Efficient Asynchronous Byzantine Lattice Agreement with Optimal Resilience

Jane Open Access
Dummy University Computing Laboratory, [optional: Address], Country
My second affiliation, Country

Joan R. Public[1]
Department of Informatics, Dummy College, [optional: Address], Country

Abstract

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Praesent convallis orci arcu, eu mollis dolor. Aliquam eleifend suscipit lacinia. Maecenas quam mi, porta ut lacinia sed, convallis ac dui. Lorem ipsum dolor sit amet, consectetur adipiscing elit. Suspendisse potenti.

2012 ACM Subject Classification Replace ccsdesc macro with valid one

Keywords and phrases Lattice agreement, Byzantine failures, Distributed algorithm, Message-passing systems

Digital Object Identifier 10.4230/LIPIcs.CVIT.2016.23

Funding *Jane Open Access*: (Optional) author-specific funding acknowledgements
Joan R. Public: [funding]

Acknowledgements I want to thank ...

1 Introduction

2 Related work

Table 1 summarizes the latest findings on lattice agreement in message passing systems particularly highlighting the global message complexity.

Time model	Failure model	Paper	Rounds/Message delays	Total messages
Sync	Byzan	Zheng and Garg [5]	$O(\log f)$, $f < \frac{n}{3}$	$O(n^2 \log f)$
	Crash	Attiya et al. [1]	$O(\log n)$, $f < n$	$O(n^2)$
	Crash	Zheng et al. [8]	$O(\log f)$, $f < n$	$O(n^2 \log f)$
Async	Byzan	Us	$O(\log f)$, $f < \frac{n}{3}$	$O(n^3 \log f)$
	Byzan	Di Luna et al. [2]	$O(f)$, $f < \frac{n}{3}$	$O(n^2)$
	Byzan	Zheng and Garg [6]	$O(\log f)$, $f < \frac{n}{5}$	$O(n^2 \log f)$
	Crash	Faleiro et al. [3]	$O(n)$, $f < \frac{n}{2}$	–
	Crash	Zheng et al. [8]	$O(f)$, $f < \frac{n}{2}$	$O(n^2 f)$
	Crash	Zheng et al. [7]	$O(\log f)$, $f < \frac{n}{2}$	$O(n^2 \log f)$

Table 1 Related Work

[1] Optional footnote, e.g. to mark corresponding author

© Jane Open Access and Joan R. Public;
licensed under Creative Commons License CC-BY 4.0

3 Model and Definitions

We assume a distributed asynchronous message passing system with n processes with unique ids in $[p_1, p_2, ..., p_n]$. The communication graph is a clique, i.e., each process can send messages to any other process in the system (including itself). We assume that the communication channel between any two processes is reliable (no loss, corruption or creation of messages). There is no upper bound on message delay. We assume that processes can have Byzantine failures but at most $\frac{n}{3}$ processes can be Byzantine in any execution of the algorithm. We say a process is correct or non-faulty if it is not a Byzantine process.

In the following section, we recall the definition of the BLA problem that we are using.

4 The Byzantine Lattice Agreement Problem

Let E be a lattice of values that can be proposed by a process. Each process $p_i, i \in [n]$ has input x_i from a join semi-lattice $(X, \leq, \sqcup)$ with X being the set of elements in the lattice E, $\leq$ being the partial order defined on X, and $\sqcup$ being the join operation. Each process p_i has to output some $y_i \in X$ such that the following properties are satisfied. Let C denote the set of correct processes in the system.

Comparability: For all $i \in C$ and $j \in C$, either $y_i \leq y_j$ or $y_j \leq y_i$.

Downward-Validity: For all $i \in C$, $x_i \leq y_i$.

Upward-Validity: $\sqcup\{y_i \mid i \in C\} \leq \sqcup(\{x_i \mid i \in C\} \cup B)$, where $B \subseteq E$ and $|B| \leq f$.

4.1 The main algorithm

In this section, we present our algorithm to solve the BLA. The main algorithm remains similar to that of Zheng et al.[6]. Initially, each process makes its value known to at least $n-f$ processes and then collects the values from at least $n-f$ distinct processes, including its own. With this set of size at least $n-f$, each process can execute the classifier for $\log f$ rounds, which will enable it to decide. The main challenges encountered are in defining a classifier that can meet these requirements in the presence of Byzantine faults, as cited in [6]. For this, we use the classifier algorithm as proposed by Attiya et al. and simulate an SWMR register ([4]) which ensures the three desired properties.

Algorithm 1 Algorithm for the BLA Problem with $O(\log f)$ Rounds

Input: x_i: input value, $\ell_i = n - \frac{f}{2}$: initial label
Output: y_i: output value

REG$[i]$.write$(x_i, 0, 0)$; `// Initial step`
$V_i^1 \leftarrow$ REG.collect(0) ; `// Initial step`
for $r := 1$ **to** $\log f$ **do**
  $(V_i^{r+1}, class) \leftarrow Classifier(V_i^r, \ell_i, r)$;
  if $class = master$ **then**
    $\ell_i \leftarrow \ell_i + \frac{f}{2^{r+1}}$;
  else
    $\ell_i \leftarrow \ell_i - \frac{f}{2^{r+1}}$;
$y_i \leftarrow \sqcup\{v \in V_i^{\log f+1}\}$;

4.1.1 The classifier procedure

We do the same as the classifier algorithm presented in , except for lines 3 and 6 where we perform a sorting operation that consists of extracting the values with the correct label (label of process that performs the classification).

Algorithm 2 Classifier (V, ℓ, r) for p_i:

Input: V: input value set, ℓ: threshold value, r: round number
Output: $(V', class)$: updated value set and class

```
1 REG[i].write(V, ℓ, r);
2 collect_i ← REG.collect(r) ;                                // First collect
3 extract_i ← ⋃{v_k | (v_k, ℓ) ∈ collect_i};
4 if |extract_i| > ℓ then
5 |   M_collect_i ← REG.collect(r) ;                         // Second collect
6 |   M_extract_i ← ⋃{v_k | (v_k, ℓ) ∈ M_collect_i};
7 |   return (M_extract_i, master);
8 else
9 |   return (V, slave);
```

4.1.2 SWMR for BLA

The classifier calls our SWMR register for BLA (BLASWMR) algorithm, which we present here. In the BLASWMR, we construct a register for each round and use reliable broadcasts to ensure message reliability. In addition to the initial properties of RB, we assume that in our case, it includes a sequencer that ensures at most one write message can be R_delivered. This BLASWR is inspired by the work of [4].

The R_broadcast specifications:

- RB-Validity. If a correct process r-delivers a pair $(v, -, r, csn)$ from a correct process p_x, then p_x invoked the operation $R_broadcast$WRITE_DONE$(v, -, r, csn)$.
- RB-Integrity. Given any process p_i and any sequence number r, a correct process r-delivers at most once a $(v, -, r, csn)$ from p_i.
- RB-Uniformity. If a correct process r-delivers a pair $(v, -, r, csn)$ from p_i (possibly faulty), then all the correct processes eventually r-deliver the same $(v, -, r, csn)$ from p_i.
- RB-Termination. If the process that invokes $R\_broadcast(v, -, r, csn)$ is correct, all the correct processes eventually r-deliver $(v, -, r, csn)$.

4.1.3 The valid condition

The predicate allows verifying if a process has the right to write a value V at a given round.

- F0 condition for $(r = 0)$. It check if the value proposed by p_j is an element of the lattice E.
- F1 condition for $(r = 1)$. It checks if the size of $|V|$ is at least $n - f$, then verifies if at least $n - 2f$ different processes claim that p_i completed its collect operation in round $r = 0$ and that V is the value that it computed according to their responses to the collect.
- F2 $(r > 1)$. The first part ensures that the process claiming to be a slave has correctly updated its label and tries to write the same value as in the previous round. Additionally, at least $n - 2f$ processes claimed that p_j read less or equal to l' values (values with label l') during the collect operation.

Algorithm 3 *BLA SWRM for p_i*

Var initialisation : Map $reg_i : reg_i[r][1..n] := [\perp, \dots, \perp]$;
$csn_i := 0$ the collect number;
$known\_csn_i[1..n] := [0, \dots, 0]$;
$collect\_responses_i[c][k][j]$ value claimed to have been sent with the collect number c by process p_k to p_j.

Operation *REG[i].write(V, ℓ, r)*:
    R_broadcast WRITE (V, ℓ, r, csn_i);
    Wait until WRITE_DONE(r) received from at least $n - f$ different processes;
    return ();
Operation *REG.collect(r)*:
    $csn_i := csn_i + 1$;
    Broadcast COLLECT(csn_i, r);
    Wait until($\exists reg$: COLLECT_VALUE($known\_csn, reg$) is R_delivered from at least $n - f$ different processes with $known\_csn[i] = csn_i$);
    return reg;
When *a message WRITE(V, l, r, csn) from p_j is R_delivered*:
    Wait until $valid(j, V, l, r, csn)$;   `// Unlock when the condition valid() becomes True`
    $reg_i[r][j] := (V, l)$;   `// add value and it label`
    send WRITE_DONE(r) to p_j;
    R_broadcast COLLECT_VALUE($known\_csn_i, reg_i[r]$);
When *a message COLLECT(csn, r) from p_j is received*:
    if $(r = 0)$ **then**
        Wait until$|\{k \mid reg_i[0][k] \neq \perp\}| \geq n - f$;   // wait until at least $n - f$ different process have written before responding to the collect of round 0
    if $(known\_csn_i[j] < csn)$ **then**
        $known\_csn_i[j] := csn$;
        R_broadcast COLLECT_VALUE($known\_csn_i, reg_i[r]$);
When *a message COLLECT_VALUE($known\_csn, reg$) from p_k is R_delivered*:
    for j *in* $[1, n]$ **do**
        $c := known\_csn[j]$;
        $collect\_responses[c][k][j]$.append($reg$);   // add all reg that p_k claims to have sent to p_j with collect number $c = known\_csn[j]$

- F3 ($r > 1$). This formula ensures that the process claiming to be a master has correctly updated its label and if at least $n - 2f$ processes claim that p_j read more than l' values (values with label l') during the collect operation.

Predicate valid(j, V, l, r, csn) **for** p_i **is:**

Let $(V', l') := reg_i[r-1][j]$; `// What` p_j `write in` p_i `memory in round` $r-1$

Let $Commitable(j, reg, V', csn) = \exists K \subseteq [1, \ldots, n], |K| = n - f, \forall k \in K,$ $collect\_responses[csn][k][j] = reg$ AND $(V', l') = reg[j]$; `// True if` $n-f$ `different processes claim to have send` reg `(such that` $V' \in reg$`) to` p_j `with the collect number` csn

Let $Admissible(j, reg, V', csn) := \bigcup\{v \mid (v, l') \in reg$ and $Commitable(j, reg, V', csn) = \text{True}\}$; `// the set of value send by at least` $n-f$ `different processes with the label` l'

Let $A := (reg_i[r-1][j] \neq \bot)$; `// Check if` p_j `has written in the previous round it's value,` $r \geq 1$

Let $F0 := (r = 0) \wedge (V \in E)$; `// ensure that the value proposed by a process in the initial round is in the base lattice` E

Let $F1 := (r = 1) \wedge (|V| \geq n - f) \wedge$ $(\exists reg$ such that $Admissible(j, reg, V', csn) = V)$; `// Round 1 condition`

Let $F2 := (r > 1) \wedge \left(l = l' - \frac{f}{2^r}\right) \wedge (V = V') \wedge$ $(\exists reg$ such that $|Admissible(j, reg, V', csn)| \leq l')$; `// Slave specifications`

Let $F3 := (r > 1) \wedge (l = l' + \frac{f}{2^r}) \wedge (|V| > l') \wedge$ $(\exists reg$ such that $Admissible(j, reg, V', csn) = V)$; `// Master specifications`

$A \bigwedge (F1 \bigvee F2 \bigvee F3) \bigvee F0$; `// the main formula`

4.2 Proof of the algorithm

First and foremost, we start with demonstrating the following property.

▶ **Property 4.1.** *Let $n > 5f$. Any two sets of processes Q_1 and Q_2 of size at least $n - 2f$ have at least one correct process in their intersection.*

Proof.
- $Q_1 \cup Q_2 \subseteq \{p_1, \ldots, p_n\}$. Hence, $|Q_1 \cup Q_2| \leq n$.
- $|Q_1 \cap Q_2| = |Q_1| + |Q_2| - |Q_1 \cup Q_2| \geq |Q_1| + |Q_2| - n$. Hence, $|Q_1 \cap Q_2| \geq n - 4f$, from which it follows that $Q_1 \cap Q_2$ contains at least one correct process if and only if $n - 4f > f$. Thus $n > 5f$.

◀

▶ **Definition 1** (group). *A group is a set of processes which have the same label. The label of a group is the label of the processes in this group. The label of a group is also the threshold value processes in this group use to do classification.*

▶ **Definition 2** (commit). *A write message that is reliable broadcast by a process is said to be* **committed** *if it satisfies the valid condition at one correct process at least.*

▶ **Definition 3** (admissible values for a group). *The admissible values for a group G with label ℓ is the set of values that can be* ***committed*** *with label ℓ.*

Variable	Definition
G	A group of processes at round r with label ℓ
$slave(G)$	The slave subgroup of G, i.e., the processes with label $s(\ell, r)$ at round $r+1$
$master(G)$	The master subgroup of G, i.e., the processes with label $m(\ell, r)$ at round $r+1$
V_i^r	The value set of process p_i at the beginning of round r
U_ℓ^r	The set of admissible values for group ℓ at round r

Table 2 Notations

Let $s(\ell, r) = \ell - \frac{f}{2^{r+1}}$ and $m(\ell, r) = \ell + \frac{f}{2^{r+1}}$. Table. 2 show the definition of some variables used in the proof.

▶ **Lemma 4.** *Let $n > 3f$. $\forall p_i \in C$, If p_i completes a collect at round r and return reg then $reg[i] = (V, l)$ where (V, l) is the input of p_i write in round r.*

Proof. Since p_i ended its write step before the collect, there exist at least $n - f$ different processes that send WRITE_DONE(r) to p_i thus at least $n-2f$ correct processes (let denote by Q_1 the set of this processes) have executed Line 12 such that $\forall p_k \in Q_1, reg_k[r][j] = (V, l)$ before sending WRITE_DONE(r) to p_i. Since $n > 3f$, at least one correct process of Q_1 will intersect the $n - f$ (let denote by Q_2 the set of this processes)that sends the same reg (Line 20 or 14) to p_i during the collect. Thus $\forall p_i \in C, reg[i] = (V, l)$.
Proof of $|Q_1 \cap Q_2| \geq 1$ if $n > 3f$.
We have that $|Q_1| \geq n - 2f, |Q_2| \geq n - f$ and $|Q_1 \cup Q_2| \leq n$.

$$\begin{aligned}
|Q_1 \cap Q_2| &= |Q_1| + |Q_2| - |Q_1 \cup Q_2| \\
&\geq Q_1| + |Q_2| - n \\
&\geq n - 3f \geq 1 \text{ if } n > 3f
\end{aligned}$$

Thus $|Q_1 \cap Q_2| \geq 1$ if $n > 3f$.

◀

▶ **Lemma 5.** *Let $n > 3f$. Let $p_i \in C$ be a process that executes the two collect operations (Line 2 and Line 5 of Algorithm 3) for the same round $r > 0$. If p_i is correct then $\cup\{v$ such that $\forall l, (v, l) \in$ collect_i$\} \subseteq \cup\{v$, such that $\forall l, (v, l) \in$M_collect_i$\}$, where collect_i is the result of the collect of the Line 2 and M_collect_i the result of line 5.*

Proof. (a.) We have assumed that the Reliable Broadcast includes a sequencer that ensures at most one write message can be R_delivered in each round. Thus $\forall p_i \in C, p_i$ performs line 12 (of algorithm 3) at most one time per process ($reg_i[r][j] := (V, l), \forall p_j \in [1, \ldots, n]$). Hence, $\forall p_i \in C$ if collect_i $= reg_i[r]$ in time t_1 and M_collect_i $= reg_i[r]$ in time t_2, $t_1 < t_2$ then $\cup\{v$ such that $\forall l, (v, l) \in$ collect_i$\} \subseteq \cup\{v$, such that $\forall l, (v, l) \in$ M_collect_i$\}$.
(b.) The operation REG.collect($-$) terminated implies that at least $n - f$ processes send the same reg to p_i (line 8 of algorithm 3). Let denote by Q_1 (respectively Q_2) the set of $n - f$ different processes that send collect_i (M_collect_i) to p_i. Since $n > 3f$, $|Q_1 \cap Q_2| \geq f + 1$ thus there exists at least one correct process that intersects Q_1 and Q_2. By (a.), we conclude $\cup\{v$ such that $\forall l, (v, l) \in$ collect_i$\} \subseteq \cup\{v$, such that $\forall l, (v, l) \in$ M_collect_i$\}$. This end the proof.

◀

▶ **Lemma 6.** *Let $p_j \in C$. If p_j executed REG.collect(r) with collect number csn then, eventually, $\forall p_i \in C, reg \in collect\_responses_i[csn][k][j], \forall p_k \in Q$; where reg is the return of p_j's collect and Q is the set of at least $n-f$ different processes that send (using the Reliable broadcast) the same reg to p_j during the collect.*

Proof. $p_j \in C$ ended its collect implies that at least $n-f$ different processes (denoted by Q_1) send him the same reg (Using the Reliable Broadcast). Hence, all $p_k \in Q_1$ had performe Line 14 or 20 (R_broadcast COLLECT_VALUE($known\_csn, reg$)). Thus eventually, all correct processes ($p_i \in C$) will receive this Reliable broadcast (from $p_k \in Q_1$) and perform Line 24 ($collect\_responses[csn][k][j].append(reg)$) where $csn = known\_csn[j]$ which concludes the proof. ◀

▶ **Lemma 7.** *Let $n > 3f$, $p_j \in C$. If p_j performs $R\_broadcast(V, l, r, csn_j)$ (Line 2 of algo 4) then the predicate $valid(j, V, \ell, r, csn)$ will eventually be true at p_i $\forall p_i \in C$ and $\forall r \geq 0$.*

Proof. To prove that `valid()` is true is equivalent to prove that $A \wedge (F1 \vee F2 \vee F3) \vee F0$ will be True for all $p_i \in C$.

First, we prove $A := (reg_i[r-1][j] \neq \perp)$ $\forall r \geq 1$.
Because $p_j \in C$ it ended its write of round $r-1$ before performs $R\_broadcast(V, l, r, csn_j)$. Due to the RB-Termination of the Reliable Broadcast, $\forall p_i \in C$, p_i will eventually receive V_j^{r-1} (send by p_j during the round $r-1$) and p_i will execute line 12 then $reg_i[r-1][j] \neq \perp$ becomes True.

Subsequently, we prove the conditions $Fi, 0 \leq i \leq 3$. The condition $F0$ is verified during the initial round ($r = 0$), and the condition $F1$ is verified only in round $r = 1$, more precisely during the first classification round. $F2$ and $F3$ are used if $r > 1$.
Case $r = 0$: Since p_j is correct and use the Reliable Broadcast (Line 2) to send its value V in round 0, all correct processes will receive the same value V (due to RB-Uniformity) and $V \in E$ (Because $V = x_i \in E$ for all $p_i \in C$). Thus $F0 := (r = 0) \wedge (V \in E)$ will become True for all $p_i \in C$.
Case $r = 1$: We need to prove $F1$ will eventually be True.
$F1 := (r = 1) \wedge (|V| \geq n-f) \wedge (\exists reg$ such that $Admissible(j, reg, V', csn) = V)$

- Let prove that $|V| \geq n-f$ is True
  Because $p_j \in C$ it ended its collect of round 0. p_j ended its collect of round 0 thus there exist at least $n-2f$ different correct processes (Q) that send the same reg to p_j in round 0 after passing the line 17 i.e $\forall p_i \in Q, |\{k | reg_i[0][k] \neq \perp\}| \geq n-f$. Thus $|V| \geq n-f$ where, $V = \cup\{v, (v, 0) \in reg\}$.
- Let prove ($\exists reg$ such that $Admissible(j, reg, V', csn) = V$) will eventually be True at every correct processes p_i.
  Since p_j is correct and ended its collect of round 0, it's clear that p_j had received via the reliable broadcast the same reg from at least $n-f$ different processes. Thus every correct process will eventually (due to the reliable broadcast) receive the same reg and performs the line 24 (append reg to its collect_responses[csn_j][$-$][j]).(Each correct process can compute the Admissible condition easly) This conclude the proof.

Case $r > 1$: We need to prove that $F2 \vee F3$ becomes True for all $p_i \in C$ if p_j is a correct process.

- We give the proof of $F2$ first
  $F2 := (r > 1) \wedge \left(l = l' - \frac{f}{2^r}\right) \wedge (V = V') \wedge$
  ($\exists reg$ such that $|Admissible(j, reg, V', csn)| \leq l'$).

Let $p_j \in C$ executing $R\_broadcast(V, l, r, csn_j), (r > 1)$. It's clear that p_j ended the rounds $r' < r$ in particular round $r-1$ collect operations.
If p_j has executed the ligne 9 of the classifier, then it has execute the ligne 8 (of algo 2) thus $(l = l' - \frac{f}{2^r}) = True$ and $V = V'$.
Due to the termination of this collect (collect of round $r-1$), there exist at least $n-f$ processes that send the same reg to p_j such that $|\bigcup\{v_k \mid (v_k, \ell') \in reg\}| \leq l'$ (Line 4 of the classifier Algo 2). Since at least $n-f$ processes execute Lines 14 or 20 (of Algo 3), every correct process will eventually (due to the reliable broadcast) receive the same reg and performs the line 24 (append reg to its collect_responses$[csn_j][-][j]$). Thus $|Admissible(j, reg, V', csn)| \leq l'$ for all $p_i \in C$. This ends the proof for $F2$.

- Proof for $F3$
  $F3 := (r > 1) \wedge (l = l' + \frac{f}{2^r}) \wedge (|V| > l') \wedge (\exists reg \text{ such that } X(j, reg, V', csn) = V)$.
  Let $p_j \in C$ executing $R\_broadcast(V, l, r, csn_j), (r > 1)$. It's clear that p_j ended the rounds $r' < r$ in particular round $r-1$ collect operations.
  If p_j has execute the ligne 7 of the classifier, then it has execute the ligne 6 (of algo 2) thus $(l = l' + \frac{f}{2^r}) = True$. In addition to that, Lines 4 and 6 (of the classifier, algo 2) and lemma 5 implies that $|V| > l$.
  Due to the termination of the last collect of the round $r-1$, there exist at least $n-f$ processes that send the same reg to p_j such that $|\bigcup\{v_k \mid (v_k, \ell') \in reg\}| > l'$ (Line 4 of the classifier Algo 2). Since $p_i \in C$, by lemma 4 and lemma 6, $Commitable(j, reg, V', csn) = True$ at every correct process p_i. More than that, $V = \cup\{v, (v, l') \in reg\}$ thus "$\exists reg$ such that $Admissible(j, reg, V', csn) = V$" will becomes True where $(V', l') = reg_i[r][j]$.

◀

▶ **Lemma 8.** (Write termination) *Let $n > 3f$ (Due to the Reliable Broadcast). If p_i is correct and invokes $REG[i].write()$, its invocation terminates.*

Proof. Let $p_i \in C$ performs $REG[i].write(-,-,r)$. Due to the RB-termination property of the underlying reliable broadcast abstraction invoked by p_i at line 2, each correct process p_j R-delivers the message write$(-,-,r,-)$. By lemma 7, the predicate $valid(-,-,-,r,-)$ will eventually be True for p_j and p_j sends the message WRITE_DONE(r) to p_i (line 13). As there are at least $n-f$ correct processes, it follows that p_i cannot remain blocked forever at line 3, and the write invocation terminates. ◀

▶ **Lemma 9.** (Collect termination) *Let $n > 3f$. If p_j is correct and invokes $REG.collect()$, its invocation terminates.*

Proof. The proof is by contradiction. Let us assume that a correct process p_j invokes $REG.collect(-)$ and this invocation never terminates. This means that the predicate associated with the wait statement of line 8 remains false forever, namely, $\nexists reg$ such that the message COLLECT_VALUE$(known_csn, reg)$ is received from at least $n-f$ different processes with the correct csn.

As p_j is correct, it broadcasts the request message COLLECT(sn, r) where $sn = csn_j$ (line 7), and this message is received by all correct processes. Moreover, sn is the greatest sequence number ever used by p_j to collect, and, due to the contradiction assumption, csn_j keeps forever the value sn.

When a correct process p_k receives the message COLLECT(sn, r) from p_j, the predicate $known\_csn_k[j] < sn$ is satisfied (line 18). This is because sn is greater than all previous sequence numbers used by p_j to collect before. It follows that p_k updates $known\_csn_k[j]$ to

$sn = csn_j$, and broadcast COLLECT_VALUE($known\_csn_k, reg_k[r_j]$) (in particularly send to p_j)(lines19-20). Moreover, as the collect by p_j never terminates, $known\_csn_k[j]$ remains forever equal to $sn = csn_j$.

As the predicate of line 8 remains forever false at p_j, and p_j receives at least $(n - f)$ messages COLLECT($known\_csn, reg$) with $known\_csn[j] = csn$ (one from each correct process), it follows that p_j receives at least two messages COLLECT_VALUE($known\_csn, reg$) and COLLECT_VALUE($known\_csn', reg'$) such that $known\_csn[j] = known\_csn'[j]$ and $reg \neq reg'$.

Due to the RB-uniformity property of the underlying broadcast abstraction, it follows that all the correct processes r-delivers the same write() messages from correct or byzantine processes. Let p_k be a correct process. It follows directly from the code of the algorithm that, each time p_k adds a value to $reg_k[r][i]$ (line 12), it broadcasts a message COLLECT_VALUE($-, reg_k[r]$) (line 14). It follows (by the write termination) that there is a finite time after which p_j has received the very same reg contained in message COLLECT_VALUE($known\_csn, reg$) from at least $n - f$ different processes (with the correct $known\_csn[j]$). The predicate of line 8 becomes then satisfied. This contradicts the initial assumption, and the lemma follows. ◀

For lemma 10 to lemma 15, let G be a group at round $r \geq 1$ with label ℓ. Let L and R be two nonnegative integers such that $L < \ell \leq R$. If $L < |V_i^r| \leq R$ for each correct process $p_i \in G$, and $|U_\ell^r| \leq R$

▶ **Lemma 10.** *For each correct process $p_i \in master(G)$ and $p_j \in slave(G)$, $\ell < |V_i^{r+1}| \leq R$ and $L < |V_j^{r+1}| \leq \ell$.*

Proof. Immediate from the classifier procedure. ◀

▶ **Property 4.2.** *Suppose that process p_i (possibly Byzantine)* **commit** *(definition 2) a write message $(V_i, s(l,r), r+1, -)$. Then at least $n - f$ different processes known V_i before $p_i's$ collect at round r.*

Proof. Otherwise $\nexists n-f$ different processes s.t the condition Commitable() of valid(i, V_i, l, r, csn) becomes True at round $r + 1$. ◀

▶ **Lemma 11.** $|U_{s(\ell,r)}^{r+1}| \leq \ell$

Proof. Consider group $s(l,r)$ at round $r+1$. We know that this group must be the slave group of group l at round r. Let P denote the set of processes that can commit a write message at round $r+1$ with label $s(l,r)$. For each process $p_i \in P$, let $(V_i, s(l,r), r+1, -)$ denote the message that is committed by process p_i. Then, $U_{r+1}^{s(l,r)} = \bigcup\{V_i \mid p_i \in P\}$. Let denote by $p_j \in P$, the last process that received at least $n - f$ write done (line 3 of Algo 3).

Let call by extract_j, the result of p_j collect (line 3 of algo 2). It's clear that $U_{r+1}^{s(l,r)} \subseteq$ extract_j. Otherwise this implies that $\exists p_k \in P$ s.t $V_k \notin reg\_l$ (where reg_l is the result of line 8 of algo 3, sent by at least $n - f$ different processes.) By property 4.2 we not that at least $|K_1|$ different processes known V_k. Let denote by K_2 the set of process that known V_j; $|K_1| \geq n - f$ and $|K_2| \geq n - f$. Hence there exist at least one correct process that intersect K_1 and K_2 since $n > 3f$.

Due to the fact that there exist a correct process $p_h \in K_2$, it's clear that |extract_j| $\leq l$ thus $|U_{r+1}^{s(l,r)}| \leq l$. ◀

▶ **Lemma 12.** $|U_{m(\ell,r)}^{r+1}| \leq R$

Proof. The value set that can be commit by each correct process for group $m(\ell, r)$ is the union of values committed (reliably broadcast and valid) by processes in group ℓ at round r. Thus, $U^{r+1}_{m(\ell,r)} \subseteq U^r_\ell \implies |U^{r+1}_{m(\ell,r)}| \leq |U^r_\ell| \leq R$. ◀

▶ **Lemma 13.** $| \cup \{V_i^{r+1} \mid p_i \in slave(G) \cap C\}| \leq \ell$

Proof. Implied by lemma 11 ◀

▶ **Lemma 14.** $| \cup \{V_i^{r+1} \mid p_i \in master(G) \cap C\}| \leq R$

Proof. Implied by lemma 12 ◀

▶ **Lemma 15.** *For each correct process $p_j \in master(G)$, $U^{r+1}_{s(\ell,r)} \subseteq V_j^{r+1}$*

Proof. Let $p_i \in \text{slave}(G)$ and $p_j \in \text{master}(G)$. We know that there exist a set $Q_i(|Q_i| \geq n-f)$ of processes that participated in the write of process p_i at round r. Let Q_j be the set of the first $n - f$ processes that participated in the second collect of p_j.

Since p_i is a slave at round r, it must completes its write before the second collect of p_j (otherwise $p_i \in \text{master}(G)$). This implies that $V_i^{r+1} = V_i^r$ is known by all $p_k \in Q_j$ before the second collect of p_j is completed. Therefore, there exists at least one correct process in $Q_i \cap Q_j$ since $n > 3f$. Consequently, V_i^r will be included in the reg (COLLECT_VALUE(T, reg)) returned by process in Q_j (otherwise the collect does not terminate - impossible by lemma 9). Hence, $V_j^{r+1} = V_j^r \subseteq V_i^{r+1}$.We thus conclude that $U^{r+1}_{s(\ell,r)} \subseteq V_j^{r+1}$. ◀

▶ **Lemma 16.** *For any correct process p_i and round r, $V_i^r \subseteq V_i^{r+1}$.*

Proof. A slave process keeps its value set unchanged and a master process updates its value set to be the set values which contains its own value set. ◀

▶ **Lemma 17.** *Let G be a group of processes at round $r \geq 1$ with label ℓ. Then*
(1) for each correct process $i \in G$, $\ell - \frac{f}{2^r} \leq |V_i^r| \leq \ell + \frac{f}{2^r}$
(2) $|U_\ell^r| \leq \ell + \frac{f}{2^r}$

Proof. By induction on round number r and apply lemma 12, 11 and 10 ◀

▶ **Lemma 18.** *Let p_i and p_j be two correct processes that are in the same group G with label ℓ at the beginning of round* $\log f + 1$. *Then $V_i^{\log f+1}$ and $V_j^{\log f+1}$ are equal.*

Proof. Let G' be the parent of G with label ℓ'. Assume without loss of generality that $G = M(G')$. The proof for the case $G = S(G')$ follows in the same manner. Since G' is a group at round $\log f$, by Lemma 17, we have:
(1) for each correct process $p \in G'$, $\ell' - 1 < |V_p^{\log f}| \leq \ell' + 1$, and
(2) $|U_{\ell'}^{\log f}| \leq \ell' + 1$

Since $p_i \in G'$ and $p_j \in G'$, (1) and (2) hold for both process p_i and p_j. By the assumption that $G = M(G')$, process p_i and p_j execute the *Classifier* procedure with label ℓ' and are both classified as *master*. Let $L = \ell' - 1$ and $R = \ell' + 1$, then by applying Lemma 10 we have $\ell' < |V_i^{\log f+1}| \leq \ell' + 1$ and $\ell' < |V_j^{\log f+1}| \leq \ell' + 1$, thus $|V_i^{\log f+1}| = |V_j^{\log f+1}| = \ell' + 1$. By Lemma 14, we have $| \cup \{V_i^{\log f+1}, V_j^{\log f+1}\}| \leq \ell' + 1$. Thus, $V_i^{\log f+1} = V_j^{\log f+1}$. Therefore, V_i^r and V_j^r are equal at the beginning of round $\log f + 1$. ◀

▶ **Lemma 19.** *(Comparability) For any two correct process p_i and p_j, y_i and y_j are comparable.*

Proof. If process p_i and j are in the same group at the beginning of round $\log f + 1$, then by Lemma 18, $y_i = y_j$. Otherwise, let G be the last group that both p_i and p_j belong to. Suppose G is a group with label ℓ at round r. Suppose $i \in slave(G)$ and $j \in master(G)$ without loss of generality. Then, $V_i^{\log f+1} \subseteq U_{s(\ell,r)}^{r+1} \subseteq V_j^{r+1} \subseteq V_j^{\log f+1}$, by Lemma 15 ◀

4.3 Message Complexity

Messages are exchange only in the algorithm 3. A write operation costs $O(n^2)$ overall due to `R_broadcast`. There are at most n writes per round, resulting in $O(n^3)$ messages per round. Another costly line is line 14, which costs $O(n^3)$ messages per round per process, totaling $O(n^4)$ globally per round. A collect operation costs at most $O(n^2)$ per process and is called at most twice per round. In summary, the total number of messages is $O(n^4+n^3+n^2) = O(n^3)$ messages per round. Hence, our algorithm exchanges at $O(n^4 \log f)$ messages i.e $O(n^3 \log f)$ messages per process.

5 Conclusion

.....

6 Draft

References

1. Hagit Attiya, Maurice Herlihy, and Ophir Rachman. Atomic snapshots using lattice agreement. *Distrib Comput*, 8(3):121–132, March 1995. `doi:10.1007/BF02242714`.
2. Giuseppe Antonio Di Luna, Emmanuelle Anceaume, and Leonardo Querzoni. Byzantine generalized lattice agreement. In *2020 IEEE International Parallel and Distributed Processing Symposium (IPDPS)*, pages 674–683. IEEE, 2020.
3. Jose M. Faleiro, Sriram Rajamani, Kaushik Rajan, G. Ramalingam, and Kapil Vaswani. Generalized lattice agreement. In *Proceedings of the 2012 ACM symposium on Principles of distributed computing*, pages 125–134, Madeira Portugal, July 2012. ACM. URL: `https://dl.acm.org/doi/10.1145/2332432.2332458`, `doi:10.1145/2332432.2332458`.
4. Damien Imbs, Sergio Rajsbaum, Michel Raynal, and Julien Stainer. Read/write shared memory abstraction on top of asynchronous Byzantine message-passing systems. *Journal of Parallel and Distributed Computing*, 93-94:1–9, July 2016. URL: `https://www.sciencedirect.com/science/article/pii/S074373151630003X`, `doi:10.1016/j.jpdc.2016.03.012`.
5. Xiong Zheng and Vijay Garg. Byzantine lattice agreement in synchronous message passing systems. In *34th International Symposium on Distributed Computing (DISC 2020)*. Schloss-Dagstuhl-Leibniz Zentrum für Informatik, 2020.
6. Xiong Zheng and Vijay Garg. Byzantine Lattice Agreement in Asynchronous Systems. In *DROPS-IDN/v2/document/10.4230/LIPIcs.OPODIS.2020.4*. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2021. URL: `https://drops.dagstuhl.de/entities/document/10.4230/LIPIcs.OPODIS.2020.4`, `doi:10.4230/LIPIcs.OPODIS.2020.4`.
7. Xiong Zheng, Vijay K. Garg, and John Kaippallimalil. Linearizable Replicated State Machines with Lattice Agreement, October 2018. arXiv:1810.05871 [cs]. URL: `http://arxiv.org/abs/1810.05871`, `doi:10.48550/arXiv.1810.05871`.
8. Xiong Zheng, Changyong Hu, and Vijay K. Garg. Lattice Agreement in Message Passing Systems. In *DROPS-IDN/v2/document/10.4230/LIPIcs.DISC.2018.41*. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2018. URL: `https://drops.dagstuhl.de/entities/document/10.4230/LIPIcs.DISC.2018.41`, `doi:10.4230/LIPIcs.DISC.2018.41`.