Beyond Single-Threshold Security: A Multi-Threshold Characterization of Byzantine Reliable Broadcast Protocols

***Abstract*—Byzantine Reliable Broadcast (BRB) is a foundational primitive for fault-tolerant distributed systems, underpinning consensus protocols, blockchain systems, and secure distributed storage. Existing BRB algorithms largely follow an all-or-nothing security model: once the number of Byzantine faults exceeds a single threshold, all guarantees are simultaneously lost. Multi-threshold BRB refines this model by assigning separate resilience thresholds t_v, t_c, and t_t for validity, consistency, and termination respectively, enabling protocols whose guarantees degrade gracefully with the level of corruption.**

This paper presents a comprehensive multi-threshold analysis of asynchronous, error-free BRB protocols, including Bracha's algorithm, Imbs–Raynal, the $(2, 3)$- and $(2, 4)$-BRB protocols, and COOL. For each protocol, we derive complete multi-threshold resilience characterizations under the most general setting where all thresholds are independent parameters. This yields unified correctness proofs and explicit resilience conditions summarized in a comparative table. We complement our theoretical results with an extensive experimental evaluation using the Quantas simulation framework, systematically stress-testing each protocol beyond its proven bounds to characterize how validity, consistency, and termination degrade under adversarial scenarios exceeding design assumptions.

***Index Terms*—Reliable Communication, Byzantine Failures, Message Adversary, Multi-hop Networks**

I. Introduction

Byzantine Reliable Broadcast (BRB) is a fundamental communication primitive in distributed computing that underpins many critical applications, from blockchain consensus to secure distributed storage. It allows a designated sender to disseminate a message M to a set of n nodes such that all honest nodes either eventually deliver the same message or deliver nothing, even when some nodes behave arbitrarily or maliciously—a failure model known as *Byzantine*, introduced by Lamport et al. [17]. BRB serves as a foundational building block for a wide range of protocols, including BFT consensus, blockchain systems, and asynchronous cryptographic services, whose correctness and performance critically depend on the underlying broadcast guarantees.

The first BRB protocol was introduced by Bracha [8]. It achieves error-free security in an asynchronous message-passing model, tolerating up to t Byzantine nodes among n total nodes provided $n > 3t$, at the cost of $\mathcal{O}(n^2|M|)$ total communication. Almost two decades later, Cachin and Tessaro [10] showed how to reduce this overhead dramatically by combining erasure coding with cryptographic hashes. Their protocol sends only $\mathcal{O}(n|M| + \kappa n^2 \log n)$ bits, where κ is the hash output length. Subsequent works refined this blueprint using erasure or error-correcting codes, Merkle trees, and more advanced cryptographic tools to push communication closer to known lower bounds and to improve latency.

Existing BRB protocols can be classified along three main axes. The first axis concerns timing assumptions. Asynchronous protocols make no assumptions about message delays and do not rely on clocks or timeouts [9], [8]. In contrast, synchronous protocols [2], [12] assume known delay bounds and proceed in lock-step rounds. Partially synchronous models lie between these extremes, assuming that timing bounds hold only after some unknown stabilization point, enabling strong safety with good performance once the network behaves well. The second axis involves cryptographic assumptions. Error-free protocols [8], [4], [1], [16] rely solely on authenticated channels and remain secure against unbounded adversaries, but typically incur higher communication costs. Assuming collision-resistant hashes enables commitment-based and erasure-coded designs with significantly lower bandwidth. Stronger assumptions permit the use of public-key and threshold cryptography [18], [4], allowing nodes to generate compact, non-repudiable proofs or aggregate signatures. The third axis relates to network structure. Most classical BRB protocols assume a fully connected network [9], [2], [3], [1], [13], which simplifies design and analysis. Other work studies BRB over general graph networks [7], [5], [6], where communication is restricted to graph edges and efficiency depends on properties such as connectivity, diameter, and the availability of node- or edge-disjoint paths.

Despite steady progress toward the lower bounds, a gap remains between known impossibility results and the concrete performance of practical BRB protocols. As shown in [18], the lower bound for communication complexity for BRB is $\Omega(n|M| + n^2)$, since all honest nodes must learn M and even broadcasting a single bit requires $\Omega(n^2)$ communication in the worst case [14]. This motivates a more systematic study of BRB under various combinations of synchrony, fault assumptions, and cryptographic power, as well as a search for constructions that are simultaneously near-optimal in communication, time, and computation.

However, beyond asymptotic complexity, a fundamental question remains largely unexplored: *what happens when the number of actual Byzantine faults exceeds the design threshold?* Classical BRB protocols offer all-or-nothing security—if the adversary controls more than t nodes, all guarantees are lost. Yet in practice, systems may face scenarios where faults

temporarily exceed design assumptions, and understanding how protocols degrade under such stress is crucial for building robust distributed systems.

Multi-threshold analysis. To address this limitation, Hirt et al. [15] proposed *multi-threshold security* in the context of Bracha's algorithm, assigning separate thresholds t_v, t_c, and t_t to validity, consistency, and termination respectively. Under this framework, each property degrades independently as the number of Byzantine faults f increases: validity holds as long as $f \leq t_v$, consistency as long as $f \leq t_c$, and termination as long as $f \leq t_t$. When these thresholds coincide ($t_v = t_c = t_t = t$), the protocol reduces to the classical single-threshold analysis. Raynal [20] independently studied Bracha's BRB under a related formulation with a different proof strategy.

Good-case/Bad-case complexity. Orthogonal to resilience thresholds, Abraham, Ren, and Xiang [3], [2] classify unauthenticated BRB protocols through their *good-case* latency (when the sender is honest) and *bad-case* latency (when the sender is Byzantine). They establish a fundamental impossibility: resilience, good-case latency, and bad-case latency cannot all be simultaneously optimal. By mapping classical protocols (e.g., Bracha's and Imbs–Raynal's) into this framework and presenting two new protocols matching their lower bounds, they fully characterize the optimal trade-offs between resilience and round complexity for unauthenticated BRB.

State of the art BRB. Chen's COOL protocol [12] introduced a deterministic, error-free secure multi-valued Byzantine agreement (BA) protocol for synchronous networks, achieving optimal resilience $t < n/3$ and, for an ℓ-bit value, asymptotically optimal round complexity $\mathcal{O}(t)$ and communication complexity $\mathcal{O}(\max\{n\ell, nt \log t\})$ through carefully crafted error-correcting codes that compress information while still enabling deterministic error detection and correction. These coding and consistency techniques later became key ingredients in more communication-efficient BRB protocols. Building on COOL and prior work, Alhaddad et al. [4] developed several near-optimal asynchronous BRB protocols: a balanced multicast primitive to equalize per-node load; an error-free protocol BalEFBRB adapting COOL to asynchrony; a hash-based protocol BalCCBRB with improved computation; and a threshold-signature protocol BalSigBRB approaching the information-theoretic communication lower bound. More recently, Abraham et al. [1] revisited COOL and abstracted its consistency mechanism into a graded-dispersal primitive, yielding a new variant of COOL whose analysis is dramatically simplified while preserving the $\mathcal{O}(n|M| + n^2 \log n)$ asymptotics for an $|M|$-bit input. Their graded-dispersal abstraction gives an asynchronous BRB protocol that strictly improves on Chen's BRB [12] in both concrete communication cost (about 40% savings) and round complexity.

Our contributions. Building on this landscape, this work makes two primary contributions:

(1) Unified multi-threshold analysis. While prior research studied multi-threshold guarantees [15], [20] only for Bracha's algorithm [8], we systematically extend this analysis to several foundational BRB constructions for fully connected networks with asynchronous communication and error-free security, including Imbs–Raynal [16], (2,3)-BRB and (2,4)-BRB [2], and COOL [1]. For each protocol, we derive its complete set of multi-threshold bounds (t_v, t_c, t_t) and provide new correctness proofs that characterize exactly how guarantees degrade as the number of Byzantine faults increases. This yields the first unified comparison of multi-threshold resilience across multiple BRB algorithms, summarized in Table I.

(2) Experimental stress testing beyond theoretical bounds. We evaluate the aforementioned protocols beyond their theoretical resilience limits. Using the Quantas abstract simulation framework, we complement our theoretical analysis with extensive experiments that intentionally stress each protocol in regimes where the number of Byzantine faults exceeds the proven bounds ($f > t$). Unlike prior evaluations that restrict attention to the guaranteed-correct setting, we investigate how each protocol behaves under adversarial conditions not captured by asymptotic proofs. Our experiments vary both the number of Byzantine nodes and their equivocation strategies, including fine-grained control of the sender's equivocation ratio. In addition to these parameters, we systematically evaluate all combinations of plausible Byzantine behaviors induced by the protocol's message types, ensuring extensive coverage of the adversarial space. Across all settings, we examine whether each protocol's validity, consistency, and termination guarantees degrade gradually or fail abruptly, focusing on the fraction of honest nodes that terminate and the resulting patterns of equivocation and disagreement.

Together, these contributions bridge theoretical analysis and practical evaluation, advancing our understanding of how BRB protocols behave both within and beyond their design assumptions.

The remainder of the paper is organized as follows. Section II formalizes the system model and presents a consolidated table summarizing our results. Sections III–VI develop our theoretical results for multi-threshold BRB. Section VII presents our Quantas-based experimental evaluation across multiple scenarios. Finally, Section VIII concludes the paper.

II. System Model

Communication Model. We consider a distributed system composed of a set $V = \{p_1, \ldots, p_n\}$ of n processes (also called *nodes* or *parties*), each one identified by a unique integer identifier. Processes collaborate to run a distributed protocol $\mathcal{P}$ and we assume that the adversary has full control over the network and can arbitrarily schedule the messages. However, each message must be eventually delivered. We require our protocols to be error-free, meaning that security holds even against a computationally unbounded adversary.

Byzantine nodes. We assume that up to $f \leq t$ processes can be Byzantine faulty [17] (i.e., they can exhibit arbitrary, possibly malicious, behavior), where t represents the resilience threshold a protocol can tolerate and f is the actual number of Byzantine processes present during the execution of the

Algorithm	Cost	Rounds Good-case vs Bad-case	Resilience	Multi-threshold resilience
Error-free protocols				
Bracha [9]	$\mathcal{O}(n^2\|M\|)$	$3-4$	$n>3t$	$n>2t_t+\max(t_c,t_v)$ [15], [20]
Imbs–Raynal [16]	$\mathcal{O}(n^2\|M\|)$	$2-3$	$n>5t$	$n>4t_t+\max(t_c,t_v)$ **our paper**
(2,4)-BRB [2]	$\mathcal{O}(n^2\|M\|)$	$2-4$	$n\geq 4t$	$n\geq\max(3t_t,2)+\max(t_c,t_v)$ **our paper**
(2,3)-BRB [2]	$\mathcal{O}(n^2\|M\|)$	$2-3$	$n\geq 5t-1$	$n\geq\max(4t_t,3)+\max(t_c,t_v)-1$ **our paper**
Simple is COOL [1]	$\mathcal{O}(n\|M\|+n^2\log n)$	$6-7$	$n>3t$	$n>2t_t+\max(t_t,t_c,t_v)$ **our paper**

TABLE I
COMPARISON OF ASYNCHRONOUS BRB PROTOCOLS.

protocol. Processes that are not Byzantine faulty are said to be *correct*.

Communication Model. Processes communicate by exchanging messages. We consider a setting in which parties have access to a complete network of authenticated channels. Moreover, we consider the setting where parties do not have any setup available.

Multi-threshold resilience. Following the approach of Hirt et al. [15] and Raynal [20], all our multi-threshold resilience results satisfy a natural collapse property: when the three thresholds coincide, $t_v = t_c = t_t = t$, our definitions and guarantees reduce exactly to the classical single-threshold Byzantine Reliable Broadcast model. Thus, the multi-threshold framework used throughout this paper can be seen as a strict generalization of the standard BRB setting. In some protocols, the resilience condition branches depending on whether all thresholds are zero; accordingly, we express the requirement as a special base case for $t_v = t_c = t_t = 0$, and a general formula otherwise.

Definition 1 (Multi-Threshold Byzantine Reliable Broadcast)**.** *Let f be the number of Byzantine nodes in the network, then a protocol $\mathcal{P}$ implements a* Multi-Threshold Byzantine Reliable Broadcast *primitive if it satisfies the following properties:*

- ***Consistency.*** *If $f \leq t_c$, then every correct node that terminates outputs the same message.*
- ***Validity.*** *If $f \leq t_v$ and the sender is correct, then every correct node that terminates outputs the sender's message.*
- ***Termination.***
  1) *If $f \leq t_t$ and the sender is correct, then eventually every correct node terminates.*
  2) *If $f \leq t_t$ and a correct node terminates, then every correct node eventually terminates.*

In our setting, *termination* means that once a process delivers the broadcast value, it has completed the protocol instance and may cease further computation or message sending for that instance.

Definition 2 (Asynchronous Round)**.** *We adopt the definition of asynchronous round from [3], [11], where a protocol runs in R asynchronous rounds if its running time is at most R times the maximum message delay between honest parties during the execution.*

III. MULTI-THRESHOLD (2,4)-PROTOCOL

Let's assume:

$$n \geq \begin{cases} 0, & \text{if } t_c = t_v = t_t = 0, \\ \max(3t_t, 2) + \max(t_c, t_v), & \text{otherwise.} \end{cases}$$

Algorithm 1: Multi-Threshold (2,4)-round BRB protocol under $n \geq max(3t_t, 2) + max(t_c, t_v)$.

1. Propose.
The Designated broadcaster L with input v sends $\langle propose, v\rangle$ to all parties.

2. Ack.
When receiving the first proposal $\langle propose, v\rangle$ from the broadcaster, a party sends an $\langle ack, v\rangle$ message to all parties.

3. 2-Round Commit.
When receiving $\langle ack, v\rangle$ from $\boxed{n - t_t - 1}$ distinct non-broadcaster parties, a party commits v, sends $\langle vote1, v\rangle$ and $\langle vote2, v\rangle$ to all parties, and terminates.

4. Vote.
- When receiving $\langle ack, v\rangle$ from $\boxed{n - 2t_t}$ distinct non-broadcaster parties, a party sends a $\langle vote1, v\rangle$ message to all parties.
- When receiving $\langle vote1, v\rangle$ from $\boxed{n - t_t - 1}$ distinct non-broadcaster parties, a party sends a $\langle vote2, v\rangle$ message to all parties.
- When receiving $\langle vote2, v\rangle$ from $\boxed{max(t_c, t_v) + 1}$ distinct non-broadcaster parties, a party sends a $\langle vote2, v\rangle$ message to all parties.

5. 4-Round Commit.
When receiving $\langle vote2, v\rangle$ from $\boxed{n - t_t - 1}$ distinct non-broadcaster parties, a party commits v and terminates.

Lemma 1. *Let $f \leq max(t_c, t_v)$ and $n \geq max(3t_t, 2) + max(t_c, t_v)$. If the broadcaster p is correct and broadcasts value v, then no correct process will send any messages for $v' \neq v$.*

Proof: To prove this, we must show that no correct process sends messages of the form $\langle ack, v' \rangle$, $\langle vote1, v' \rangle$, or $\langle vote2, v' \rangle$.

No correct process sends $\langle ack, v' \rangle$. By definition, correct processes only send $\langle ack, \cdot \rangle$ messages in response to a valid *propose* message from the designated sender. Since Byzantine processes cannot forge a valid *propose* on behalf of the sender, no correct process can send $\langle ack, v' \rangle$.

No correct process sends $\langle vote1, v' \rangle$. For a correct process q to send $\langle vote1, v' \rangle$, it must receive at least $n - t_t - 1$ messages of the form $\langle ack, v' \rangle$ from non-broadcasters in Step 2 or $n - 2t_t$ in Step 3. Since $n - t_t - 1 > max(t_c, t_v)$ and $n - 2t_t > max(t_c, t_v)$, this implies that at least one correct process would need to have sent $\langle ack, v' \rangle$, which we have shown is impossible. Therefore, no correct process sends $\langle vote1, v' \rangle$.

No correct process sends $\langle vote2, v' \rangle$. Assume for contradiction that there exists a correct process q that sends a message $\langle vote2, v' \rangle$ and that it is the first to do so. Since q is correct, it can have received at most $max(t_c, t_v)$ such messages from Byzantine processes. Therefore, to send $\langle vote2, v' \rangle$, the process q must have received at least $n - t_t - 1 > \max(t_c, t_v)$ $\langle vote1, v' \rangle$. However, as shown above, no correct process ever sends a message $\langle vote1, v' \rangle$. Thus, q could not have gathered the required number of $\langle vote1, v' \rangle$ messages, yielding a contradiction.

Therefore, no correct process sends $\langle ack, v' \rangle$, $\langle vote1, v' \rangle$, or $\langle vote2, v' \rangle$. ∎

Lemma 2. *Let $f \leq max(t_c, t_v)$ and $n \geq max(3t_t, 2) + max(t_c, t_v)$. If a correct proces p commits v at Step 3, then no correct process will send $vote1$ or $vote2$ for any other value $v' \neq v$*

Proof: If the sender is correct, then by Lemma 1 no correct process will send any message for $v' \neq v$. Suppose now that the sender is Byzantine, and that there exists a correct process q that sends a message $\langle vote1, v' \rangle$. Then:

- For p to commit at Step 3, it must receive at least $n - t_t - 1$ *ack* messages for v, among which at least $(n - t_t - 1) - (f - 1) = n - t_t - f$ must come from correct processes.
- For q to send $\langle vote1, v' \rangle$, it must receive at least $n - t_t - 1$ *ack* messages for v' at Step 2 or $n - 2t_t$ at Step 3. Since there are at most $f - 1$ Byzantine non-broadcaster processes, at least $n - t_t - f$ (Step 2) and $n - 2t_t - f + 1$ (Step 3) must come from correct processes.
- The total number of correct processes in the system is at most $n - f \geq n - max(t_c, t_v)$.

In the Step 3 subcase we have $(n - t_t - f) + (n - 2t_t - f + 1) - (n - f) \geq n - 3t_t - f + 1 \geq n - 3t_t - max(t_c, t_v) + 1 \geq 1$ while in the Step 2 subcase we have $2(n - t_t - f) - (n - f) \geq n - 2t_t - f \geq n - 2t_t - max(t_c, t_v) \geq 1$. These results imply that at least one correct process must have sent *ack* messages for both v and v'. This is impossible. Therefore, no correct process sends $\langle vote1, v' \rangle$.

Finally, as in Lemma 1, if no correct process sends a *vote1* message for v', then a Byzantine sender cannot collect enough distinct messages to cause any correct process to send $\langle vote2, v' \rangle$.

∎

Lemma 3. *If $f \leq t_v$, $n \geq max(3t_t, 2) + max(t_c, t_v)$ and the sender is correct, then every correct process that terminates commits the sender's message.*

Proof: Suppose that the correct sender p proposes value v, and that there exists a correct process q that terminates and commits a value $v' \neq v$. Then q must terminate either at Step 3 or at Step 5. If q terminates at Step 3, then it must have received at least $n - t_t - 1$ messages of the form $\langle ack, v' \rangle$. Since $n - t_t - 1 > \max(t_c, t_v) \geq t_v \geq f$, at least one of these messages must have been sent by a correct process. If q terminates at Step 5, then it must have received at least $n - t_t - 1$ messages of the form $\langle vote2, v' \rangle$. Again, since $n - t_t - 1 > max(t_c, t_v) \geq t_v \geq f$, at least one of these messages must have been sent by a correct process. In both cases, this implies that some correct processes sent an *ack* or *vote2* message for $v' \neq v$, which contradicts Lemma 1. Therefore, no correct process can commit a value different from v. ∎

Lemma 4. *If $f \leq t_c$ and $n \geq max(3t_t, 2) + max(t_c, t_v)$, then every correct process that terminates commits the same message.*

Proof: If the sender is correct, then by Lemma 1 and Lemma 3, all correct processes that terminate must commit the sender's value. If instead the sender is Byzantine, we now show that it is impossible for two correct processes p and q to terminate and commit two different values v and v' with $v \neq v'$.

p and q both terminate at Step 3. For both p and q to terminate at Step 3, at least $n - t_t - f$ correct non-broadcaster processes must send $\langle ack, v \rangle$ to p and at least $n - t_t - f$ correct non-broadcaster processes must send $\langle ack, v' \rangle$ to q. Since there are at most $n - f$ correct processes in total, we would require: $2(n - t_t - f) \leq n - f$, which simplifies to $n \leq 2t_t + f \leq 2t_t + tc$. But this contradicts the assumption $n \geq max(3t_t, 2) + max(t_c, t_v)$, so this case is impossible.

p and q both terminate at Step 5. If p and q terminate at Step 5, then each must receive at least $n - t_t - 1 > max(t_c, t_v) \geq t_c \geq f$ messages of the form $\langle vote2, v \rangle$ and $\langle vote2, v' \rangle$ respectively. Thus, at least one correct process must send $\langle vote2, v \rangle$, and at least one correct process must send $\langle vote2, v' \rangle$. Let x be the first correct process to send $\langle vote2, v \rangle$ and y the first to send $\langle vote2, v' \rangle$. By the protocol rules, both x and y must have received at least $n - t_t - 1$ $\langle vote1, \cdot \rangle$ messages. As before, this would require: $2(n - t_t - f) \leq n - f$ which again implies $n \leq 2t_t + t_c$, contradicting our assumption. Thus, this case is also impossible.

p and q terminate at different steps. Assume p terminates at Step 3 committing v. Then, by Lemma 2, no correct process can send *vote1* or *vote2* messages for any value $v' \neq v$. Hence, no correct process can collect enough messages to terminate at

Step 5 with value $v' \neq v$. Conversely, suppose q terminates at Step 5 committing v'. Then q must have received a $\langle vote2, v' \rangle$ from some correct process. By the contrapositive of Lemma 2, this implies that no correct process can terminate at Step 3 with a value $v \neq v'$.

In all cases, we reach a contradiction. Therefore, if $f \leq t_c$ and $n \geq \max(3t_t, 2) + max(t_c, t_v)$, two correct processes can't commit different values. ■

Lemma 5. *If $f \leq t_t$, $n \geq max(3t_t, 2) + max(t_c, t_v)$ and the sender is correct, then every correct process eventually terminates.*

Proof: If the sender is correct, then it broadcasts $\langle propose, v \rangle$ to all processes. All $n - t_t$ correct processes will subsequently broadcast $\langle ack, v \rangle$ and each of them will terminate as soon as it receives $n - t_t - 1$ $\langle ack, v \rangle$ messages from the other correct (non-broadcaster) processes. Hence, in the good case, all correct processes terminate in 2 communication rounds. ■

Lemma 6. *If $f \leq t_t$, $n \geq max(3t_t, 2) + max(t_c, t_v)$, the sender is Byzantine and a correct process p terminates, then every correct process eventually terminates.*

Proof: If p terminates at Step 3, then it must have received at least $n - t_t - 1$ messages of the form $\langle ack, v \rangle$, of which at least $n - t_t - 1 - (f - 1) = n - t_t - f \geq n - 2t_t$ come from correct processes. Hence, every correct process will eventually receive $n - 2t_t$ messages of the form $\langle ack, v \rangle$ and will thus send $\langle vote1, v \rangle$. Consequently, all $n - f \geq n - t_t$ correct non-broadcaster processes will eventually receive the necessary $\langle vote1, v \rangle$ messages to send $\langle vote2, v \rangle$, and will commit as soon as they receive $n - t_t - 1$ such messages.

If p terminates at Step 5, then it must have received at least $n - t_t - 1$ messages of the form $\langle vote2, v \rangle$, of which at least $n - t_t - 1 - (f - 1) = n - t_t - f \geq n - 2t_t$ come from correct processes. Since $n - 2t_t \geq 1 + max(t_c, t_v)$, every correct process will eventually receive $1 + max(t_c, t_v)$ messages of the form $\langle vote2, v \rangle$, send $\langle vote2, v \rangle$ itself, and will then commit as soon as it receives $n - t_t - 1$ such messages.

It is clear from the protocol that after at most 2 rounds (*vote1* and *vote2*) since any correct process commits, all correct processes also commit. Hence, the bad-case latency is 4 rounds. ■

Theorem 1. *Let $f \leq t_c, t_v, t_t < n$, then Algorithm 1 solves Byzantine Reliable Broadcast with $n \geq max(3t_t, 2) + max(t_c, t_v)$ and an optimal good-case latency of 2 rounds, and bad-case latency of 4 rounds.*

Proof: **Validity**: This follows from Lemma 3.
Consistency: This follows from Lemma 4.
Termination 1: These follows from Lemma 5
Termination 2: This follows from Lemma 6 ■

IV. Multi-Threshold (2,3)-Protocol

Let's assume:

$$n \geq \begin{cases} 0, & \text{if } t_c = t_v = t_t = 0, \\ \max(4t_t, 3) + \max(t_c, t_v) - 1, & \text{otherwise.} \end{cases}$$

Algorithm 2: Multi-Threshold (2,3)-round BRB protocol under $n \geq max(4t_t, 3) + max(t_c, t_v) - 1$.

1. Propose.
The Designated broadcaster L with input v sends $\langle propose, v \rangle$ to all parties.

2. Ack.
- When receiving the first proposal $\langle propose, v \rangle$ from the broadcaster, a party sends $\langle ack, v \rangle$ to all parties
- When receiving $\langle ack, v \rangle$ from $\boxed{n - 2t_t}$ distinct non-broadcaster parties, a party sends $\langle ack, v \rangle$ to all parties if not yet sent $\langle ack, v \rangle$.

3. Commit.
When receiving $\langle ack, v \rangle$ from $\boxed{n - t_t - 1}$ distinct non-broadcaster parties, a party commits v and terminates.

Lemma 7. *Let $f \leq max(t_c, t_v)$ and $n \geq max(4t_t, 3) + max(t_c, t_v) - 1$. If the broadcaster p is correct and broadcasts value v, then no correct process will send any messages for $v' \neq v$.*

Proof: Suppose, for contradiction, that there exists a correct process q that sends $\langle ack, v' \rangle$ with $v' \neq v$, and let q be the first correct process to do so. Then q sends this message either (i) after receiving $\langle propose, v' \rangle$, or (ii) after collecting at least $n - 2t_t$ messages $\langle ack, v' \rangle$. *Case (i).* Impossible: the sender is correct and therefore never sends $\langle propose, v' \rangle$ for $v' \neq v$, and a Byzantine process cannot impersonate the sender. *Case (ii).* Among those $n - 2t_t$ acknowledgments, at most $f \leq \max(t_c, t_v)$ can be Byzantine. Since $n - 2t_t > \max(t_c, t_v) \geq f$, at least one of these acknowledgments must come from a correct process. That correct process sent $\langle ack, v' \rangle$ before q, contradicting the choice of q as the first correct process to do so. Hence, no correct process sends $\langle ack, v' \rangle$ with $v' \neq v$. ■

Theorem 2. *Let $f \leq t_c, t_v, t_t < n$, then Algorithm 2 solves Byzantine Reliable Broadcast with $n \geq max(4t_t, 3) + max(t_c, t_v) - 1$ and an optimal good-case latency of 2 rounds, and bad-case latency of 3 rounds.*

Proof: **Validity**: Let $f \leq t_v$, if the sender is correct and *propose* value v, then by Lemma 7 no correct process will send $\langle ack, v' \rangle$ for $v' \neq v$. This implies that no correct process can obtain $n - t_t - 1 > max(t_c, t_v) \geq f$ $\langle ack, v' \rangle$ necessary for committing v' and terminating.

Consistency: Let $f \leq t_c$. If the sender is correct, then by Lemma 7 and Validity, all correct processes that terminate output the sender's value. Assume now that the sender is Byzantine (so $f > 0$), and that among the non-broadcaster processes there are $f - 1$ Byzantine processes. Suppose, for contradiction, that two correct processes p and q

commit different values v and v', with $v \neq v'$. Then p must have received at least $(n - t_t - 1)$ messages $\langle ack, v\rangle$, of which at least $(n - t_t - f)$ are from correct processes; symmetrically, q must have received at least $(n - t_t - f)$ messages $\langle ack, v'\rangle$ from correct processes. Since there are at most $(n - f)$ correct processes, we have $2(n-t_t-f)-(n-f) \geq n-2t_t-f \geq n-2t_t-max(t_c,t_v) > 0$, which implies that there exists at least one correct process that sent $\langle ack, \cdot\rangle$ due to satisfying the second condition at Step 2. If this situation occurs only for v, then at least $n - 2t_t - (f - 1) = n - 2t_t - f + 1$ correct processes send $\langle ack, v\rangle$ due to receiving the $\langle propose, v\rangle$ message. This contradicts the fact that at least $(n - t_t - f)$ correct processes send $\langle ack, v'\rangle$ for v' due to receiving $\langle propose, v'\rangle$, since $(n - 2t_t - f + 1) + (n - t_t - f) - (n - f) \geq n - 3t_t - f + 1 \geq n - 3t_t - max(t_c, t_v) + 1 > 0$. If the above applies to both v and v', then at least $(n-2t_t-f+1)$ correct processes send $\langle ack, v\rangle$ and at least $(n-2t_t-f+1)$ correct processes send $\langle ack, v'\rangle$ due to receiving the corresponding *propose* messages. This is impossible, since $2(n - 2t_t - f + 1) - (n - f) \geq n - 4t_t - f + 2 \geq n - 4t_t - max(t_c, t_v) + 2 > 0$. Therefore, no two correct processes can commit different values, and thus all honest processes commit the same value.

Termination 1 and 2: Let $f \leq t_t$. If the sender is correct, then it broadcasts $\langle propose, v\rangle$ to all processes. All $n-f \geq n-t_t$ correct processes receive this *propose* message and subsequently send $\langle ack, v\rangle$. Eventually, each correct process (including the sender) receives $\langle ack, v\rangle$ messages from the $n-t_t-1$ other correct (non-broadcaster) processes, commits the value v, and terminates. Hence, in the good case, all correct processes terminate in 2 rounds.

Termination 3: Let $f \leq t_t$. If the sender is correct, then by Termination 1 and 2, all correct processes will terminate. If the sender is Byzantine and a correct process p commits v, then p must have received at least $(n - t_t - 1)$ messages $\langle ack, v\rangle$, of which at least $(n-t_t-f) \geq (n-2t_t)$ must come from correct processes. Therefore, every correct process will eventually satisfy the second condition in Step 2 and broadcast $\langle ack, v\rangle$ (if it has not already done so). Consequently, all $n - f \geq n-t_t$ correct processes will receive the required $\langle ack, v\rangle$ messages, commit v, and terminate. Since, once one correct process commits, all correct processes commit within at most one additional round, the bad-case latency is 3 rounds. ■

V. Multi-Threshold Imbs & Raynal Protocol

We assume $n > 4t_t + max(t_c, t_v)$, which, in the special case $t_t = t_c = t_v = 0$, collapses to the classical condition $n > 5t$.

Lemma 8. *If $f \leq max(t_c, t_v, t_t)$ and $n > 4t_t + max(t_c, t_v)$, let INIT(i, v) be a message that is never broadcast by a correct process p_i. If Byzantine processes broadcast the message WITNESS(i, v), no correct process will forward this message at line 4.*

Algorithm 3: Multi-Threshold Imbs & Raynal BRB protocol with $n > 4t_t + max(t_c, t_v)$.

Operation BRB_broadcst MSG(v_i) **is**
(1) broadcast INIT(i,v_i)

When INIT(j, v) **is received from** p_j **do**
(2) **if** *(first reception of INIT(j, v) and WITNESS($j, \cdot$) not yet broadcast)* **then**
broadcast WITNESS(j, v)
end

When WITNESS(j, v) **is received do**
(3) **if** *(WITNESS(j, v) received from $\boxed{n - 2t_t}$ different processes and WITNESS(j, v) not yet broadcast* **then**
(4) broadcast WITNESS(j, v)
end
(5) **if** *WITNESS(j, v) is received from $\boxed{n - t_t}$ different processes and MSG($j, \cdot$) not yet BRB_delivered* **then**
(6) BRB_deliver MSG(j, v) and terminate
end

Proof: For a correct process p_j to forward this message at line 4, the forwarding predicate of line 3 must be satisfied. But, for this predicate to be true at a correct process p_j, this process must receive the message WITNESS(i, v) from $n-2t_t$ different processes. As $n - 2t_t > 2t_t + max(t_c, t_v) \geq f$, this cannot occur. ■

Theorem 3. *Let $f \leq t_c, t_v, t_t < n$, then Algorithm 3 solves Byzantine Reliable Broadcast with $n > 4t_t + max(t_c, t_v)$ and an optimal good-case latency of 2 rounds, and bad-case latency of 3 rounds.*

Proof: **Validity**: Let $f \leq t_v$, and suppose a correct sender p_i broadcasts INIT(i, v). For a correct process p_j to deliver MSG(i, v') with $v' \neq v$, it must receive at least $n - t_t > f$ messages WITNESS(i, v'). However, by Lemma 8, no correct process broadcasts WITNESS(i, v') for $v' \neq v$. Therefore, such a set of $n - t_t > f$ witnesses cannot be obtained, and a correct process can only terminate by delivering MSG(i, v).

Consistency: Let $f \leq t_c$ and p_k be a process that sends at least one message INIT$(k, \cdot)$. 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 INIT(k, v_1), INIT(k, v_2), ..., 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 INIT(k, v_x), and these receptions direct them to broadcast the message WITNESS(k, v_x) at line 2. It follows from the predicate at 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 $|Q_x| < n-2t_t-t_c$. Let p_j be any correct process that does not belong to Q_x. Process p_j can receive the message WITNESS(k, v_x)

(a) from each process of Q_x and, (b) from each of the $f \leq t_c$ Byzantine processes. It follows that p_j can receive WITNESS(k, v_x) from at most $f + |Q_x| \leq t_c + |Q_x|$ different processes. As $t_c + |Q_x| < t_c + n - 2t_t - t_c = n - 2t_t$ the predicate of line 3 cannot be satisfied at p_j, and consequently, any process $p_j \notin Q_x$ will never send the message WITNESS(k, v_x). Hence, no correct process will receive $n-t_t$ such messages and so they will also not deliver MSG(k, v_x). It follows that if there is a single set of correct processes Q_z and this set is such that $|Q_z| \geq n - 2t_t - t_c$, at most one message MSG$(k, \cdot)$ may be BRB_delivered by a correct process, and this message is then MSG(k, v_z).

- Let us consider now the case where there are at least two different sets of correct processes Q_x and Q_y, each of size at least $n - 2t_t - t_c$. Since there are at most $f \leq t_c$ Byzantine processes, we have that $|Q_x| + |Q_y| + f \leq |Q_x| + |Q_y| + t_c \leq n$, which implies that $2(n - 2t_t - t_c) + t_c \leq n$, from which we obtain $n \leq 4t_t + t_c \leq 4t_t + max(t_c, t_v)$, which contradicts the assumption about on $n > 4t_t + max(t_c, t_v)$. Consequently, it is impossible to have more than one set composed of at least $n-2t_t-t_c$ correct processes.

From these results, it follows that, if p_k sends several messages INIT(k, v_1), INIT(k, v_2), ..., INIT(k, v_m), at most one of them can give rise to a set Q_x such that $|Q_x| \geq n - 2t_t - t_c$, and, consequently, at most one message MSG(k, v_x) can be BRB_delivered by any correct process.

Termination 1 and 2: Let $f \leq t_t$ and let the correct sender p_i broadcast message MSG(i, v), then it follows that all the correct process p_j receives this message. We know by Lemma 8 that no message WITNESS(i, v') with $v' \neq v$ can be forwarded by a correct process, hence, when p_j receives INIT(i, v), it broadcasts the message WITNESS(i, v) at line 2. Every correct process (including the sender) will eventually receive these messages, BRB_deliver MSG(i, v), and terminate.

Termination 3: Let $f \leq t_t$ and let p_i be a correct process that BRB_deliver MSG(k, v). It follows that p_i must have received at least $n-t_t$ WITNESS(k, v) messages of which at least $n-t_t-f \geq n-2t_t > t_t$ from correct processes. It follows that at least $n-2t_t$ correct processes broadcast WITNESS(k, v), and consequently, the predicate of line 3 is eventually true at each correct process. Every correct process will then forward (if not already done) WITNESS(k, v) and BRB_deliver MSG(k, v) as soon as $n - t_t$ WITNESS(k, v) arrives and then terminate. ∎

VI. Multi-Threshold COOL Protocol

A. Introduction

In this section, we provide a technical overview of the different primitives used by our protocols. We assume a finite field $\mathbb{F}$ of size at least $n + 1$. The protocols also rely on Reed-Solomon decoding. Let $R = \{(i, v_i)\}$ denote a set of evaluations of a degree-d polynomial over $\mathbb{F}$, where up to f of the points may be incorrect. If $|R| > 2f + d$, then there exists an efficient algorithm which, given R, d, and f, can recover a degree-d polynomial $f(x)$ such that for at least $|R| - f$ points $(i, v_i) \in R$, it holds that $f(i) = v_i$. We denote this algorithm by $RSDec(R)$. We assume that the input is a polynomial of degree at most d over $\mathbb{F}$. For the general case of an L-bit input, the parties segment the L-bit message into $\ell = \lceil \frac{L}{(d+1)\log|\mathbb{F}|} \rceil$ blocks of $(d + 1)\log|\mathbb{F}|$ bits each. The parties then execute ℓ instances of the same protocol, where the input to the mth instance is the mth block interpreted as a polynomial $f(x)$ of degree at most d.

We set the degree parameter to $d = \lfloor t_t/3 \rfloor$, so in particular $3d < t_t$. We also assume $n > 2t_t + \max(t_t, t_c, t_v)$.

Useful notion. Given two distinct polynomials $f(x) \neq g(x)$ of degree at most d, they can share at most d evaluation points. If they have more than d points in common (i.e., at least $d+1$), then necessarily $f(x) = g(x)$.

B. Dispersal

Definition 3 (Dispersal)**.** *A protocol for parties $P_1, \ldots, P_n$ where the input of each party P_i is some $v_i \in \mathbb{F}$, is an asynchronous dispersal protocol if the following properties hold:*

- ***Weak Consistency.***
  1) *If $f \leq t_c$ and an honest party terminates with $f(x)$, then all honest parties that terminate with a non-$\perp$ output terminate with the same polynomial $f(x)$.*
- ***Weak Validity.*** *If $f \leq t_v$ and all honest parties start with the same polynomial $f(x)$ of degree at most d, then all honest parties that terminate output either $f(x)$ or $\perp$.*
- ***Termination.***
  1) *If $f \leq t_t$ and an honest party terminates, then all honest parties terminate and at least $max(t_t, t_c, t_v)+1$ output $f(x)$ while the other outputs $\perp$.*
  2) *If $f \leq t_t$ and all honest parties start with the same polynomial $f(x)$ of degree at most d, then all parties terminate and at least $max(t_t, t_c, t_v) + 1$ output $f(x)$ while the other outputs $\perp$.*

Lemma 9. *If $f \leq max(t_t, t_c, t_v)$ and $n > 2t_t + max(t_t, t_c, t_v)$, then there can be at most two distinct inputs such that there exist honest parties holding these inputs and sending OK_1 messages.*

Proof: Assume for contradiction that there exist honest parties P_a, P_b, P_c holding distinct input polynomials f_a, f_b, f_c such that P_a, P_b, P_c all send OK_1 messages. Let H be the set of all honest parties. For any $x \in \{a, b, c\}$, define S_x as the set of honest parties that agree with P_x, namely $S_x := A_x^1 \cap H$. Observe that:

- For any $x \in \{a, b, c\}$, $|S_x| \geq n - t_t - f$. Indeed, if P_x sent OK_1, it must agree with at least $n - t_t$ parties, and

Algorithm 4: Multi-Threshold Dispersal protocol under $n > 2t_t + max(t_t, t_c, t_v)$.

Input.

Each party P_i holds $f_i(x)$ of degree at most d over $\mathbb{F}$

The Protocol.

1) Initialization: Each party initializes $S_i = \emptyset$, $A_i^1 = A_i^2 = \emptyset$, and $f_i(x) = \perp$.
2) **Exchange:**
   a. P_i sends $(Exchange, f_i(i), f_i(j))$ to each P_j.
3) **Dynamic set** A_i^1:
   a. Upon receiving $(Exchange, u_i, v_j)$ from P_j, if $f_i(j) = u_j$ and $f_i(i) = v_i$ then add j to A_i^1.
   b. Upon $|A_i^1| \geq \boxed{n - t_t}$, send OK_1 to all.
4) **Dynamic set** A_i^2**:**
   a. Upon receiving message (OK_1) from P_j for which $j \in A_i^1$, then add j to A_i^2.
   b. Upon $|A_i^2| \geq \boxed{n - t_t}$, send OK_2 to all.
5) **Sending** $Done$**:**
   a. if P_i sent OK_2, and **Upon** receiving $\boxed{t_t + max(t_t, t_c, t_v) + 1}$ messages, send $Done$ message to everyone.
   b. Upon receiving $\boxed{max(t_t, t_c, t_v) + 1}$ $Done$ messages from distinct parties, send $Done$ to everyone.
   c. Upon receiving $\boxed{t_t + max(t_t, t_c, t_v) + 1}$ $Done$ messages, if P_i sent OK_2 message, then terminate and output $f_i(x)$. If P_i did not send OK_2 message, then terminate and output $\perp$.

therefore with at least $n - t_t - f \geq n - t_t - max(t_t, t_c, t_v)$ honest parties.

- For any $x, y \in \{a, b, c\}$, $|S_x \cap S_y| \leq d$. For any $j \in S_x \cap S_y$, we have $f_x(j) = f_j(j) = f_y(j)$. Thus, if $|S_x \cap S_y| \geq d + 1$, then f_x and f_y agree on at least $d + 1$ points and must therefore be identical. Hence, by inclusion–exclusion: $|S_a \cup S_b \cup S_c| = |S_a| + |S_b| + |S_c| - |S_a \cap S_b| - |S_a \cap S_c| - |S_b \cap S_c| + |S_a \cap S_b \cap S_c| \geq 3(n - t_t - f) - 3d + 0 \geq 3n - 4t_t - 3f$ On the other hand, $(S_a \cup S_b \cup S_c) \subseteq H$ and therefore $3n - 4t_t - 3f \leq n - f$. Since $f \leq max(t_t, t_c, t_v)$ this then implies: $2n \leq 4t_t - 2max(t_t, t_c, t_v)$ and so $4t_t + 2max(t_t, t_c, t_v) + 2 \leq 4t_t - 2max(t_t, t_c, t_v)$ which is a contradiction.

Therefore, let T_a and T_b be the two sets of honest parties with inputs f_a and f_b, respectively, such that only members of these sets send OK_1 messages. Since $|A_j^2| \subseteq |A_j^1|$ for each honest party P_j, only parties in T_a or T_b may send OK_2. Assume without loss of generality that $|T_a| \geq |T_b|$. Define:
$T_{a+} = \{i \in T_a | f_a(i) = f_b(i)\}$, $T_{a-} = T_a \setminus T_{a+}$
$T_{b+} = \{i \in T_b | f_a(i) = f_b(i)\}$, $T_{b-} = T_b \setminus T_{b+}$ ∎

Lemma 10. *If $f \leq \max(t_t, t_c, t_v)$, $n > 2t_t + max(t_t, t_c, t_v)$, and $|T_a| \geq t_t + 1$, then no party outside T_a sends OK_2.*

Proof: First, observe that parties in T_{b-} do not send OK_1. Since $|T_a| \geq t_t + 1$, and every $j \in T_{b-}$ satisfies $f_a(j) \neq f_b(j)$ by definition, any $j \in T_{b-}$ receives no support from T_a. Therefore, $|A_j^1| \leq n - |T_a| \leq n - t_t - 1 < n - t_t$, which prevents them from sending OK_1.

Next, we show that parties in T_{b+} do not send OK_2. The only honest parties whose OK_1 messages can be accepted by members of T_{b+} are those in $T_{a+} \cup T_{b+}$. Moreover, $|T_{a+} \cup T_{b+}| \leq d$, since otherwise $f_a = f_b$, contradicting their distinctness. Thus, for any $j \in T_{b+}$ we have $|A_j^2| \leq |T_{a+}| + |T_{b+}| + f \leq d + f \leq t_t + max(t_t, t_c, t_v) < n - t_t$, and therefore no party in T_{b+} sends OK_2. ∎

Lemma 11. *If $f \leq max(t_t, t_c, t_v)$, $n > 2t_t + max(t_t, t_c, t_v)$, and $|T_a| \leq t_t$, then no party output $f(x) \neq \perp$.*

Proof: First observe that parties in T_{a-} and T_{b-} do not send OK_2. All parties not in T_a or T_b do not send OK_1, and for $x \in \{a, b\}$, a party in T_{x-} can receive OK_1 only from T_x and from Byzantine parties. Thus, for any such party j, we have $|A_j^2| \leq |T_x| + f \leq t_t + max(t_t, t_c, t_v) < n - t_t$.

Given this, only parties in T_{a+} and T_{b+} could potentially send OK_2. However, no party can receive $t_t + max(t_t, t_c, t_v) + 1$ OK_2 messages. Since $|T_{a+} \cup T_{b+}| \leq d$, it follows that for any such party j, $|A_j^2| \leq |T_{a+}| + |T_{b+}| + f \leq d + f \leq t_t + max(t_t, t_c, t_v) < t_t + max(t_t, t_c, t_v) + 1$. Hence, no honest party satisfies the condition at line 5a, and so none of them outputs $f(x) \neq \perp$. ∎

Theorem 4. *If $n > 2t_t + max(t_t, t_c, t_v)$, then Protocol 4 is an asynchronous dispersal protocol tolerating $f \leq max(t_t, t_c, t_v)$ Byzantine parties.*

Proof: We show each one of the properties separately.

Termination. If $f \leq t_t$ and an honest party p terminates, then p received at least $t_t + max(t_t, t_c, t_v) + 1$ $Done$ messages, of which at least $t_t + max(t_t, t_c, t_v) + 1 - f \geq max(t_t, t_c, t_v) + 1 > f$ came from honest parties. This implies:

- At least one honest party sent a $Done$ message at line 5a, meaning it must have received at least $max(t_t, t_c, t_v) + 1$ OK_2 messages from honest parties.
- All honest parties will eventually execute step 5b and terminate with output $\perp$ (unless they previously executed step 5a, in which case they terminate with output $f(x)$ by Lemma 10).
- The $max(t_t, t_c, t_v) + 1$ honest parties that sent OK_2 messages will terminate with output $f(x)$.

Since an honest party executed line 5a, then by Lemma 11 there must be a set T_a of honest parties with the same polynomial $f(x)$ such that $|T_a| \geq t_t + 1$. From this, Lemma 10 implies that no honest party outside T_a sends OK_2, meaning that no honest party can output a polynomial $f'(x) \neq f(x)$.

If $f \leq t_t$ and all honest parties start with the same polynomial $f(x)$, then all $n - f \geq n - t_t$ honest parties will send OK_1, receive at least $n-t_t$ OK_1 messages, and therefore send OK_2. All honest parties will eventually either:

- receive $t_t + max(t_t, t_c, t_v) + 1$ OK_2 messages at step 5a, send $Done$, and terminate as soon as they receive $n-f \geq n-t_t > t_t+max(t_t, t_c, t_v)+1$ $Done$ messages, or
- receive $max(t_t, t_c, t_v) + 1$ $Done$ messages at step 5b, send $Done$, and terminate as soon as they receive $t_t + max(t_t, t_c, t_v) + 1$ $Done$ messages.

In either case, at least one honest party triggers step 5a, implying that at least $max(t_t, t_c, t_v) + 1$ honest parties sent OK_2, and these parties terminate with output $f(x)$.

Weak Consistency. If $f \leq t_c$ and an honest party terminates with output $f^*(x) \neq \perp$, then it must have sent an OK_2 message. Moreover, since it terminated, it must have received at least $t_t + max(t_t, t_c, t_v) + 1$ $Done$ messages, meaning it received $Done$ messages from at least $t_t + 1$ honest parties. An honest party sends $Done$ if it either received $t_t + max(t_t, t_c, t_v) + 1$ OK_2 messages, or if it received $max(t_t, t_c, t_v) + 1 > t_c \geq f$ $Done$ messages. The latter implies that at least one honest party sent $Done$ due to receiving $t_t+max(t_t, t_c, t_v)+1$ OK_2 messages, and therefore at least t_t+1 honest parties must have sent OK_2. By Lemma 9, there are at most two sets of parties that could have sent OK_1. Thus:

- By Lemma 11, if no set of size $t_t + 1$ shares the same polynomial, then no party receives $t_t+max(t_t, t_c, t_v)+1$ OK_2 messages. Hence, no honest party sends $Done$, and no party terminates.
- If there exists a set of size t_t+1 with the same polynomial $f^*(x)$, then no other party sends OK_2.

Therefore, if any honest party terminates, there must be a large set of size t_t+1 holding polynomial $f^*(x)$. This ensures that at least $max(t_t, t_c, t_v) + 1$ honest parties terminate with output $f^*(x)$, and any other honest party that terminates with a non-$\perp$ value must output the same polynomial $f^*(x)$.

Weak Validity. If $f \leq t_v$ and all honest parties start with a polynomial $f(x)$ of degree d, assume that a party P_i terminates with output $f_i(x)$. For P_i to output $f_i(x)$, it must have sent OK_2, meaning it received at least $|A_i^2| \geq n - t_t - f \geq n - t_t - t_c \geq d + 1$ OK_1 messages from honest parties. Since $|A_i^2| \subseteq |A_i^1|$, this implies that there are at least $d + 1$ points on which P_i agrees with the honest parties. Since each honest party contributes a point $u_j = f(j)$, there are at least $d + 1$ common points between $f_i(x)$ and $f(x)$, which is possible only if $f_i(x) = f(x)$.

■

Theorem 5. *Protocol 4 has a good-case latency of 4 asynchronous rounds and a bad-case latency of 5 asynchronous rounds.*

Proof:

Good-case latency. If all honest parties follow the fast path and execute step 5a, then every honest party performs the sequence $Exchange \to OK_1 \to OK_2 \to Done$. Hence, such executions are complete in 4 asynchronous rounds. If instead some honest parties execute step 5b, this does not increase the asynchronous-round complexity. The $Done$ messages that trigger step 5b may simply arrive earlier than the maximal honest-to-honest delay D used to define the round length. Thus, step 5b does not introduce an additional round, and the good-case latency remains exactly 4 asynchronous rounds.

Bad-case latency. When not all honest parties begin with the same polynomial, Byzantine processes can exploit this by sending valid OK_2 messages only to a subset of honest parties. These honest parties then obtain the required threshold of OK_2 messages and execute step 5a, sending $Done$. The remaining honest parties do not receive enough OK_2 messages and can send $Done$ only via step 5b. This selective behavior of Byzantine parties forces an additional causal wave of $Done$ messages. Thus the bad-case latency is exactly 5 asynchronous rounds. ■

C. Data Dissemination

Lemma 12. *If $f \leq max(t_c, t_v, t_t)$, $n > 2t_t + max(t_t, t_c, t_v)$ and fewer than $max(t_t, t_c, t_v) + 1$ parties start the Data Dissemination protocol with input $f(x)$, while all other honest parties start with $\perp$, then if two honest parties terminate with a non-$\perp$ output, it must be $f(x)$.*

Proof: Suppose that at most $max(t_t, t_c, t_v)$ honest parties begin the Data Dissemination protocol with input $f(x)$ while the others start with $\perp$.

- Since $f \leq max(t_t, t_c, t_v) < max(t_t, t_c, t_v) + 1$, the Byzantine parties cannot convince an honest party to send $(MyPoint, u_i)$ for any value $u_i \neq f(i)$.
- Since at most $max(t_t, t_c, t_v) < max(t_t, t_c, t_v)+1$ honest parties start with input $f(x)$, they cannot convince an honest party to send $(MyPoint, u_i = f(i))$ either.

From this, it follows that if an honest party P_i sends $(MyPoint, u_i)$, it must be because some Byzantine party behaved correctly toward P_i by sending $(YourPoint, u_i = f(i))$.

Now suppose two honest parties P_i and P_j terminate with outputs $f_i(x)$ and $f_j(x)$, respectively. This means they each received at least $max(t_t, t_c, t_v) + d + 1$ $(MyPoint, u)$ messages, of which at least $d+1$ came from honest parties. From the arguments above, and from the fact that all honest parties with a non-$\perp$ input begin with the same polynomial $f(x)$, it follows that these $d+1$ honest points must be generated using $f(x)$. Therefore, $f_i(x)$ and $f(x)$ share at least $d + 1$ points, and similarly $f_j(x)$ and $f(x)$ share at least $d+1$ points. This is only possible if $f_i(x) = f(x) = f_j(x)$. ■

Definition 4 (Data Dissemination)**.** *A protocol for parties $P_1, \ldots, P_n$ where each party P_i has input $f_i(x)$ (of degree d over $\mathbb{F}$, might be $\perp$), is an asynchronous data dissemination protocol if the following properties hold:*

Algorithm 5: Multi-Threshold Data Dissemination protocol under $n > 2t_t + max(t_t, t_c, t_v)$.

Input.

Each party P_i holds $f_i(x)$ as input, of degree at most d. Some P_i might have input $\perp$.

The Protocol.

1) Initialize a multi-set $M_i = S_i = \emptyset$. If $f_i(x) \neq \perp$, send to each P_j its point $(YourPoint, f_i(j))$.
2) **Upon** receiving $(YourPoint, u_j)$ from P_j, add u_j to M_i.
3) **Upon** some u_i appearing $max(t_t, t_c, t_v) + 1$ times in M_i, send $(MyPoint, u_i)$ to all parties.
4) **Upon** receiving $(MyPoint, u_j)$ from P_j, add (j, u_j) to S_i. **Upon** $|S_i| \boxed{\geq max(t_t, t_c, t_v) + d + 1}$ execute the following:
   a. Run $RSDec(S_i)$ and try to decode a polynomial $f_i(x)$ of degree at most d that agrees with s_i on at least $\boxed{max(t_t, t_c, t_v) + d + 1}$ values.
   b. If no such polynomial exists, then wait to receive more points in S_i and retry.
   c. If such a polynomial $f_i(x)$ is computed, set $f_i(x)$ to be the resultant value.
5) **Upon** unique decoding of $f_i(x)$, terminate and output $f_i(x)$.

- ***Consistency and Validity.*** *If $f \leq max(t_c, t_v)$ and there exist a polynomial $f(x)$ such that at least $max(t_t, t_c, t_v) + 1$ honest parties start with the input $f(x)$, and all other honest parties start with $\perp$, then all honest parties that terminate output $f(x)$.*
- ***Termination.*** *If $f \leq t_t$ and there exist a polynomial $f(x)$ such that at least $max(t_t, t_c, t_v) + 1$ honest parties start with the input $f(x)$, and all other honest parties start with $\perp$, then all honest parties terminate.*

Theorem 6. *If $n > 2t_t + max(t_t, t_c, t_v)$, then Protocol 5 is an asynchronous data dissemination protocol tolerating $f \leq max(t_t, t_c, t_v)$ Byzantine parties and has a latency of 2 asynchronous rounds.*

Proof: We show each one of the properties separately.

Termination. If $f \leq t_t$ and all honest parties start with either the same polynomial $f(x)$ or with $\perp$, and at least $\max(t_t, t_c, t_v) + 1$ honest parties start with $f(x)$, then:

- Since $f \leq max(t_t, t_c, t_v) < max(t_t, t_c, t_v) + 1$, the Byzantine parties cannot force an honest party to send $(MyPoint, u_i)$ with $u_i \neq f(i)$.
- Because at least $\max(t_t, t_c, t_v) + 1$ honest parties start with $f(x)$, every honest party will receive $(YourPoint, u_i = f(i))$ from at least $max(t_t, t_c, t_v)+1$ distinct parties, and therefore will send $(MyPoint, u_i = f(i))$.
- All $n - f \geq n - t_t \geq max(t_t, t_c, t_v) + d + 1$ honest parties will eventually succeed in performing the Reed–Solomon decoding and terminate with $f(x)$. Hence, a 2 asynchronous round latency.

Consistency and Validity. Suppose, for contradiction, that an honest party P_i terminates with output $f_i(x) \neq f(x)$. Then P_i must have received at least $|S_i| \geq d + max(t_t, t_c, t_v) + 1$ points, of which at least $d + 1$ are from honest parties. Since $f \leq max(t_c, t_v) \leq max(t_t, t_c, t_v) < max(t_t, t_c, t_v) + 1$, the Byzantine parties cannot cause an honest party to send $(MyPoint, u_i)$ with $u_i \neq f(i)$. Therefore, the $d + 1$ honest points in S_i must have been generated from $f(x)$. Thus $f_i(x)$ and $f(x)$ agree on at least $d + 1$ points, which is impossible unless $f_i(x) = f(x)$, yielding a contradiction.

■

D. Byzantine Reliable Broadcast

Algorithm 6: Multi-Threshold COOL protocol under $n > 2t_t + max(t_t, t_c, t_v)$.

Input.

The sender holds a polynomial $f(x)$ of degree at most d.

The Protocol.

1) **The sender:** send $f(x)$ to each party P_i.
2) **Each party P_i: Upon** receiving $f_i^{(1)}(x)$ from the sender, participate in Dispersal (Protocol 4) with input $f_i^{(1)}(x)$.
3) **Each party P_i: Upon** receiving an output $f_i^{(2)}(x)$ from the Dispersal protocol, participate in Data Dissemination protocol (Protocol 5) with input $f_i^{(2)}(x)$.
4) **Upon** receiving an output $f_i^{(3)}(x)$ from the Data Dissemination protocol, terminate and output $f_i^{(3)}(x)$.

Theorem 7. *If $n > 2t_t + max(t_t, t_c, t_v)$, then Protocol 6 is a BRB protocol tolerating $f \leq max(t_t, t_c, t_v)$ Byzantine parties, and it achieves good-case and bad-case latencies of 6 and 7 asynchronous rounds, respectively.*

Proof: We show each one of the properties separately.

Consistency. If $f \leq t_c$, then the Weak Consistency property of the Dispersal protocol guarantees that if two honest parties terminate with $f(x) \neq \perp$, they must output the same value. From this, all honest parties that terminate the Dispersal protocol will start the Data Dissemination protocol, and:

- If fewer than $max(t_t, t_c, t_v)$ honest parties terminate with output $f(x)$, then by Lemma 12, all honest parties that terminate the Data Dissemination protocol will output the same value.
- If at least $max(t_t, t_c, t_v)+1$ honest parties terminate with output $f(x)$, then these conditions satisfy the Consistency property of Data Dissemination, and all honest parties that terminate output $f(x)$.

In both cases, all terminating honest parties output the same value $f(x)$.

Validity. If $f \leq t_v$ and the sender P^* is honest with input $f(x)$, then eventually all honest parties start the Dispersal protocol with input $f(x)$. By the Weak Validity property of Dispersal, all honest parties that terminate do so with an output in $\{\perp, f(x)\}$. As in the Consistency argument:

- If sufficiently many honest parties output $f(x)$, we meet the Consistency and Validity conditions of the Data Dissemination protocol, and so all parties that terminate output $f(x)$.
- If not enough honest parties output $f(x)$, then by Lemma 12, all honest parties that terminate in the Data Dissemination protocol output $f(x)$.

In both cases, all honest parties that terminate output $f(x)$.

Termination. If $f \leq t_t$ and the sender P^* is honest with input $f(x)$, then eventually all honest parties start the Dispersal protocol with input $f(x)$. By the Termination property of Dispersal, all honest parties terminate with an output in $\{f(x), \perp\}$, and at least $max(t_t, t_c, t_v)+1$ honest parties output $f(x)$. All honest parties then start the Data Dissemination protocol, which under these conditions guarantees that all parties terminate. Hence, all honest parties eventually terminate the protocol. In this good-case scenario, the Dispersal protocol completes in 4 asynchronous rounds, yielding a total latency of 6 asynchronous rounds for the full protocol.

Finally, if $f \leq t_t$ and an honest party terminates, then it must have terminated both the Dispersal and Data Dissemination protocols. From the Termination property of Dispersal, if an honest party terminates, then at least $max(t_t, t_c, t_v) + 1$ honest parties terminate with some polynomial $f(x)$, while the remaining honest parties terminate with $\perp$. Thus all honest parties start Data Dissemination with such inputs and under these assumptions its, Termination property guarantees that all honest parties eventually terminate. Therefore, all honest parties will also eventually terminate the protocol. In this bad-case scenario, the Dispersal protocol completes in 5 asynchronous rounds, leading to a total latency of 7 asynchronous rounds for the full protocol. ∎

VII. Experimental Evaluation

Simulation Environment. All simulations were conducted using QUANTAS [19], a timestep-based simulator designed for the quantitative performance analysis of distributed algorithms. By operating in discrete time steps, QUANTAS ensures that results are independent of specific network conditions (e.g., connection delays), operating system architectures, and hardware characteristics such as processor speed, memory size, or number of cores. This abstraction enables fair and consistent comparisons across different algorithmic solutions.

Equivocation model. To introduce adversarial asymmetry, we vary the equivocation level of the Byzantine sender by distributing initial values among nodes according to predefined split ratios: $50/50, 60/40, 70/30, 80/20, 90/10$, and $100/0$. A split of 70/30, for example, indicates that 70% of nodes (correct and Byzantine) initially receive value 0 and the remaining 30% receive value 1.

Byzantine Behavior Modeling. To model Byzantine behavior in our experiments, we let a randomly selected Byzantine node act as the sender and distribute initial values to correct nodes according to the chosen equivocation split. Whenever a Byzantine process is required to transmit a message, it may adopt different behaviors depending on the message type. For messages that carry a value, i.e., messages of the form (*msg_type*, *value*), a Byzantine node may behave as *silent* (omit the message), *consistent* (send the value it should have sent), or *opposite* (send the opposite value). For message types that do not include an associated value (such as OK1, OK2, or DONE in the Dispersal algorithm), the possible behaviors reduce to *silent* or *send*.

Experiment Setup. All simulations are performed on a network of 100 nodes, with the number of Byzantine nodes selected according to each protocol's theoretical resilience condition, thus obtaining $t \in \{19, 20, 25, 33, 40\}$, where $t = 40$ represents the case in which all algorithms operate outside their guaranteed resilience range. In Figure 1, each algorithm is annotated in the legend with its corresponding theoretical threshold (e.g., since Bracha tolerates at most $t \leq n/3$, it is labeled as "Bracha $t \leq 33$").

Each network edge is assigned a random delay parameter λ drawn uniformly from the interval $[0.05, 0.2]$. This choice is motivated by the behavior of the geometric distribution governing message delays: as λ increases, the expected delay converges rapidly toward 1, yielding dynamics that resemble near-synchronous communication. By keeping λ within the lower range $[0.05, 0.2]$, we bias the system toward larger and more heterogeneous delays, better reflecting the asynchronous assumptions used in our theoretical analysis and producing a more challenging evaluation environment.

Evaluation Methodology. For every combination of algorithm, number of Byzantine nodes, equivocation ratio, and Byzantine behavior pattern, we perform 50 independent simulation runs. Although this number is relatively modest, preliminary experiments showed that 50 repetitions already yield very high statistical confidence: when plotting results with 95% confidence intervals, almost all plots exhibited confidence regions so narrow that they were visually indistinguishable from the central lines. Only a few plots displayed visible shading. This empirical observation confirms that 50 runs are sufficient to achieve high-confidence estimates for all reported metrics. All figures include confidence intervals, even when they collapse visually into a single line.

A. Research Questions and Results

In our experimental evaluation, we focus on understanding how asynchronous, error-free BRB protocols behave *outside* the theoretical multi-threshold resilience bounds established in the previous sections. Specifically, we investigate whether the fundamental BRB properties, *termination* and *agreement* (i.e., validity + consistency), degrade gradually or fail abruptly as

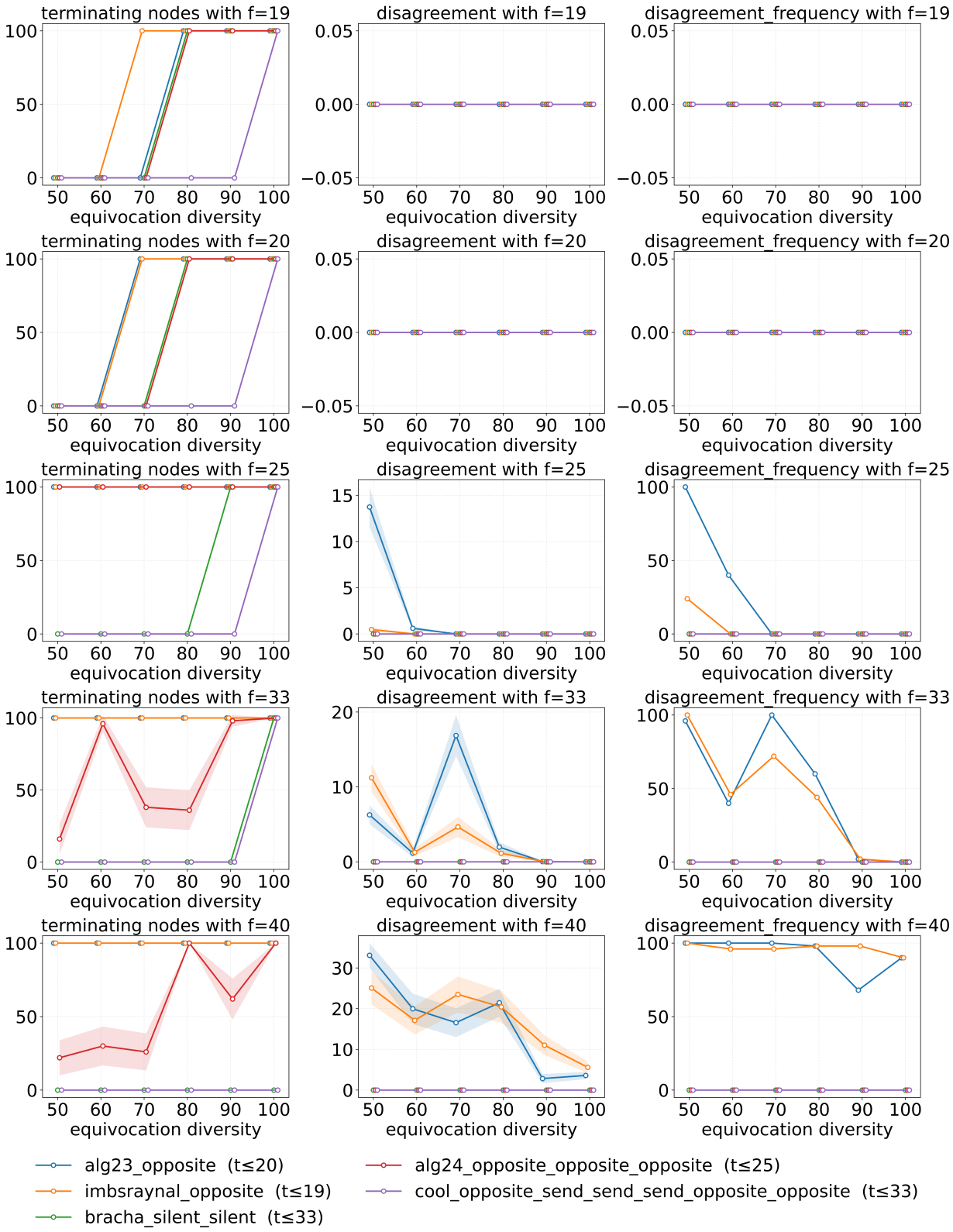

Fig. 1. Termination (left), disagreement (middle), and frequency of disagreement (right) for all protocols under varying Byzantine and equivocation levels. The figure highlights how protocols behave once pushed beyond their resilience bounds.

the number of Byzantine faults exceeds the proven resilience limits. Our experiments therefore, address two central research

questions:

- **RQ1: Termination Beyond Resilience.** How does the fraction of honest processes that terminate evolve as (i) the number of Byzantine nodes increases, and (ii) the equivocation level of the Byzantine sender varies? Do protocols degrade smoothly, or do they exhibit an all-or-nothing termination pattern?
- **RQ2: Agreement Beyond Resilience.** When the system operates outside its theoretical bounds, to what extent do honest nodes disagree on their outputs? Which protocols maintain agreement even under heavy equivocation or high Byzantine levels, and which ones exhibit disagreement?

To answer these questions, we measured: (i) the fraction of correct nodes that terminate; (ii) the degree of equivocation (e.g., percentage of correct nodes that delivered differently from the majority); (iii) the disagreement frequency (how often the system exhibits any disagreement);

RQ1: Termination As illustrated in the first row of Figure 1, all protocols display a strong dependence on the sender's equivocation level. Even for relatively small numbers of Byzantine nodes, equivocation levels close to 50% dramatically hinder termination across all protocols. This indicates that message equivocation has a greater negative impact on termination than the absolute number of Byzantine nodes.

With the exception of Alg(2,4), all algorithms exhibit a binary "0% or 100%" termination pattern: in each run, either all honest nodes terminate or none of them do. This phenomenon is consistent with Definition 1, which requires honest nodes to terminate only if the sender is correct or if another honest node terminates. Consequently, when the sender is Byzantine and termination fails for one honest node, termination typically fails for all.

RQ2: Agreement The second row of Figure 1 reports the level of disagreement among honest nodes. Only the Alg(2,3) and Imbs–Raynal protocols violate the agreement property, exhibiting high disagreement once the number of Byzantine nodes reaches 25 or more. This outcome is unsurprising: beyond the prescribed thresholds, the adversary obtains sufficient power to undermine all correctness guarantees, enabling widespread equivocation and driving honest nodes toward inconsistent decisions. The third row of Figure 1 confirms that this disagreement is systematic rather than sporadic: for these two protocols, disagreement occurs consistently across most runs.

In contrast, an unexpected outcome emerges for the remaining protocols: despite operating well beyond their proven resilience, they maintain a disagreement ratio of 0 across all experiments. That is, they *always* satisfy the agreement property under every evaluated condition, even when termination fails. This suggests that these protocols preserve consistency more robustly than theory guarantees, hinting at additional structural properties that shield them from adversarial equivocation.

VIII. Conclusion

This paper presented the first unified multi-threshold characterization of several foundational asynchronous, error-free Byzantine Reliable Broadcast protocols. By deriving explicit resilience conditions for validity, consistency, and termination, we showed how each protocol's guarantees degrade as the number of Byzantine faults increases, thereby generalizing and extending prior single-threshold analyses. Our results provide a consolidated understanding of the structural differences between classical BRB designs and offer a principled basis for comparing their fault-tolerance capabilities.

Complementing our theoretical analysis, we conducted an extensive empirical study using QUANTAS, evaluating all protocols beyond their proven resilience bounds. These experiments revealed that termination is highly sensitive to equivocation, often failing even at low fault levels, whereas agreement behaves in a more diverse and sometimes unexpectedly robust manner. Notably, some protocols preserve agreement in all tested settings despite operating far outside their guaranteed thresholds, while others exhibit systematic disagreement once adversarial power grows.

Together, our theoretical and experimental findings highlight both the fragility and resilience of BRB protocols under adversarial stress, and provide new insights into how protocol design influences behavior beyond worst-case bounds. We hope this multi-threshold perspective will inspire more fine-grained analyses and guide the development of future broadcast primitives with more predictable degradation under faults.

References

[1] Ittai Abraham, Gilad Asharov, and Anirudh Chandramouli. Simple is cool: graded dispersal and its applications for byzantine fault tolerance. *Cryptology ePrint Archive*, 2024.

[2] Ittai Abraham, Kartik Nayak, Ling Ren, and Zhuolun Xiang. Good-case latency of byzantine broadcast: A complete categorization. In *Proceedings of the 2021 ACM Symposium on Principles of Distributed Computing*, pages 331–341, 2021.

[3] Ittai Abraham, Ling Ren, and Zhuolun Xiang. Good-case and bad-case latency of unauthenticated byzantine broadcast: A complete categorization. *arXiv preprint arXiv:2109.12454*, 2021.

[4] Nicolas Alhaddad, Sourav Das, Sisi Duan, Ling Ren, Mayank Varia, Zhuolun Xiang, and Haibin Zhang. Balanced byzantine reliable broadcast with near-optimal communication and improved computation. In *Proceedings of the 2022 ACM Symposium on Principles of Distributed Computing*, pages 399–417, 2022.

[5] Silvia Bonomi, Jérémie Decouchant, Giovanni Farina, Vincent Rahli, and Sébastien Tixeuil. Practical byzantine reliable broadcast on partially connected networks. In *2021 IEEE 41st International Conference on Distributed Computing Systems (ICDCS)*, pages 506–516. IEEE, 2021.

[6] Silvia Bonomi, Jérémie Decouchant, Giovanni Farina, Vincent Rahli, and Sébastien Tixeuil. Practical byzantine reliable broadcast on partially connected networks. In *41st IEEE International Conference on Distributed Computing Systems, ICDCS 2021, Washington DC, USA, July 7-10, 2021*, pages 506–516. IEEE, 2021. doi:10.1109/ICDCS51616.2021.00055.

[7] Silvia Bonomi, Giovanni Farina, and Sébastien Tixeuil. Multi-hop byzantine reliable broadcast with honest dealer made practical. *Journal of the Brazilian Computer Society*, 25(1):9, 2019.

[8] Gabriel Bracha. Asynchronous byzantine agreement protocols. *Information and computation*, 75(2):130–143, 1987.

[9] Gabriel Bracha and Sam Toueg. Asynchronous consensus and broadcast protocols. *J. ACM*, 32(4):824–840, October 1985. doi:10.1145/4221.214134.

[10] Christian Cachin and Stefano Tessaro. Asynchronous verifiable information dispersal. In *24th IEEE Symposium on Reliable Distributed Systems (SRDS'05)*, pages 191–201. IEEE, 2005.

[11] Ran Canetti and Tal Rabin. Fast asynchronous byzantine agreement with optimal resilience. In *Proceedings of the twenty-fifth annual ACM symposium on Theory of computing*, pages 42–51, 1993.

[12] Jinyuan Chen. Optimal error-free multi-valued byzantine agreement. In *35th International Symposium on Distributed Computing (DISC 2021)*, pages 17–1. Schloss Dagstuhl–Leibniz-Zentrum für Informatik, 2021.

[13] Sourav Das, Zhuolun Xiang, and Ling Ren. Asynchronous data dissemination and its applications. In *Proceedings of the 2021 ACM SIGSAC Conference on Computer and Communications Security*, pages 2705–2721, 2021.

[14] Danny Dolev and Rüdiger Reischuk. Bounds on information exchange for byzantine agreement. *Journal of the ACM (JACM)*, 32(1):191–204, 1985.

[15] Martin Hirt, Ard Kastrati, and Chen-Da Liu-Zhang. Multi-threshold asynchronous reliable broadcast and consensus. *Cryptology ePrint Archive*, 2020.

[16] Damien Imbs and Michel Raynal. Trading off t-resilience for efficiency in asynchronous byzantine reliable broadcast. *Parallel Processing Letters*, 26(04):1650017, 2016.

[17] Leslie Lamport, Robert Shostak, and Marshall Pease. The byzantine generals problem. *ACM Trans. Program. Lang. Syst.*, 4(3):382–401, July 1982. doi:10.1145/357172.357176.

[18] Kartik Nayak, Ling Ren, Elaine Shi, Nitin H Vaidya, and Zhuolun Xiang. Improved extension protocols for byzantine broadcast and agreement. *arXiv preprint arXiv:2002.11321*, 2020.

[19] Joseph Oglio, Kendric Hood, Mikhail Nesterenko, and Sébastien Tixeuil. Quantas: quantitative user-friendly adaptable networked things abstract simulator. In *Proceedings of the 2022 Workshop on Advanced tools, programming languages, and PLatforms for Implementing and Evaluating algorithms for Distributed systems*, pages 40–46, 2022.

[20] Michel Raynal. On the versatility of bracha's byzantine reliable broadcast algorithm. *Parallel Processing Letters*, 31(03):2150006, 2021.