An Improved Approximate Consensus Algorithm in the Presence of Mobile Faults

Lewis Tseng

Computer Science
Boston College

Email: lewis.tseng@bc.edu

Abstract. This paper explores the problem of reaching approximate consensus in synchronous point-to-point networks, where each pair of nodes is able to communicate with each other directly and reliably. We consider the *mobile Byzantine fault model* proposed by Garay '94 – in the model, an omniscient adversary can corrupt up to f nodes in each round, and at the beginning of each round, faults may "move" in the system (i.e., different sets of nodes may become faulty in different rounds). Recent work by Bonomi et al. '16 proposed a simple iterative approximate consensus algorithm which requires at least $4f + 1$ nodes. This paper proposes a novel technique of using "confession" (a mechanism to allow others to ignore past behavior) and a variant of reliable broadcast to improve the fault-tolerance level. In particular, we present an approximate consensus algorithm that requires only $\lceil 7f/2 \rceil + 1$ nodes, an $\lfloor f/2 \rfloor$ improvement over the state-of-the-art algorithms. Moreover, we also show that the proposed algorithm is *optimal* within a family of *round-based algorithms*.

1 Introduction

Fault-tolerant *consensus* has received significant attentions over the past three decades since the seminal work by Lamport, Shostak, and Pease [14]. Recently, a new type of fault model – *mobile fault model* – has been proposed to address the needs in emerging areas such as mobile robot systems, sensor networks, and smart phones [20]. The mobile fault model (in the round-based computation systems) has the following two characteristics:

- Up to f nodes may become faulty in a given round, and
- Different sets of nodes may become faulty in different rounds.

This type of fault model is very different from the traditional "fixed" fault model [14,15,2] – once a node becomes faulty, it remains faulty throughout the lifetime of the computation.

The mobile fault model is motivated by the observation that for long-living computations, e.g., aggregation, leader election, and clock synchronization, nodes may experience different phases throughout the lifetime such as cured/curing, healthy, and faulty phases [20]. For example, a worm-type of malware may gradually infect and corrupt healthy nodes while some infected nodes detected the malware and became cured (e.g., by routine checks from administrators) [20]. Another example is that fragile sensor nodes or robots may be impacted by the environment change, e.g., sensor malfunction due to high wind [5].

A rich set of mobile Byzantine fault models has been proposed [4,7,9,16], and subsequent work addressed the consensus problem in these models, e.g., [3,5,6]. These models are all defined over the *round-based computation* system (to be formally defined in Section 3.1), and they differ in two main dimensions [5,6]: (i) at which point in a round, faults can "move" to other nodes? and (ii) does a node have a knowledge when it is cured (i.e., after a fault moves to another node)? In this paper, we adopt the model proposed by Garay [9]:

- At the beginning of round t, the Byzantine adversary picks the set of up to f nodes that behave faulty in round t, and
- Once a node is *cured* (i.e., the node that was faulty in the previous round, and becomes fault-free in the current round), it is aware of the condition and can remain silent to avoid disseminating faulty information.

Recently, in Garay's model, Banu et al. [3] proposed an exact Byzantine consensus algorithm for at least $4f + 1$ nodes, and Bonomi et al. [5,6] proposed an iterative approximate Byzantine consensus algorithm for at least $4f+1$ nodes. Bonomi et al. also proved that for a constrained class of *memory-less* algorithms, their iterative algorithm is optimal. In this paper, we show that $4f+1$ is not tight for a more general class of algorithms. In particular, we present an approximate consensus algorithm that requires only $\lceil 7f/2 \rceil + 1$ nodes.

Mobile Faults and Round-based Algorithms The mobile Byzantine fault model considered in this paper is defined over round-based algorithms, in which the system proceeds in synchronous rounds that consist of three steps: *send, receive, compute* [5,6,9]. There are three types of nodes in the system: *faulty, healthy,* and *cured.* For a slight abuse of terminology, we also call *healthy* and *cured* nodes as *fault-free* nodes. In the round-based algorithms, each *fault-free* node maintains a special state variable v. After a sufficient number of rounds, the state variable v can be viewed as the *output* of the fault-free nodes.[1] With mobile faults, each node may become Byzantine faulty and have its local storage (including the state variable and other bookkeeping variables) corrupted in any round. When a node is cured, it needs to recover its state variable and potentially other information. Therefore, for a given round, we are only interested in the state variable v at the *healthy* nodes, since if majority of nodes remain healthy, *cured* nodes can easily learn a fault-free state variable from other nodes.

Approximate Consensus Approximate consensus can be related to many distributed computations in emerging areas, such as data aggregation [12], decentralized estimation [17], and flocking [11]; hence, the problem of reaching approximate consensus in the presence of Byzantine faults has been studied extensively, including synchronous systems [8], asynchronous systems [1], arbitrary networks [19], transient link faults [18], and time-varying networks [10] $\cdots$ etc. Bonomi et al. [5,6] are among the first to study approximate consensus algorithms in the presence of mobile Byzantine fault models.

Roughly speaking, the round-based approximate consensus algorithms of interest have the properties below, which we will define formally in Section 3.1:

- *Initial state* of each node is equal to a real-valued input provided to that node.
- *Validity*: after each round of an algorithm, the state variable v of each healthy node must remain in the range of the initial values of fault-free nodes.
- *Convergence*: for $\epsilon > 0$, after a sufficiently large number of rounds, the state variable of the healthy nodes are guaranteed to be within ϵ of each other.

Main Contribution

- We propose an approximate consensus algorithm that requires only $\lceil 7f/2 \rceil + 1$ nodes. The algorithm relies on "confession" (a mechanism to ask others to ignore past behavior) and a variant of reliable broadcast (to learn information from other *healthy* nodes reliably). The technique may be applied to other problems under the mobile fault models.
- We show that the proposed algorithm is *optimal* within a family of round-based algorithms, i.e., which only allows nodes to "remember" what happened in the previous rounds (but not the entire execution history).

---

[1] Using the technique from [1], nodes can also estimate the number of required rounds and decide when to "output" the state variable v.

2 Related Work

There is a rich literature on consensus-related problems [15,2]. Here, we only discuss two most relevant categories.

Exact Consensus under Mobile Byzantine Faults References [4,7,9,16,3] studied the problem of reaching *exact* consensus under different mobile Byzantine fault models. In exact consensus algorithms, every fault-free node reaches exactly the *same* output. Garay is among the first to study mobile faults [9]. In his model, the faults can "move" freely, and the cured nodes are aware of its condition. Garay proposed an algorithm requiring $6f + 1$ nodes [9]. Later, Banu et al. [3] improved the fault-tolerance level to $4f + 1$ nodes. References [4,16] considered a mobile fault model in which nodes are *not* aware when they are cured. Sasaki et al. [16] presented an algorithm requiring at least $6f + 1$ nodes, whereas, Bonnet et al. [4] proposed an algorithm requiring at least $5f + 1$ nodes, and proved that $5f + 1$ is tight in their fault model. Reference [7] also assumed that the nodes has the knowledge when it is cured; however, the ability of the adversary is more *constrained* than the above models. The adversary *cannot* choose an arbitrary set of nodes to be faulty, i.e., the faults can only "move" with message dissemination. Buhrman et al. [7] presented an optimal algorithm that requires $3f + 1$ nodes. References [4,7,9,16,3] considered only exact consensus; hence, the techniques are very different from the one in this paper. Moreover, to the best of our knowledge, we are the first to show that (approximate) consensus is solvable with only $\lceil 7f/2 \rceil + 1$ nodes in Garay's model.

Approximate Consensus Approximate consensus can be related to many distributed computations in networked systems, e.g., [12,17,11]. Since many networked systems are tend to be fragile, the problem of reaching approximate consensus in the presence of Byzantine faults has been studied extensively. Most work assumed the "fixed" fault model; that is, once the Byzantine adversary picks a faulty node, then throughout the execution of the algorithm, the node remains faulty and will not be cured. Dolev et al. studied the problem in both synchronous and asynchronous systems [8]. Dolev et al. proposed an optimal synchronous algorithm, but the asynchronous one requires at least $5f + 1$ nodes, which is only optimal within the family of *iterative* algorithms. Later, Abraham et al. proposed an optimal asynchronous algorithm that requires only $3f + 1$ nodes [1], which is optimal for all *general* algorithms. The technique in this paper is inspired by the usage of "witness" and reliable broadcast in [1]; however, due to different synchrony assumptions and fault models, our technique differs from the ones in [1] (we will address more details in Sections 4 and 5.1).

Kieckhafer and Azadmanesh studied the behavior of iterative algorithms (i.e., *memory-less* algorithms) and proved some lower bounds under Mixed-Mode faults model, where nodes may suffer crash, omission, symmetric, and/or asymmetric Byzantine failures [13]. Researchers also studied iterative approximate consensus under different communication assumptions, including arbitrary

communication networks [19], networks with transient link faults [18], and time-varying networks [10] $\cdots$ etc. These works only assumed fixed fault model.

Bonomi et al. [5,6] are among the first to study approximate consensus algorithms in the presence of mobile Byzantine fault models. They presented optimal *iterative* algorithms under different mobile fault models, and they proposed a mapping (or reduction) from the existing mobile Byzantine models to the Mixed-Mode faults model [13]. As we will show later in this paper, the bound does not hold for a more general class of algorithms. In other words, the "memory" from previous rounds helps improve the fault-tolerance level. This paper essentially demonstrates how to use the "memory" effectively.

3 Preliminary

3.1 Models and Round-based Algorithms

System Model We consider a synchronous message-passing system of n nodes. The communication is through a point-to-point network, in which each pair of nodes is connected by a direct communication link. All the links are assumed to be reliable, and the messages *cannot* be forged by the adversary. We assume that $n \geq \lceil 7f/2 \rceil + 1$, where f is the upper bound on the number of faulty nodes in a given round.

Round-based Algorithms As in the prior work [9,3,5,6], we consider the round-based algorithms in this paper. The algorithm consists of three steps:

- *Send*: send *one* message to all other nodes
- *Receive*: receive the messages from other nodes
- *Compute*: based on the messages and local states, perform local computation

In addition, each node also maintains a special state variable v such that after a sufficient number of rounds, the state becomes the *output* at the node. Note that Bonomi et al. only considered iterative algorithms [5,6], in which each node only sends and keeps a real-value state at all time, and there is no other information maintained (i.e., *memory-less* algorithms or *iterative* algorithms), whereas, we and references [9,3] consider a more general types of algorithms, where nodes may send and keep arbitrary state information.

Mobile Byzantine Fault Model In this paper, we consider mobile Byzantine fault model proposed by Garay [9]. There are three types of nodes:

- *Byzantine nodes*: in the beginning of each round, up to f nodes may become Byzantine faulty. A Byzantine faulty node may misbehave arbitrarily, and the local storage may be corrupted. Possible misbehavior includes sending incorrect and mismatching (or inconsistent) messages to different nodes. We consider an omniscient adversary – a single adversary that control which set

of nodes would become faulty. Moreover, the Byzantine adversary is assumed to have a complete knowledge of the execution of the algorithm, including the states of all the nodes, contents of messages the other nodes send to each other, and the algorithm specification.

– *Cured nodes*: a node is "cured" in the current round if it was faulty in the previous round, and becomes fault-free in the beginning of the current round. Under the model, a cured node has the knowledge that it just got cured, and hence can choose to stay silent at the current round, since the local states are potentially corrupted. A cured node follows the algorithm specification – it receives messages and performs local computation accordingly.

– *Healthy nodes*: all the other nodes belong to the set of healthy nodes. Particularly, they follow the algorithm specification, and the local storage is <u>not</u> corrupted in the past and current rounds.

3.2 Notation

Nodes: To facilitate the discussion, we introduce the following notations to represent sets of nodes throughout the paper:

– `faulty`$[t]$: the set of nodes that are faulty in round t
– `cured`$[t]$: the set of nodes that are cured in round t
– `healthy`$[t]$: the set of nodes that are healthy in round t

Nodes in `healthy`$[t] \cup$ `cured`$[t]$ are said to be *fault-free* in round t.

Values: Given a given round t, let us define $v[t],$ `max_state`$[t]$ and `min_state`$[t]$:

– $v_i[t]$ is the special state variable (that later will be the output) maintained at node i <u>in the end of round t</u>. Notation $v_i[0]$ is assumed to be the input given to node i. For brevity, when the round index or node index is obvious from the context, we will often ignore t or i.

– `max_state`$[t] = \max_{i \in \mathtt{healthy}[t] \cup \mathtt{cured}[t]} v_i[t]$. Notation `max_state`$[t]$ is the largest state variable among the fault-free nodes at the end of round t. Since the initial state of each node is equal to its input, `max_state`$[t]$ is equal to the maximum value of the initial input at the fault-free nodes.

– `min_state`$[t] = \min_{i \in \mathtt{healthy}[t] \cup \mathtt{cured}[t]} v_i[t]$. Notation `min_state`$[t]$ is the smallest state variable among the fault-free nodes at the end of round t. Since the initial state of each node is equal to its input, `min_state`$[t]$ is equal to the minimum value of the initial input at the fault-free nodes.

3.3 Correctness of Round-based Approximate Algorithms

We are now ready to formally state the *correctness condition* of round-based approximate algorithms under the mobile Byzantine fault model:

– *Validity*: $\forall t > 0$,

$$\texttt{min\_state}[t] \geq \texttt{min\_state}[0] \quad \text{and} \quad \texttt{max\_state}[t] \leq \texttt{max\_state}[0]$$

– *Convergence*: for a given constant ϵ, there exists an t such that

$$\texttt{max\_state}[t] - \texttt{min\_state}[t] < \epsilon$$

4 Algorithm CC

We now present *Algorithm CC (Consensus using Confession)*, a round-based approximate algorithm. Throughout the execution of the algorithm, each node i maintains a special state variable v_i. Recall that $v_i[t]$ represents the state at node i <u>in the end of round t</u> (i.e., after the state variable is updated). The convergence condition requires the state variables $v_i[t]$ to converge for a large enough t.

Similar to the algorithms in [9,1], Algorithm CC proceeds in *phases*. There are two phases in the algorithm: in the first phase (*Collection Phase*), nodes exchange their state variables v and construct a vector E that stores others' state variables. $E_i[j]$ represents the value that i receives from j. If j is faulty or cured, $E_i[j]$ may not be the state variable at node j. The second phase (*Confession Phase*) has three functionalities:

– Exchange the vector E constructed in the *Collection Phase*. If a node i is cured in the beginning of this phase (round $t+1$), then it sends $\emptyset$ to "confess" to all other fault-free nodes that it was faulty and subsequently, fault-free nodes will ignore messages from node i from the previous round. If node i is faulty, it may choose to send confession to only a subset of nodes; however, as long as we have enough redundancy, such misbehavior can be tolerated.

– Construct a vector V of "trustworthy" state variables. $V_i[j]$ represents the value what i believe is $v_j[t-1]$, the state variable at node j in the end of round $t-1$. A value u from node j is "trustworthy" if node j does not confess (*Condition 2* below), and enough nodes confess or "endorse" the value u (*Condition 1* below). Node k is said to endorse the value u if node k does not confess, sends legit message, and has $E_k[j] = u$. Node k may or may not be healthy.

– Update the local state variable using the `reduce` function on the vector V. The `reduce` function is designed to trim enough values from V so that none of the extreme values proposed by faulty nodes is used.

4.1 Algorithm Specification

Algorithm CC: Steps to be executed by node i in round t for $t \geq 0$

– **Round t:**   *+++ Collection Phase +++*
- *Send*:
    if i *is cured*,
         send $(\bot, i)$
    otherwise, send $(v_i[t-1], i)$

- *Receive*:[2]
    receive (u, j) from node j

- *Compute*:
    * $E_i[j] \leftarrow u$
    * if i *is healthy*,
             $v_i[t] \leftarrow v_i[t-1]$

– **Round $t+1$:**   *+++ Confession Phase +++*
- *Send*:
    if i *is cured*,
         send $(\emptyset, i)$ *//Comment: "confess" faulty behavior*
    otherwise, send (E_i, i)

- *Receive*:
    if a legitimate tuple (E_j, j) is received from node j,[3]

$$R_i[j] \leftarrow E_j$$

- *Compute*:
    * if the following two conditions are satisfied:
        · *Condition 1*:   there are $\geq n - f$ distinct nodes k such that (i) $R_i[k] = E_k \neq \emptyset$ and $E_k[j] = u$, or (ii) $R_i[k] = \emptyset$

        · *Condition 2*: $R_i[j] \neq \emptyset$

    then   *//Comment: u is "trustworthy"*

$$V_i[j] \leftarrow u$$

    otherwise,

$$V_i[j] \leftarrow \bot$$

    * update state variable as follows:

$$v_i[t+1] \leftarrow \texttt{reduce}(V_i)$$

---

[2] If nothing is received from j, then u is assumed to be $\bot$, a null value. Also, we assume that a node can send a message to itself.

[3] Here, $E_j = \emptyset$ is legitimate.

4.2 Reduce Function

Reduce function is widely used in iterative approximate Byzantine consensus algorithms, e.g., [1,5,6,13,8]. We adopt the same structure: order the values, trim potentially faulty values, and update local state. Different from the prior work, our `reduce` function trims different number of values at each round. The exact number depends on the number of $\perp$ values received. A $\perp$ value may be a result of faulty behavior or a confession. Below, we define the number of values to be trimmed.

Definition 1. *Suppose that node i receives $x \perp$ values in the vector V_i at round $t + 1$. Then, define*

$$nTrim_i = \begin{cases} f, & \text{if } x \leq f \\ \lceil f - \frac{x-f}{2} \rceil, & otherwise \end{cases}$$

The value `nTrim` counts the number of potentially faulty values in the vector V_i. In general, the more confessions that i sees in V_i, the less faulty values are in V_i. Lemma 6 formally shows that $nTrim_i$ is large enough to trim all the extreme values proposed by faulty nodes. Now, we present our `reduce` function below:

Reduce function: `reduce`(V_i) at node i

- Calculate $nTrim_i$ as per Definition 1.
- Remove all $\perp$ values in V_i. Denote the new vector by V_i'.
- Order V_i' in a non-decreasing order. Denote the ordered vector by O_i.
- Trim the bottom $nTrim_i$ and the top $nTrim_i$ values in O_i. In other words, generate a new vector containing the values $O_i[nTrim_i + 1], O_i[nTrim_i + 2], \cdots, O_i[|O_i| - nTrim_i - 1]$. Denote the trimmed vector by O_i^t.
- Return

$$\frac{\min(O_i^t) + \max(O_i^t)}{2} \tag{1}$$

---

5 Analysis

5.1 Key Properties of V

Before the `reduce` function is executed, the vector V at all fault-free nodes satisfies nice properties as stated in the lemmas below. The first four lemmas (Integrity I-IV) show that Algorithm CC achieves properties similar to reliable broadcast [1] – all fault-free nodes are able to see identical values in V if the sender node is either healthy or cured. Reliable broadcast in [1] also guarantees

Uniqueness – if the value sent from a node is not $\perp$, then the value appears identically in all fault-free node's V vector. However, the V vectors in Algorithm CC may still contain faulty values, since a faulty node that just moved in round $t + 1$ can send different E vectors to different fault-free nodes to "endorse" different values. This is the main reason that why Algorithm CC requires more than $3f + 1$ nodes. In the proofs below, we will often denote $v_i[t-1]$ by v for brevity. The indices should be clear from the context.

Lemma 1. (Integrity I) *If node i is healthy in both rounds t and $t + 1$, then for all fault-free $j \in$ healthy$[t+1] \cup$ cured$[t+1]$, $V_j[i] = v_i[t-1]$, the value sent by node i in round t.*

Proof. Fix a node $i \in$ healthy$[t] \cap$ healthy$[t+1]$ which sends the value $v_i[t-1]$ in round t. In the receive step of round t, each node $k \in$ healthy$[t] \cup$ cured$[t]$ receives the value and has $E_k[i] = v$. By definition, $|$healthy$[t] \cup$ cured$[t]| \geq n-f$. Suppose in the beginning of round $t + 1$, $b \leq f$ of the mobile Byzantine faults move to the nodes in healthy$[t] \cup$ cured$[t]$. Then, observe that

- $|$healthy$[t] \cup$ cured$[t]| - b$ healthy nodes send a legitimate tuple to all other nodes, and $E_k \neq \emptyset$ and $E_k[i] = v$ for node $k \in$ healthy$[t] \cup$ cured$[t] -$ faulty$[t+1]$. Denote this set of healthy nodes by A.
- Since b mobile faults move in round $t+1$, exactly b nodes are cured and send the confession ($\emptyset$) in round $t + 1$. Denote this set of cured nodes by B.

Note that nodes in $A \cup B$ are either cured or healthy; hence, all fault-free nodes will observe their behavior identically.

Now, consider a node $j \in$ healthy$[t+1] \cup$ cured$[t+1]$. From its perspective, *Condition 1* in the compute step in round $t + 1$ is met due to the observations above and the fact that $|A| + |B| \geq (|$healthy$[t] \cup$ cured$[t]| - b) + b = |$healthy$[t] \cup$ cured$[t]| \geq n - f$. Moreover, by definition, i is healthy in round $t + 1$; hence, *Condition 2* is also met. Therefore, node j will have $V_j[i] = v$. $\qquad\square$

Lemma 2. (Integrity II) *If node i is healthy in round t and becomes faulty in round $t + 1$, then for all fault-free $j \in$ healthy$[t + 1] \cup$ cured$[t + 1]$, either $V_j[i] = \perp$ or $V_j[i] = v_i[t-1]$, the value sent by node i in round t.*

Proof. The proof is by contradiction. Suppose that at some node $j \in$ healthy$[t+1] \cup$ cured$[t + 1]$, $V_j[i] = u$ such that $u \neq \perp$. Now, observe that:

- *Obs 1*: $V_j[i] = u$ only if there are enough node k that endorses or confesses (Condition 1 in Algorithm CC). Denote this set of nodes by W_u. And we have $|W_u| \geq n - f$.
- *Obs 2*: Since node i is healthy in round t, every node $k \in$ healthy$[t + 1]$ did *not* endorse value u (they heard value v and endorses v in round t).
- *Obs 3*: Obs 2 together with the fact that $|$healthy$[t + 1]| \geq n - 2f$ imply that there are $\leq n - (n - 2f) = 2f$ nodes in the set W_u.

We have $|W_u| \leq 2f < \lceil 7f/2 \rceil + 1 - f = \lceil 5f/2 \rceil + 1$, contradicting Obs 1. $\quad\square$

Lemma 3. (Integrity III) *If node i is cured in round t, then for all fault-free $j \in healthy[t+1] \cup cured[t+1]$, $V_j[i] = \perp$.*

The proof is similar to the proof of Lemma 1 and omitted here for brevity.

Lemma 4. (Integrity IV) *If node i is cured in round $t+1$, then for all fault-free $j \in healthy[t+1] \cup cured[t+1]$, $V_j[i] = \perp$.*

Proof. Since node i is cured in round $t+1$, it will send the confession ($\emptyset$) to all fault-free nodes in round $t+1$. Thus, for all $j \in \text{healthy}[t+1] \cup \text{cured}[t+1]$, $R_j[i] = \emptyset$, violating Condition 2. Therefore, $V_j[i] = \perp$. $\qquad \square$

The only case left is analyzing the behavior of nodes which remain faulty in both rounds t and $t+1$. These nodes are indeed able to produce different values in the V vectors at fault-free nodes; however, by construction, these nodes are limited in number. To see this, consider the following two scenarios:

- When no faulty node moves, i.e., $\text{faulty}[t] = \text{faulty}[t+1]$. Then, all fault-free nodes receive identical V vectors, since Condition 1 cannot be satisfied if faulty nodes send different values to different nodes in round t.
- When all faulty nodes move, i.e., $\text{faulty}[t] \cap \text{faulty}[t+1] = \emptyset$. Then, by Lemmas 2, 3, and 4, no fault-free nodes will see different values in round $t+1$.

The lemma below characterizes the bound on the number of different values.

Lemma 5. *Suppose $n \geq \lceil 7f/2 \rceil + 1$. For a pair of fault-free nodes $i, j \in healthy[t+1] \cup cured[t+1]$, at most $\lfloor f/2 \rfloor - 1$ non-$\perp$ values differs in V_i and V_j. In other words, there are $\geq n - \lceil f/2 \rceil + 1$ identical values in V_i and V_j.*

Proof. The proof is by contradiction. Suppose that there exists a pair of fault-free nodes i, j such that $\lfloor f/2 \rfloor$ different values appear in V_i and V_j. Consider the value belonging to some node k, i.e., $V_i[k] \neq V_j[k]$ and $V_i[k], V_j[k] \neq \perp$. Then, we can make the following observations:

- *Obs 1*: By Lemmas 1, 2, 3, and 4, node k must remain faulty in both rounds t and $t+1$.
- *Obs 2*: By Condition 1, there are $\geq n - f$ nodes that send the value $V_i[k]$ or send the confession ($\emptyset$) to node i in round $t+1$. Denote this set of nodes by W_i. For easiness of discussion, let us call these nodes the "witnesses" of the value $V_i[k]$.
- *Obs 3*: By assumption and Obs 1, at most $\lceil f/2 \rceil$ faults move from round t to round $t+1$.
- *Obs 4*: Among the nodes in W_i, at least $|W_i| - (f + \lceil f/2 \rceil)$ are nodes that are healthy in both rounds t and $t+1$. This is because (i) by Obs 3, at most $\lceil f/2 \rceil$ faults move, and (ii) cured node l in round $t+1$ (i.e., $l \in \text{cured}[t+1]$) send the confession ($\emptyset$) in round $t+1$, which result into $R_i[l] = R_j[l] = \emptyset$ in the receive step of round $t+1$.

Now, consider node j. By Obs 4, it has $\leq n - (|W_i| - (f + \lceil f/2 \rceil))$ witnesses of the value $V_j[k]$, since this is the number of nodes that are healthy in both rounds t and $t+1$ and endorses the value $V_i[k]$. Denote this set of witness of the value $V_j[k]$ by W_j. Then, we have

$$
\begin{aligned}
|W_j| &\leq n - (|W_i| - (f + \lceil f/2 \rceil)) \\
&\leq n - ((n - f) - (f + \lceil f/2 \rceil)) = \lceil 5f/2 \rceil \qquad \text{by Obs 2} \\
&< \lceil 5f/2 \rceil + 1 = n - f
\end{aligned}
$$

Therefore, Condition 1 is not satisfied at node j; hence, $V_j[k]$ can only be either the value $V_i[k]$ or $\bot$, a contradiction. $\qquad\square$

5.2 Correctness

For brevity, we only prove the correctness properties for healthy nodes, since cured nodes will have valid state variables if they remain fault-free in the next round. We begin with a useful lemma on `nTrim` (as per Definition 1). Here, a faulty value is the non-$\bot$ value sent by faulty nodes.

Lemma 6. *For a given odd round $t \geq 1$ and $i \in \mathtt{healthy}[i]$, there are at most $\mathtt{nTrim}_i$ faulty values in $V_i[t]$.*

Proof. If $V_i[t]$ contains $\leq f$ $\bot$ values, then the lemma holds by assumption. Now, consider the case when there are x $\bot$ values in $V_i[t]$, where $x > f$. There are only three ways to produce $\bot$ values: (i) by cured nodes in round $t - 1$ (due to Lemma 3), (ii) by cured nodes in round t (due to due to Lemma 4), and (iii) by faulty nodes in round t. Assume that b faults move in round t and b' faulty nodes produce $\bot$ values. Observe that (i) at most f cured nodes in round $t - 1$, (ii) exactly b cured nodes in round t, and (iii) exactly $f - (b + b')$ faulty values in $V_i[t]$. Then, we have

$$
\begin{aligned}
\mathtt{nTrim}_i &= \lceil f - \frac{x - f}{2} \rceil \qquad\qquad\qquad \text{by Definition 1} \\
&\geq \lceil f - \frac{(b + f + b') - f}{2} \rceil = \lceil f - \frac{b + b'}{2} \rceil \qquad \text{by observations above} \\
&\geq f - (b + b') = \text{number of faulty values in } V_i \qquad\qquad \square
\end{aligned}
$$

Lemma 7. (Validity) *For a given round $t \geq 0$, if $i \in \mathtt{healthy}[t]$, then*

$$
\mathtt{max\_state}[0] \geq v_i[t] \geq \mathtt{min\_state}[0]
$$

Proof. The proof is by induction on the number of rounds.

- *Initial Step*: When $t = 0$, the statement holds, since by definition,
  $\mathtt{max\_state}[0] \geq v_i[0] \geq \mathtt{min\_state}[0]$.

   – *Induction Step*: suppose the statement holds for some $h > 0$, consider round $h+1$. If h is a *Collection Round* (h is even), then the statement holds trivially, since $v_i[h] \leftarrow v_i[h-1]$ in the compute step. Now, consider the case when h is odd (h is an *Update Round*). Lemma 6 implies that in the trim step of the `reduce` function (the fourth step), all the faulty values will be trimmed if they are too large or too small. Therefore, the maximal and minimal values in O_i^t will always be inside the range of the maximal and minimal values of state values of the fault-free nodes in round h. Hence the return value of the `reduce` function satisfies Validity by the induction hypothesis. $\qquad\square$

Before proving convergence, we show a lemma that bounds the range of the updated state variables. Recall that `max_state`[t] and `min_state`[t] represent the maximal and minimal state variables, respectively, at healthy nodes in round t. We only care about the state variables in the even round, since in the odd round, the state variable remains the same at healthy nodes.

Lemma 8. *For some even integer $t > 0$, we have*

$$\texttt{max\_state}[t+1] - \texttt{min\_state}[t+1] \leq \frac{\texttt{max\_state}[t-1] - \texttt{min\_state}[t-1]}{2}$$

Proof. To prove the lemma, we need to show that for any pair of fault-free nodes i, j, we have

$$|v_j[t+1] - v_i[t+1]| \leq \frac{\texttt{max\_state}[t-1] - \texttt{min\_state}[t-1]}{2} \qquad (2)$$

Let V_i and V_j denote the V vectors at i and j, respectively, at the compute step (the third step) of round $t+1$. Then, define $R = V_i \cap V_j$. Recall that O_i^t and O_j^t represent the trimmed vector in the `reduce` function at i and j, respectively. Then, we have the following key claim:

Claim. Let m be the median of the values in R. Then, $m \in O_i^t$ and $m \in O_j^t$.

Proof. We make the following observations:

   – *Obs 1*: By Lemma 5, $|R| \geq n - \lceil f/2 \rceil + 1 \geq 3f + 1$.
   – *Obs 2*: Suppose there are $x \perp$ values in R. Consider two cases:
      • Case I: if $x \leq f$, then $m \in O_i^t$, because after removing $f \perp$ values from V_i, we trim f elements from each side. Similarly, we can show $m \in O_j^t$.
      • Case II: if $x \geq f$, then $m \in O_i^t$, because after removing $x \perp$ values from V_i, we trim `nTrim`$_i$ elements from each side. In other words, we trim at most

$$x + 2 * \texttt{nTrim}_i = x + 2\lceil(f - \frac{x-f}{2})\rceil = x + 2f - x + f = 3f$$

      values from R. This together with Obs 1 implies that $m \in O_i^t$. Similarly, we can show $m \in O_j^t$.
   These two cases proves the claim. $\qquad\square$

The rest of the proof of Lemma 8 follows from the claim using the standard tricks from prior work, e.g., [15,1,19]. We include the proof in Appendix A. □

Lemma 8 and simple arithmetic operations imply the following:

Lemma 9. (Convergence) *Given a $\epsilon > 0$, there exists a round t such that* $\texttt{max\_state}[t] - \texttt{min\_state}[t] < \epsilon$.

Lemmas 7 and 9 imply that Algorithm CC is correct:

Theorem 1. (Correctness) *Algorithm CC solves approximate consensus in under Garay's model given that $n \geq \lceil 7f/2 \rceil + 1$.*

6 Impossibility Result

This section proves that for a certain family of round-based algorithms, $\lceil 7f/2 \rceil + 1$ is the lower bound on the number of nodes (fault-tolerance level), proving that Algorithm CC is optimal within this family of algorithms.

2-Memory Round-based Algorithms As discussed before, the iterative algorithms considered in [5,6,8,19] are *memory-less*, i.e., it can only send its own state, and it updates state in every round. As proved in [5,6], such type of memory-less algorithms requires $4f + 1$ nodes. For the lower bound proof, we consider a slightly more general type of algorithms – 2-*memory round-based algorithms* – in which nodes can send arbitrary messages, carry information from the previous round, but nodes have to update their state variables every two rounds (hence, the name 2-memory). While the definition seems constrained, many Byzantine consensus algorithms belong to this family of algorithms, e.g., [9,3,1]. Note that the original algorithm proposed by Lamport, Shostak, and Pease [14] *does not* belong to 2-memory round-based algorithms, as nodes collect many more rounds of information before updating their state variables.

Lower Bound Proof The lower bound proof is similar to the lower bound proof for iterative algorithms, e.g., [8,19]; however, we also need to consider how faulty nodes move, which makes the proof slightly more complicated. Note that using Integrity I-IV (Lemmas 1, 2, 3, and 4), it is fairly easy to show that for $f = 1$, Algorithm CC solves the problem for $n = 3f + 1 = 4$.

Theorem 2. *It is impossible for any 2-memory round-based algorithm to solve approximate consensus under Garay's model if $n \leq \lceil 7f/2 \rceil$ and $f > 1$.*

Proof. Consider the case when $f = 2$, and $n = 7$. Denote by the set of nodes $S = \{a, b, c, d, e, f, g\}$. For simplicity, assume that a node can be in the *cured* phase in round 0. Then, suppose in round 0: a, b are cured, c, d are faulty, and e, f, g are healthy. And, nodes e, f has input m, and node g has input m', where $m' > m$ and $m' - m > \epsilon$.

In round 0, faulty nodes c, d behave to nodes a, e, f as if they have input m, and behave to nodes b, g as if they have input m'. In the beginning of round 1, the adversary moves the fault from node d to node e; hence, a, b, f, g are healthy, c, e are faulty, d is cured in round 1. The new faulty node e and the original faulty node c behave in the following way (i) behave to nodes a, d, f as if node c, d, e have input m, (ii) behave to node b, g as if nodes c, d, e have input m', and (iii) otherwise follow the algorithm specification.

Now, from the perspective of node f, there are two scenarios:

- If nodes c, d are faulty, then the fault-free inputs are m, m, m', and
- If nodes d, g are faulty, then the fault-free inputs are m, m, m, and

By assumption, node e needs to update the state variable now and it could not distinguish from the two scenarios, since it cannot exchange more messages. Therefore, node e must choose some value that satisfies the validity condition in *both* scenarios, and the value is m.[4] Therefore, in round 1, the state variable at node e remains m. We can show the same situation holds for node a, d.

From the perspective of node g, there are also two scenarios:

- If nodes c, d are faulty, then the fault-free inputs are m, m, m', and
- If nodes e, f are faulty, then the fault-free inputs are m', m', m', and

Then, node g has to choose m' to satisfy the validity condition in round 1. Similarly, node b has to choose m'.

Then in round 2, the adversary picks nodes a, b to be faulty. Observe that this scenario is identical to round 0: two cured nodes, two faulty nodes, and three healthy nodes with state variables m, m, and m'. Therefore, the adversary can behave in the same way so that no healthy node will change their state variables; hence, convergence cannot be achieved. □

7 Conclusion

Under Garay's mobile Byzantine fault model [9], we present an approximate consensus algorithm that requires only $\lceil 7f/2 \rceil + 1$ nodes, an $\lfloor f/2 \rfloor$ improvement over the state-of-the-art algorithms [5,6]. Moreover, we also show that the proposed algorithm is *optimal* within the family of 2-memory round-based algorithms. Whether $\lceil 7f/2 \rceil + 1$ is tight for general approximate algorithms remains open.

References

1. I. Abraham, Y. Amit, and D. Dolev. Optimal resilience asynchronous approximate agreement. In *OPODIS*, pages 229–239, 2004.

---

[4] There are other scenarios not discussed in the proof for brevity; however, m is the only value works for each of the scenarios.

2. H. Attiya and J. Welch. *Distributed Computing: Fundamentals, Simulations, and Advanced Topics*. Wiley Series on Parallel and Distributed Computing, 2004.
3. N. Banu, S. Souissi, T. Izumi, A. N. Bessani, M. Correia, N. F. Neves, H. Buhrman, and J. A. Garay. An improved byzantine agreement algorithm for synchronous systems with mobile faults. 2012.
4. F. Bonnet, X. Défago, T. D. Nguyen, and M. Potop-Butucaru. *Tight Bound on Mobile Byzantine Agreement*, pages 76–90. Springer Berlin Heidelberg, 2014.
5. S. Bonomi, A. D. Pozzo, M. Potop-Butucaru, and S. Tixeuil. Approximate agreement under mobile byzantine faults. *CoRR*, abs/1604.03871, 2016.
6. S. Bonomi, A. D. Pozzo, M. Potop-Butucaru, and S. Tixeuil. Approximate agreement under mobile byzantine faults. In *2016 IEEE 36th International Conference on Distributed Computing Systems (ICDCS)*, pages 727–728, June 2016.
7. H. Buhrman, J. A. Garay, and J. H. Hoepman. Optimal resiliency against mobile faults. In *Twenty-Fifth International Symposium on Fault-Tolerant Computing. Digest of Papers*, pages 83–88, June 1995.
8. D. Dolev, N. A. Lynch, S. S. Pinter, E. W. Stark, and W. E. Weihl. Reaching approximate agreement in the presence of faults. *J. ACM*, 33:499–516, May 1986.
9. J. A. Garay. Reaching (and maintaining) agreement in the presence of mobile faults (extended abstract). In *Proceedings of the 8th International Workshop on Distributed Algorithms*, WDAG '94, pages 253–264, London, UK, UK, 1994. Springer-Verlag.
10. A. Haseltalab and M. Akar. Approximate byzantine consensus in faulty asynchronous networks. In *2015 American Control Conference (ACC)*, pages 1591–1596, July 2015.
11. A. Jadbabaie, J. Lin, and A. Morse. Coordination of groups of mobile autonomous agents using nearest neighbor rules. *Automatic Control, IEEE Transactions on*, 48(6):988 – 1001, june 2003.
12. D. Kempe, A. Dobra, and J. Gehrke. Gossip-based computation of aggregate information. pages 482–491. IEEE Computer Society, 2003.
13. R. M. Kieckhafer and M. H. Azadmanesh. Reaching approximate agreement with mixed-mode faults. *IEEE Transactions on Parallel and Distributed Systems*, 5(1):53–63, Jan 1994.
14. L. Lamport, R. Shostak, and M. Pease. The byzantine generals problem. *ACM Trans. Program. Lang. Syst.*, 4(3):382–401, July 1982.
15. N. A. Lynch. *Distributed Algorithms*. Morgan Kaufmann, 1996.
16. T. Sasaki, Y. Yamauchi, S. Kijima, and M. Yamashita. *Mobile Byzantine Agreement on Arbitrary Network*, pages 236–250. Springer International Publishing, Cham, 2013.
17. I. Schizas, A. Ribeiro, and G. Giannakis. Consensus in ad hoc WSNs with noisy links – Part I: Distributed estimation of deterministic signals. *Signal Processing, IEEE Transactions on*, 56(1):350–364, Jan 2008.
18. L. Tseng and N. H. Vaidya. Iterative approximate consensus in the presence of Byzantine link failures. In *Networked Systems - Second International Conference, NETYS 2014, Marrakech, Morocco, May 15-17, 2014. Revised Selected Papers*, pages 84–98, 2014.
19. N. H. Vaidya, L. Tseng, and G. Liang. Iterative approximate Byzantine consensus in arbitrary directed graphs. In *Proceedings of the thirty-first annual ACM symposium on Principles of distributed computing*, PODC '12. ACM, 2012.
20. M. Yung. The mobile adversary paradigm in distributed computation and systems. In *Proceedings of the 2015 ACM Symposium on Principles of Distributed Computing*, pages 171–172. ACM, 2015.

Appendices

A   Proof of Lemma 8

The claim proved in Lemma 8 implies that $\max(O_i^t) \geq m$, and by the last step of the `reduce` function and the fact that all the extreme value proposed by faulty nodes are trimmed (by Lemma 6), we have $\min(O_i^t) \geq \mathtt{min\_state}[t-1]$. Therefore,

$$v_i[t+1] \geq \frac{m + \mathtt{min\_state}[t-1]}{2}$$

Similarly, we can show $\min(O_i^t) \leq m$, $\max(O_i^t) \geq \mathtt{max\_state}[t-1]$, and

$$v_j[t+1] \leq \frac{m + \mathtt{max\_state}[t-1]}{2}$$

Now, we need to show that (2) holds. Without loss of generality, assume that $v_j[t+1] \geq v_i[t+1]$. Then, we have

$$v_j[t+1] - v_i[t+1] \leq \frac{m + \mathtt{max\_state}[t-1]}{2} - \frac{m + \mathtt{min\_state}[t-1]}{2}$$
$$= \frac{\mathtt{max\_state}[t-1] - \mathtt{min\_state}[t-1]}{2}$$

This completes the proof.