Abstract

We show how to upgrade a Reliable Broadcast (RB) primitive to Atomic Reliable Broadcast (ARB) by leveraging a synchronous DenyList (DL) object. In a purely asynchronous message-passing model with crashes, ARB is impossible without additional power. The DL supplies this power by enabling round closing and agreement on a set of "+winners" for each round. We present the algorithm, its safety arguments, and discuss liveness and complexity under the assumed synchrony of DL.

Keywords Atomic broadcast, total order broadcast, reliable broadcast, consensus, synchrony, shared object, linearizability.

1 Introduction

Atomic Reliable Broadcast (ARB)—a.k.a. total order broadcast—ensures that all processes deliver the same sequence of messages. In asynchronous message-passing systems with crashes, implementing ARB is impossible without additional assumptions, as it enables consensus. We assume a synchronous DenyList (DL) object and demonstrate how to combine DL with an asynchronous RB to realize ARB.

2 Model

We consider a static set of n processes with known identities, communicating by reliable point-to-point channels, in a complete graph. Messages are uniquely identifiable.

Synchrony. The network is asynchronous. Processes may crash; at most f crashes occur.

Communication. Processes can exchange through a Reliable Broadcast (RB) primitive (defined below) which's invoked with the functions $\mathsf{RB\text{-}cast}(m)$ and $\mathsf{RB\text{-}received}(m)$. There exists a shared object called DenyList (DL) (defined below) that is interfaced with the functions $\mathsf{APPEND}(x)$, $\mathsf{PROVE}(x)$ and $\mathsf{READ}()$.

Notation. Let Π be the finite set of process identifiers and let $n \triangleq |\Pi|$. Two authorization subsets are $\Pi_M \subseteq \Pi$ (processes allowed to issue APPEND) and $\Pi_V \subseteq \Pi$ (processes allowed to issue PROVE). Indices $i, j \in \Pi$ refer to processes, and p_i denotes the process with identifier i. Let $\mathcal{M}$ denote the universe of uniquely identifiable messages, with $m \in \mathcal{M}$. Let $\mathcal{R} \subseteq \mathbb{N}$ be the set of round identifiers; we write $r \in \mathcal{R}$ for a round. We use the precedence relation $\prec$ for the DL linearization: $x \prec y$ means that operation x appears strictly before y in the linearized history of DL. For any finite set $A \subseteq \mathcal{M}$, $\mathsf{ordered}(A)$ returns a deterministic total order over A (e.g., lexicographic order on $(\mathit{senderId}, \mathit{messageId})$ or on message hashes). For any round $r \in \mathcal{R}$, define $\mathsf{Winners}_r \triangleq \{\, j \in \Pi \mid (j, \mathsf{PROVE}(r)) \prec \mathsf{APPEND}(r) \,\}$, i.e., the set of processes whose $\mathsf{PROVE}(r)$ appears before the first $\mathsf{APPEND}(r)$ in the DL linearization.

3 Primitives

3.1 Reliable Broadcast (RB)

RB provides the following properties in the model.

- **Integrity**: Every message received was previously sent. $\forall p_i : \mathsf{RB\text{-}received}_i(m) \Rightarrow \exists p_j : \mathsf{RB\text{-}cast}_j(m)$.
- **No-duplicates**: No message is received more than once at any process.
- **Validity**: If a correct process broadcasts m, every correct process eventually receives m.

3.2 DenyList (DL)

The DL is a *shared, append-only* object that records attestations about opaque application-level tokens. It exposes the following operations:

- APPEND(x)
- PROVE(x): issue an attestation for token x; this operation is *valid* (return true) only if no APPEND(x) occurs earlier in the DL linearization. Otherwise, it is invalid (return false).
- READ(): return a (permutation of the) valid operations observed so far; subsequent reads are monotone (contain supersets of previously observed valid operations).

Validity. APPEND(x) is valid iff the issuer is authorized (in Π_M) and x belongs to the application-defined domain S. PROVE(x) is valid iff the issuer is authorized (in Π_V) and there is no earlier APPEND(x) in the DL linearization.

Progress. If a correct process invokes APPEND(x), then eventually all correct processes will be unable to issue a valid PROVE(x), and READ at all correct processes will (eventually) reflect that APPEND(x) has been recorded.

Termination. Every operation invoked by a correct process eventually returns.

Interface and Semantics. The DL provides a single global linearization of operations consistent with each process's program order. READ is prefix-monotone; concurrent updates become visible to all correct processes within bounded time (by synchrony). Duplicate requests may be idempotently coalesced by the implementation.

4 Target Abstraction: Atomic Reliable Broadcast (ARB)

Processes export AB-broadcast(m) and AB-deliver(m). ARB requires total order:

$$\forall m_1, m_2,\ \forall p_i, p_j : \quad \mathsf{AB\text{-}deliver}_i(m_1) < \mathsf{AB\text{-}deliver}_i(m_2) \Rightarrow \mathsf{AB\text{-}deliver}_j(m_1) < \mathsf{AB\text{-}deliver}_j(m_2),$$

plus Integrity/No-duplicates/Validity (inherited from RB and the construction).

5 Algorithm

Each process p_i maintains:

- received — set of RB-received messages not yet delivered;
- delivered — set of messages already delivered;

- $\mathsf{prop}[r][j]$ — proposal set announced by process p_j for round r (possibly $\perp$ locally);
- Local view of DL via $\mathsf{READ}()$.
- *next* — lowest round index not yet delivered.

5.1 Handlers and Procedures

Algorithm A RB handler (at process p_i)

on $\mathsf{RB\text{-}received}(m, S, r, j)$ **do**
$\mathsf{received} \leftarrow \mathsf{received} \cup \{m\}$
$\mathsf{prop}[r][j] \leftarrow S$ ▷ Record sender j's proposal S for round r

An $\mathsf{AB\text{-}broadcast}(m)$ chooses the next open round from the DL view, proposes all pending messages together with the new m, disseminates this proposal via RB, then executes $\mathsf{PROVE}(r)$ followed by $\mathsf{APPEND}(r)$ to freeze the winners of the round. The loop polls DL until (i) some winner's proposal includes m in a *closed* round and (ii) all winners' proposals for closed rounds are known locally, ensuring eventual inclusion and delivery.

Algorithm B $\mathsf{AB\text{-}broadcast}(m)$ (at process p_i)

on $\mathsf{AB\text{-}broadcast}(m, S, r, j)$ **do**
$P \leftarrow \mathsf{READ}()$ ▷ Fetch latest DL state to learn recent PROVE operations
$r \leftarrow \max\{r' : \exists j,\ (j, \mathsf{PROVE}(r')) \in P\}$ ▷ Pick next open round: one past the most recent proved round
$S \leftarrow (\mathsf{received} \setminus \mathsf{delivered}) \cup \{m\}$ ▷ Propose all pending messages plus the new m
$\mathsf{RB\text{-}cast}(m, S, r, \mathit{self})$ ▷ Asynchronously disseminate proposal via RB
$\mathsf{PROVE}(r)$ ▷ Attest participation in round r while it is still open
$\mathsf{APPEND}(r)$ ▷ Close round r; freezes $\mathsf{Winners}_r$ in the DL linearization
while $(\nexists j : \exists r, (j, \mathsf{PROVE}(r)) \in P \wedge\ m \in \mathsf{prop}[r][j])$ **do** ▷ Wait until m is included in some closed round and all needed proposals are known
    $P \leftarrow \mathsf{READ}()$ ▷ Refresh local view of DL
    $\mathsf{RB\text{-}cast}(m, S, r, \mathit{self})$
    $\mathsf{PROVE}(r)$
    $\mathsf{APPEND}(r)$
end while

Algorithm C AB-deliver() at process p_i

on AB-deliver() **do** ▷ Called when the process wants to receive the next message
$P \leftarrow \mathsf{READ}()$ ▷ Fetch latest DL state to learn recent PROVE operations
if $\nexists j : (j, \mathsf{PROVE}(next)) \in P$ **then** ▷ No process has proved round *next*
    return $\perp$ ▷ No closed round ready for delivery
end if
$\mathsf{APPEND}(next)$ ▷ Close round *next* if not already closed
if $\mathsf{PROVE}(next) == FALSE$ **then** ▷ Process closed rounds strictly in order
    $P \leftarrow \mathsf{READ}()$ ▷ Refresh local view of DL
    $\mathsf{Winners}_{next} \leftarrow \{j : (j, \mathsf{PROVE}(next)) \in P\}$ ▷ Frozen winners of round *next*
    if $\forall j \in \mathsf{Winners}_{next} :\ \mathsf{prop}[next][j] \neq \perp$ **then**
        $M_{next} \leftarrow \bigcup_{j \in \mathsf{Winners}_{next}} \mathsf{prop}[next][j]$ ▷ Round-*next* message set
        for all $m \in \mathsf{ordered}(M_{next})$ **do** ▷ Deterministic per-round order
            if $m \notin \mathsf{delivered}$ **then**
                $\mathsf{delivered} \leftarrow \mathsf{delivered} \cup \{m\}$
                return m ▷ Deliver exactly one message per call
            end if
        end for
        $next \leftarrow next + 1$ ▷ This round fully processed; advance
        continue ▷ Try the next closed round (if any)
    else
        return $\perp$ ▷ Some winners' proposals not yet known via RB
    end if
end if
return $\perp$ ▷ No closed round ready for delivery

6 Correctness

Definition 1 (Closed round)**.** Given a DL linearization H, a round $r \in \mathcal{R}$ is *closed* in H iff H contains an operation $\mathsf{APPEND}(r)$. Equivalently, there exists a time after which every $\mathsf{PROVE}(r)$ is invalid in H.

Lemma 1 (Stable round closure)**.** *If a round r is closed, then there exists a unique linearization point t_0 of APPEND(r) in the DL, and from that point on, no PROVE(r) can be valid. Once closed, a round never becomes open again.*

Lemma 2 (Across rounds)**.** *Rounds are delivered in strictly increasing r. Concatenating the per-round sequences yields a single global total order. Equivalently, $\forall r_1, r_2$ such that r_1, r_2 are closed and $r_1 < r_2$, all round r such that $r_1 < r < r_2$ are also closed.*

Lemma 3 (Well-defined winners)**.** *In any execution, whenever some process computes* $\mathsf{Winners}_r$*, its current DL-view contains APPEND(r); hence* $\mathsf{Winners}_r$ *is computed only for closed round.*

Lemma 4 (View-Invariant Winners)**.** *For any closed round r, there exists a unique set*

$$\mathsf{Winners}_r = \{j : (j, \mathit{PROVE}(r)) \prec \mathit{APPEND}^{\star}(r)\}$$

with APPEND$^{\star}(r)$ being the earliest APPEND(r) in the DL linearization.

Proof. We admit that some $\mathsf{APPEND}(r)$ occurs; if $\mathsf{APPEND}(r)$ occurs then it exists an operation $\mathsf{APPEND}^\star(r)$ that is the earliest in the linearization.

Due to the validity property of DL, a $\mathsf{PROVE}(r)$ is valid iff $\mathsf{PROVE}(r) \prec \mathsf{APPEND}^\star(r)$. Thus,

$$\mathsf{Winners}_r \triangleq \{j : (j, \mathsf{PROVE}(r)) \prec \mathsf{APPEND}^\star(r)\}$$

Let's consider two correct processes p_i and p_j and two READ responses P_i and P_j such that

$$\mathsf{APPEND}^\star(r) \prec (i, \mathsf{PROVE}(r)) \prec P_i \mathsf{APPEND}^\star(r) \prec (j, \mathsf{PROVE}(r)) \prec P_j$$

By the DL interface, READ returns a permutation of the valid operations observed so far, and hence every $\mathsf{PROVE}(r)$ that precedes $\mathsf{APPEND}^\star(r)$ is valid while any $\mathsf{PROVE}(r)$ that follows $\mathsf{APPEND}^\star(r)$ is invalid. We ensures that

$$\{j : (j, \mathsf{PROVE}(r)) \in P_i\} = \{j : (j, \mathsf{PROVE}(r)) \in P_j\} = \mathsf{Winners}_r.$$

□

Lemma 5 (No empty winners)**.** *For any round r closed,* $\mathsf{Winners}_r \neq \emptyset$.

Proof. Assume some correct process invokes $\mathsf{APPEND}(r)$. Hence an $\mathsf{APPEND}(r)$ occurs, and there exists an operation $\mathsf{APPEND}^\star(r)$ that is earliest in the DL linearization.

Let p_k be the issuer of $\mathsf{APPEND}^\star(r)$. By the algorithm ($\mathsf{AB\text{-}broadcast}$), any process that issues $\mathsf{APPEND}(r)$ (B6) must have previously invoked $\mathsf{PROVE}(r)$ (B5) in its program order. Because the DL is linearizable and preserves per-process program order in the linearization, we obtain

$$(k, \mathsf{PROVE}(r)) \prec \mathsf{APPEND}^\star(r).$$

Consequently, $(k, \mathsf{PROVE}(r))$ is a *valid* attestation for round r (no prior $\mathsf{APPEND}(r)$ exists before $\mathsf{APPEND}^\star(r)$ by definition of $\mathsf{APPEND}^\star(r)$). By the definition of $\mathsf{Winners}_r$,

$$k \in \mathsf{Winners}_r.$$

Therefore $\mathsf{Winners}_r \neq \emptyset$, i.e., $|\mathsf{Winners}_r| > 0$. □

Lemma 6 (Proposal convergence)**.** *For any closed round r, after all RB messages from* $\mathsf{Winners}_r$ *are received, every correct process computes the same* $M_r = \bigcup_{j \in \mathsf{Winners}_r} \mathsf{prop}[r][j]$.

Proof. Fix a round r such that some $\mathsf{APPEND}(r)$ occurs; hence r is *closed.* By Lemma 3, all correct processes that observe $\mathsf{READ}()$ after $\mathsf{APPEND}^\star(r)$ compute the same winner set $\mathsf{Winners}_r$.

For any $j \in \mathsf{Winners}_r$, in process p_j's program order we have, for round r:

$$\mathsf{RB\text{-}cast}(\_, S_j, r, j) \text{ (B4)} \quad \text{before} \quad \mathsf{PROVE}(r) \text{ (B5)} \quad \text{before} \quad \mathsf{APPEND}(r) \text{ (B6)}.$$

Since $\mathsf{Winners}_r = \{\, j : (j, \mathsf{PROVE}(r)) \prec \mathsf{APPEND}^\star(r) \,\}$, each winner j executed (B4) before $\mathsf{APPEND}^\star(r)$, thereby broadcasting a RB message for (r, j) with some (uniquely determined) payload S_j.

The round chosen at (B2) is the next open round. After (B6) closes r, any later invocation of $\mathsf{AB\text{-}broadcast}$ by p_j picks a strictly larger round. Thus p_j executes at most one $\mathsf{RB\text{-}cast}(\_, \cdot, r, j)$, so the set S_j attached to (r, j) is unique.

Now consider any two correct processes p_a and p_b that have (via RB) received all winners' messages for round r. By the RB *Integrity* and *No-duplicates* properties, every delivery of p_j's RB message for (r, j) carries the same S_j, and the handler sets

$$\mathsf{prop}[r][j] \leftarrow S_j \text{ (A3)}$$

at both p_a and p_b. Therefore,

$$M_r^{(a)} \triangleq \bigcup_{j \in \mathsf{Winners}_r} \mathsf{prop}^{(a)}[r][j] = \bigcup_{j \in \mathsf{Winners}_r} S_j = \bigcup_{j \in \mathsf{Winners}_r} \mathsf{prop}^{(b)}[r][j] \triangleq M_r^{(b)}.$$

Hence, after all RB messages from $\mathsf{Winners}_r$ have been received, every correct process computes the same M_r. □

Lemma 7 (Per-round determinism)**.** *Given Lemma 6, all correct processes iterate* $\mathsf{ordered}(M_r)$ *in the same order; hence the delivery order inside round* r *is identical at all correct processes.*

Corollary 8 (No duplicates)**.** *Each message is delivered at most once since membership in* delivered *is tested before delivery.*

Lemma 9 (Inclusion)**.** *If some process invokes* $\mathsf{AB\text{-}broadcast}(m)$*, then there exist a (closed) round* r *and a process* $j \in \mathsf{Winners}_r$ *such that* j *invokes*

$$\mathsf{RB\text{-}cast}(\_, S, r, j) \quad \textit{with} \quad m \in S.$$

Equivalently, every $\mathsf{AB\text{-}broadcast}(m)$ *is included in the proposal of some winner of some closed round.*

Theorem 10 (ARB)**.** *Under the assumed* DL *synchrony and* RB *reliability, the algorithm implements Atomic Reliable Broadcast.*

References