# 1 Introduction

## 1.1 Model

### 1.1.1 Model Properties

The system consists of $n$ asynchronous processes communicating via reliable point-to-point message passing.
Each process has a unique, unforgeable identifier and knows the identifiers of all other processes.
Up to $f < n$ processes may crash (fail-stop).
The network is reliable : if a correct process sends a message to another correct process, it is eventually delivered.
Messages are uniquely identifiable : two messages sent by distinct processes or at different rounds are distinguishable
2 messages sent by the same processus in two differents rounds are differents

**Property 1 (Message Uniqueness)** *If two messages are sent by different processes, or by the same process in different rounds, then the messages are distinct.*
*Formally :*

$$\forall p_1, p_2,\ \forall r_1, r_2,\ \forall m_1, m_2,\ \left( \begin{array}{l} \mathit{send}(p_1, r_1, m_1) \wedge \mathit{send}(p_2, r_2, m_2) \\ \wedge\ (p_1 \neq p_2 \vee r_1 \neq r_2) \end{array} \right) \Rightarrow m_1 \neq m_2$$

### 1.1.2 Reliable Broadcast Properties

**Property 2** *Integrity Every message received was previously sent.*
*Formally :*
$\forall p_i : \textit{bc-recv}_i(m) \Rightarrow \exists p_j : \textit{bc-send}_j(m)$

**Property 3** *No Duplicates No message is received more than once at any single processor.*
*Formally :*
$\forall m, \forall p_i : \textit{bc-recv}_i(m)$ *occurs at most once*

**Property 4** *Validity All messages broadcast by a correct process are eventually received by all non faulty processors.*
*Formally :*
$\forall m, \forall p_i : \mathit{correct}(p_i) \wedge \textit{bc-send}_i(m) => \forall p_j : \mathit{correct}(p_j) \Rightarrow \textit{bc-recv}_j(m)$

### 1.1.3 AtomicBroadcast Properties

**Property 5** *AB Totally ordered* $\forall m_1, m_2, \forall p_i, p_j : \textit{ab-recv}_{p_i}(m_1) < \textit{ab-recv}_{p_i}(m_2) \Rightarrow \textit{ab-recv}_{p_j}(m_1) < \textit{ab-recv}_{p_j}(m_2)$

### 1.1.4 DenyList Properties

Let $\Pi_M$ be the set of processes authorized to issue `APPEND` operations, and $\Pi_V$ the set of processes authorized to issue `PROVE` operations.
Let $S$ be the set of valid values that may be appended. Let `Seq` be the linearization of operations recorded in the DenyList.

**Property 6** *APPEND Validity An operation* ***`APPEND`***$(x)$ *is valid iff : the issuing process* $p \in \Pi_M$*, and the value* $x \in S$

**Property 7** *PROVE Validity An operation $\texttt{PROVE}(x)$ is valid iff : the issuing process $p \in \Pi_V$, and there exists no $\texttt{APPEND}(x)$ that appears earlier in $\texttt{Seq}$.*

**Property 8** *PROGRESS If an APPEND(x) is invoked by a correct process, then all correct processes will eventually be unable to PROVE(x).*

**Property 9** *READ Validity READ() return a list of tuples who is a random permutation of all valids PROVE() associated to the identity of the emiter process.*

## 1.2 Algorithms

We consider a set of processes communicating asynchronously over reliable point-to-point channels. Each process maintains the following local or shared variables :

— ***received*** : the set of messages that have been received via the reliable broadcast primitive but not yet ordered.

— ***delivered*** : the set of messages that have been ordered.

— ***prop***$[r][j]$ : the proposal set announced by process $j$ at round $r$. It contains a set of messages that process $j$ claims to have received but not yet delivered.

— ***winner***$^r$ : the set of processes that have issued a valid `PROVE` for round $r$, as observed through the registry.

— `RB-cast`$(\texttt{PROP}, S, r, j)$ : a reliable broadcast invocation that disseminates the proposal $S$ from process $j$ for round $r$.

— `RB-delivered`$(\texttt{PROP}, S, r, j)$ : the handler invoked upon reception of a `RB-cast`, which stores the received proposal $S$ into *prop*$[r][j]$.

— `READ()` : returns the current view of all valid operations stored in the DenyList registry.

— `ordered`$(S)$ : returns a deterministic total order over a set $S$ of messages.

## 1.3 proof

**Theorem 1 (Integrity)** *If a message $m$ is delivered by any process, then it was previously broadcast by some process via the `AB-broadcast` primitive.*

**Proof 1** *Let $j$ be a process such that $\texttt{AB-deliver}_j(m)$ occurs.*

**Algorithm 1** Atomic Broadcast with DenyList

$\textit{proves} \leftarrow \emptyset$
$\textit{received} \leftarrow \emptyset$
$\textit{delivered} \leftarrow \emptyset$
$r_1 \leftarrow 0$

**AB-Broadcast**$_j(m)$
  $\texttt{RB-Broadcast}_j(m)$

**RB-delivered**$_j(m)$
  $\textit{received} \leftarrow \textit{received} \cup \{m\}$
  **repeat until** $\textit{received} \setminus \textit{delivered} \neq \emptyset$
    $S \leftarrow \textit{received} \setminus \textit{delivered}$
    $\textit{proves} \leftarrow \texttt{READ}()$
    $r_2 \leftarrow \max\{r : j,\ (j, \texttt{PROVE}(r)) \in \textit{proves}\} + 1$
    $\texttt{RB-cast}(\texttt{PROP}, S, r_2, j)$
    $\texttt{PROVE}(r_2)$
    **for** $r \in [r_1 + 1, \ldots, r_2]$ **do**
      $\texttt{APPEND}(r)$
      $\textit{proves} \leftarrow \texttt{READ}()$
      $\textit{winner}^r \leftarrow \{j : (j, \texttt{PROVE}(r)) \in \textit{proves}\}$
      **wait** $\forall j \in \textit{winner}^r,\ \textit{prop}[r][j] \neq \bot$
      $T \leftarrow \bigcup_{j \in \textit{winner}^r} \textit{prop}[r][j] \setminus \textit{delivered}$
      **for each** $m \in \texttt{ordered}(T)$
        $\textit{delivered} \leftarrow \textit{delivered} \cup \{m\}$
        $\texttt{AB-deliver}_j(m)$
    $r_1 \leftarrow r_2$

**RB-delivered**$_j(\texttt{PROP}, S, r_1, j_1)$
  $\textit{prop}[r_1][j_1] \leftarrow S$

$$
\begin{aligned}
& \texttt{AB-deliver}_j(m) && \textit{(line 18)} \\
\Rightarrow\ & m \in \texttt{ordered}(T),\ \text{with } T = \bigcup_{j' \in winner^r} prop[r][j'] \setminus delivered && \textit{(lines 16-17)} \\
\Rightarrow\ & \exists j_0,\ r_0 : m \in prop[r_0][j_0] && \textit{(line 16)} \\
\Rightarrow\ & prop[r_0][j_0] = S,\ \text{with } \texttt{RB-delivered}_j(PROP, S, r_0, j_0) && \textit{(line 22)} \\
\Rightarrow\ & S \text{ was sent in } \texttt{RB-cast}(PROP, S, r_0, j_0) && \textit{(line 9)} \\
\Rightarrow\ & S = received_{j_0} \setminus delivered_{j_0} && \textit{(line 6)} \\
\Rightarrow\ & m' \in received_{j_0} \text{ where } m' \text{ broadcast by } j_0 && \textit{(line 4)} \\
\Rightarrow\ & \textbf{if } m = m' \\
& \quad \Rightarrow \texttt{RB-Broadcast}_{j_0}(m) \text{ occurred} && \textit{(line 3)} \\
& \quad \Rightarrow \texttt{AB-Broadcast}_{j_0}(m) \text{ occurred} && \textit{(line 1)} \quad \square \\
& \textbf{else :}\ m \in received_{j_0} \setminus delivered_{j_0} \\
& \quad \Rightarrow m \in received_{j_0} && \textit{(line 4)} \\
& \quad \Rightarrow \texttt{RB-delivered}_{j_0}(m) \text{ occurred} && \textit{(line 3)} \\
& \quad \Rightarrow \exists j_1 : \texttt{RB-Broadcast}_{j_1}(m) \text{ occurred} && \textit{(line 2)} \\
& \quad \Rightarrow \texttt{AB-Broadcast}_{j_1}(m) \text{ occurred} && \textit{(line 1)} \quad \square
\end{aligned}
$$

*Therefore, every delivered message $m$ must originate from some call to `AB-Broadcast`.*

**Theorem 2 (No Duplication)** *No message is delivered more than once by any process.*

**Proof 2** *Assume by contradiction that a process $j$ delivers the same message $m$ more than once, i.e.,*

$$\texttt{AB-deliver}_j(m) \text{ occurs at least twice.}$$

$$
\begin{aligned}
& \texttt{AB-deliver}_j(m) \text{ occurs} && \textit{(line 19)} \\
\Rightarrow\ & m \in \texttt{ordered}(T),\ \text{where } T = \bigcup_{j' \in winner^r} prop[r][j'] \setminus delivered && \textit{(lines 16-17)} \\
\Rightarrow\ & m \notin delivered \text{ at that time}
\end{aligned}
$$

*However :*

$$
\begin{aligned}
& \texttt{delivered} \leftarrow \texttt{delivered} \cup \{m\} && \textit{(line 18)} \\
\Rightarrow\ & m \in delivered \text{ permanently} \\
\Rightarrow\ & \text{In any future round, } m \notin T' \text{ since } T' = \bigcup_{j' \in winner^r} prop[r'][j'] \setminus delivered \\
\Rightarrow\ & m \text{ will not be delivered again} \\
\Rightarrow\ & \text{Contradiction.}
\end{aligned}
$$

*Therefore, no message can be delivered more than once by the same process.* □

**Theorem 3 (Validity)** *If a correct process invokes $\texttt{AB-Broadcast}_j(m)$, then all correct processes eventually deliver $m$.*

**Proof 3** *Let $j$ be a correct process such that* $\texttt{AB-Broadcast}_j(m)$ *occurs (line 5).*

$$
\begin{array}{llr}
& \texttt{AB-Broadcast}_j(m) & \textit{(line 1)} \\
\Rightarrow & \texttt{RB-Broadcast}_j(m) \textit{ occurs} & \textit{(line 2)} \\
\Rightarrow & \forall j_0 : \texttt{RB-delivered}_{j_0}(m) & \textit{(line 3)} \\
\Rightarrow & m \in \textit{received}_{j_0} & \textit{(line 4)} \\
\Rightarrow & \textbf{\textit{if}}\ m \in \texttt{delivered}_{j_0} & \\
& \quad \Rightarrow \textit{delivered}_{j_0} \leftarrow \textit{textitdelivered}_{j_0} \cup \{m\} & \textit{(line 18)} \\
& \quad \Rightarrow \texttt{AB-delivered}_{j_0}(m) & \textit{(line 19)} \quad \square \\
& \textbf{\textit{else}}\ m \notin \textit{delivered}_{j_0} : & \\
& \quad \Rightarrow m \in S_{j_0} \textit{ since } S_{j_0} = \textit{receieved}_{j_0} \setminus \textit{delivered}_{j_0} & \textit{(line 6)} \\
& \quad \Rightarrow \exists r : \texttt{RB-cast}_{j_0}(\textit{texttt}PROP, S_{j_0}, r, j_0) & \textit{(line 9)} \\
& \quad\quad \Rightarrow \forall j_1 : \texttt{RB-Deliver}_{j_1}(\texttt{PROP}, S_{j_0}, r, j_0) \textit{ occurs} & \textit{(line 21)} \\
& \quad\quad \Rightarrow \textit{prop}[r][j_0] = S_{j_0} & \textit{(line 22)} \\
& \quad \Rightarrow \exists j_2 \in j_0 : \texttt{PROVE}_{j_2}(r) \textit{ is valid} & \textit{(line 10)} \\
& \quad \Rightarrow j_2 \in \textit{textitwinner}^r & \textit{(line 14)} \\
& \quad \Rightarrow T_{j_0} \ni \textit{prop}[r][j_2] \setminus \textit{delivered}_{j_0} & \textit{(line 16)} \\
& \quad \Rightarrow T_{j_0} \ni S_{j_2} \setminus \textit{delivered}_{j_0} \ni m & \textit{(line 16)} \\
& \quad \Rightarrow \texttt{AB-deliver}_{j_0}(m) & \textit{(line 19)} \quad \square
\end{array}
$$

**Theorem 4 (Total Order)** *If two correct processes deliver two messages $m_1$ and $m_2$, then they deliver them in the same order.*

**Proof 4**

$$
\begin{array}{llr}
& \forall j_0 : \texttt{AB-Deliver}_{j_0}(m_0) \wedge \texttt{AB-Deliver}_{j_0}(m_1) & \textit{(line 19)} \\
\Rightarrow & \exists r_0, r_1 : m_0 \in \texttt{ordered}(T^{r_0}) \wedge m_1 \in \texttt{ordered}(T^{r_1}) & \textit{(line 17)} \\
\Rightarrow & T^{r_0} = \bigcup_{j' \in winner^{r_0}} \textit{prop}[r_0][j'] \setminus \textit{delivered} \wedge & \\
& T^{r_1} = \bigcup_{j' \in winner^{r_1}} \textit{prop}[r_1][j'] \setminus \textit{delivered} & \textit{(line 16)} \\
\Rightarrow & \textbf{\textit{if}}\ r_0 = r_1 & \\
& \quad \Rightarrow T^{r_0} = T^{r_1} & \\
& \quad \Rightarrow m_0, m_1 \in \texttt{ordered}(T^{r_0}) \textit{ since } \texttt{ordered} \textit{ is deterministic} & \\
& \quad \Rightarrow \textbf{\textit{if}}\ m_0 < m_1 : & \\
& \quad\quad \Rightarrow \texttt{AB-Deliver}_{j_0}(m_0) < \texttt{AB-Deliver}_{j_0}(m_1) & \square \\
& \textbf{\textit{else if}}\ r_0 < r_1 & \\
& \quad \Rightarrow \forall m \in T^{r_0}, \forall m' \in T^{r_1} : \texttt{AB-Deliver}(m) < \texttt{AB-Deliver}(m') & \square
\end{array}
$$

*Therefore, for all correct processes, messages are delivered in the same total order.*