (k_U + k_F)(c - d) - c(q_f - 1), \\ (x + k_{NB})(x - q_f + 1) &\geq x(c - d) - c(q_f - 1), && \text{(as } x = k_U + k_F\text{)} \\ x^2 - x(c - d + q_f - 1 - k_{NB}) &\geq -(c - k_{NB})(q_f - 1). \end{aligned}$$ ◀

▶ **Lemma 7.** *If $k_{NB} = 0$, and at least one correct process $k2\ell$-delivers (m, id), then $k_U + k_F \geq q_d$.*

Proof. By Lemma 6 we have:

$$x^2 - x(c - d + q_f - 1 - k_{NB}) \geq -(c - k_{NB})(q_f - 1), \tag{7}$$

As (7) holds for all, values of $c \in [n-t, n]$, we can in particular consider $c = n - t$. Moreover, as by hypothesis, $k_{NB} = 0$, we have.

$$\begin{aligned} &x^2 - x(n - t - d + q_f - 1) + (q_f - 1)(n - t) \geq 0, \\ &x^2 - \alpha x + (q_f - 1)(n - t) \geq 0. \qquad \text{(by definition of } \alpha) \end{aligned} \tag{8}$$

Let us first observe that the discriminant of the second-degree polynomial in (8) is non negative, i.e. $\alpha^2 - 4(q_f - 1)(n - t) \geq 0$ by sf-$k2\ell$-Assumption 2. This allows us to compute the two real-valued roots as follows:

$$r_0 = \frac{\alpha}{2} - \frac{\sqrt{\alpha^2 - 4(q_f - 1)(n - t)}}{2} \qquad \text{and} \qquad r_1 = \frac{\alpha}{2} + \frac{\sqrt{\alpha^2 - 4(q_f - 1)(n - t)}}{2}.$$

Thus (8) is satisfied if and only if $x \leq r_0 \vee x \geq r_1$.

- Let us prove $r_0 \leq q_d - 1 - t$. We need to show that:

$$\begin{aligned} \frac{\alpha}{2} - \frac{\sqrt{\alpha^2 - 4(q_f - 1)(n - t)}}{2} &\leq q_d - 1 - t \\ \frac{\alpha}{2} - (q_d - 1) + t &\leq \frac{\sqrt{\alpha^2 - 4(q_f - 1)(n - t)}}{2} \\ \frac{\sqrt{\alpha^2 - 4(q_f - 1)(n - t)}}{2} &\geq \frac{\alpha}{2} - (q_d - 1) + t \\ \sqrt{\alpha^2 - 4(q_f - 1)(n - t)} &\geq \alpha - 2(q_d - 1) + 2t. \end{aligned}$$

  The inequality is trivially satisfied if $\alpha - 2(q_d - 1) + 2t < 0$. For all other cases, we need to verify that:

$$\begin{aligned} \alpha^2 - 4(q_f - 1)(n - t) &\geq (\alpha - 2(q_d - 1) + 2t)^2, \\ \alpha^2 - 4(q_f - 1)(n - t) &\geq \alpha^2 + 4(q_d - 1)^2 + 4t^2 - 4\alpha(q_d - 1) + 4\alpha t - 8t(q_d - 1), \\ -4(q_f - 1)(n - t) &\geq 4(q_d - 1)^2 + 4t^2 - 4\alpha(q_d - 1) + 4\alpha t - 8t(q_d - 1), \\ -(q_f - 1)(n - t) &\geq (q_d - 1)^2 + t^2 - \alpha(q_d - 1) + \alpha t - 2t(q_d - 1), \\ -(q_f - 1)(n - t) &\geq (q_d - 1 - t)^2 - \alpha(q_d - 1 - t), \end{aligned}$$

  and thus $\alpha(q_d - 1 - t) - (q_f - 1)(n - t) - (q_d - 1 - t)^2 \geq 0$, which is true by sf-$k2\ell$-Assumption 4.
- Let us prove $r_1 > q_d - 1$. We want to show that:

$$\frac{\alpha}{2} + \frac{\sqrt{\alpha^2 - 4(q_f - 1)(n - t)}}{2} > q_d - 1$$

  Let us rewrite the inequality as follows:

$$\begin{aligned} \alpha + \sqrt{\alpha^2 - 4(q_f - 1)(n - t)} &> 2(q_d - 1) \\ \sqrt{\alpha^2 - 4(q_f - 1)(n - t)} &> 2(q_d - 1) - \alpha \end{aligned}$$

The inequality is trivially satisfied if $2(q_d-1)-\alpha<0$. For all other cases, we can take the squares as follows:

$$\begin{aligned}\alpha^2-4(q_f-1)(n-t) &> (2(q_d-1)-\alpha)^2,\\ \alpha^2-4(q_f-1)(n-t) &> 4(q_d-1)^2+\alpha^2-4\alpha(q_d-1),\\ -4(q_f-1)(n-t) &> 4(q_d-1)^2-4\alpha(q_d-1),\\ 4\alpha(q_d-1)-4(q_f-1)(n-t)-4(q_d-1)^2 &> 0,\\ \alpha(q_d-1)-(q_f-1)(n-t)-(q_d-1)^2 &> 0,\end{aligned}$$

which is true by sf-$k2\ell$-Assumption 3.

We now know that $r_0 \leq q_d-1-t$ and that $r_1 > q_d-1$. In addition, as $x \leq r_0 \vee x \geq r_1$, we have $x \leq q_d-t-1 \vee x > q_d-1$. But Lemma 6 states that $x \geq q_d-t$, which is incompatible with $x \leq q_d-t-1$. So we are left with $x > q_d-1$, which implies, as q_d and x are integers that $x \geq q_d$, thus proving the lemma for $c=n-t$.

Let us now consider the set E_0 of all executions in which t processes are Byzantine, and therefore $c=n-t$, and a set E_c of executions in which there are fewer Byzantine processes, and thus $c>n-t$ correct processes. We show that $E_c \subseteq E_0$ in that a Byzantine process can always simulate the behavior of a correct process. In particular, if the simulated correct process is not subject to the message adversary, the simulating Byzantine process simply operates like a correct process. If, on the other hand, the simulated correct process misses some messages as a result of the message adversary, the Byzantine process can also simulate missing such messages. As a result, the executions that can happen when $c>n-t$ can also happen when $c=n-t$. Thus our result proven for $c=n-t$ can be extended to all possible values of c. ◀

▶ **Lemma 8.** *If $k_{NB}=0$ and $k_U+k_F \geq q_d$, then at least $\left\lceil c\left(1-\frac{d}{c-q_d+1}\right)\right\rceil$ correct processes $k2\ell$-deliver some app-message with identity id (not necessarily m).*

Proof. As $k_{NB}=0$ and $k_U+k_F \geq q_d$, we can rewrite the inequality of Lemma 2 into:

$$\ell_e \times (k_U+k_F-q_d+1) \geq (k_U+k_F)(c-d-q_d+q_f)-c(q_f-1).$$

From $k_U+k_F \geq q_d$ we derive $k_U+k_F-q_d+1>0$, and we transform the above inequality into:

$$\ell_e \geq \frac{(k_U+k_F)(c-d-q_d+q_f)-c(q_f-1)}{k_U+k_F-q_d+1}.$$

Let us now focus on the case in which $c=n-t$, we obtain:

$$\ell_e \geq \frac{(k_U+k_F)(n-t-d-q_d+q_f)-(n-t)(q_f-1)}{k_U+k_F-q_d+1}.$$

The right side of the inequality is of the form:

$$\ell_e \geq \frac{\phi x-\beta}{x-\gamma} = \phi+\frac{\phi\gamma-\beta}{x-\gamma} \tag{9}$$

with:

$$\begin{aligned}x &= k_U+k_F,\\ \gamma &= q_d-1,\\ \alpha &= n-t-d+q_f-1,\\ \phi &= n-t-d-q_d+q_f,\\ \beta &= c(q_f-1).\end{aligned}$$

Since, by hypothesis, $x = k_U + k_F \geq q_d$, we have:

$$x - \gamma = k_U + k_F - q_d + 1 > 0. \tag{10}$$

We also have:

$$\begin{aligned}\phi\gamma - \beta &= (\alpha - \gamma)\gamma - c(q_f - 1) = \alpha\gamma - \gamma^2 - c(q_f - 1),\\ &= \alpha(q_d - 1) - (q_d - 1)^2 - (n - t)(q_f - 1) > 0, && \text{(by sf-}k2\ell\text{-Assumption 3)}\\ \phi\gamma - \beta &> 0. && (11)\end{aligned}$$

Injecting (10) and (11) into (9), we conclude that $\phi + \frac{\phi\gamma-\beta}{x-\gamma}$ is a *decreasing hyperbole* defined over $x \in]\gamma, \infty]$ with *asymptotic value* ϕ when $x \to \infty$. As x is a number of correct processes, $x \leq c$. The decreasing nature of the right-hand side of (9) leads us to: $\ell_e \geq \phi + \frac{\phi\gamma-\beta}{c-\gamma} = \frac{\phi c-\beta}{c-\gamma} \geq \frac{c(c-d-q_d+q_f)-c(q_f-1)}{c-q_d+1} \geq c \times \frac{c-d-q_d+1}{c-q_d+1} = c\left(1 - \frac{d}{c-q_d+1}\right)$.

Since ℓ_e is a positive integer, we conclude that at least $\ell_{\mathsf{min}} = \left\lceil c\left(1 - \frac{d}{c-q_d+1}\right)\right\rceil$ correct processes receive at least q_d message ENDORSE(m, id) at line 9. As each of these processes either $k2\ell$-delivers (m, id) when this first happens, or has already $k2\ell$-delivered another app-message $m' \neq m$ with identity id, we conclude that at least ℓ_{min} correct processes $k2\ell$-deliver some app-message (whether it be m or $m' \neq m$) with identity id when $c = n - t$. The reasoning for extending this result to any value of $c \in [n - t, n]$ is identical to the one at the end of the proof of Lemma 7 just above. ◀

▶ **Lemma 9** ($k2\ell$-WEAK-GLOBAL-DELIVERY)**.** *If single* = `false`*, and a correct process $k2\ell$-delivers an app-message m with identity id, then at least $\ell = \left\lceil c\left(1 - \frac{d}{c-q_d+1}\right)\right\rceil$ correct processes $k2\ell$-deliver an app-message m' with identity id (each possibly different from m).*

Proof. Let us assume *single* = `false`, and one correct process $k2\ell$-delivers (m, id). By Lemma 5, $k_{NB} = 0$. The prerequisites for Lemma 7 are verified, and therefore $k_U + k_F \geq q_d$. This provides the prerequisites for Lemma 8, from which we conclude that at least $\ell = \left\lceil c\left(1 - \frac{d}{c-q_d+1}\right)\right\rceil$ correct processes $k2\ell$-deliver an app-message m' with identity id, which concludes the proof of the lemma. ◀

▶ **Lemma 10** ($k2\ell$-STRONG-GLOBAL-DELIVERY)**.** *If single* = `true`*, and a correct process $k2\ell$-delivers an app-message m with identity id, and no correct process $k2\ell$-casts an app-message $m' \neq m$ with identity id, then at least $\ell = \left\lceil c\left(1 - \frac{d}{c-q_d+1}\right)\right\rceil$ correct processes $k2\ell$-deliver m with identity id.*

Proof. Let us assume that *(i) single* = `true`, *(ii)* no correct process $k2\ell$-casts (m', id) with $m' \neq m$, and *(iii)* one correct process $k2\ell$-delivers (m, id). Lemma 3 holds and implies that $k_{NB} = 0$. From there, as above, Lemmas 7 and 8 hold, and at least $\ell = \left\lceil c\left(1 - \frac{d}{c-q_d+1}\right)\right\rceil$ correct processes $k2\ell$-deliver an app-message for identity id.

By hypothesis, no correct process ur-broadcasts ENDORSE(m', id) at line 3 with $m' \neq m$. Similarly, because of Lemma 3, no correct process ur-broadcasts ENDORSE(m', id) at line 7 with $m' \neq m$. As a result, a correct process can receive at most receive t messages ENDORSE(m', id) at line 9 (all from Byzantine processes). As $q_d > t$ (by sf-$k2\ell$-Assumption 1), the condition of line 9 never becomes true for $m' \neq m$, and as result no correct process delivers an app-message $m' \neq m$ with identity id. All processes that $k2\ell$-deliver an app-message with identity id, therefore, $k2\ell$-deliver m, which concludes the lemma. ◀