Impossibility of $n - 1$-strong-equilibrium for Distributed Consensus with Rational Agents

Amit Jacob-Fanani

Tel-Aviv University

amitj@mail.tau.ac.il

Itay Harel

Tel-Aviv University

itayharel@mail.tau.ac.il

Abstract

An algorithm for $n - 1$-strong-equilibrium for distributed consensus in a ring with rational agents was proposed by Afek et al. [1]. A proof of impossibility of $n - 1$-strong-equilibrium for distributed consensus in every topology with rational agents, when $n$ is even, is presented. Furthermore, we show that the only algorithm which can solve the problem when $n$ is odd is the one proposed in [1]. Finally, we prove that the proposed algorithm provides a $n - 2$-strong-equilibrium in a synchronous ring when $n$ is even.

I. INTRODUCTION

The problem of distributed consensus, where $n$ processors, some of which may be faulty, have to agree on the same value which is also an input to one of them, has been well studied [3-7]. More recently, there is growing interest in the model of distributed computing with rational agents which may deviate from a distributed algorithm in order to affect the result of the algorithm [8-12].

The main concern regarding distributed computation with rational agents is to provide an algorithm which is resilient to rational agents, meaning that rational agents would gain nothing by deviating from the protocol. In order to address this more formally, Afek et al. use the following definitions (refer to [1] for full details):

Definition 1 (Utility Function). Let $A$ be a distributed algorithm, and $O$ be the set of all possible final states of $A$. Let $p$ be an agent, then $u_p : O \rightarrow \mathbb{R}$ is the utility function, such that $\forall o \in O$, $u_p(o)$ is the profit $p$ would gain from reaching the final state $o$.

Definition 2 (Solution Preference). Let $A$ be a distributed algorithm, and $O$ be the set of all possible final states of $A$. Let $p$ be an agent, and $u_p$ be its utility function, then $u_p$ satisfy the solution preference iff $\forall o_L, o_E \in O$ such that $o_L$ is a legal final state of $A$ and $o_E$ is an erroneous final state of $A$, $u_p(o_L) \geq u_p(o_E)$.

Definition 3 (K-Strong-Equilibrium). Let $A$ be a distributed algorithm, and $P$ the set of processors, then $A$ reaches $k$-strong equilibrium iff $\forall C \subseteq P$, such that $|C| \leq k$, for every $p \in C$ the following holds:

$$E[ u_p | C \text{ deviates from } A ] \leq E[ u_p | C \text{ runs } A ]$$

The problem of distributed consensus with rational agents is addressed in [1] and in [2]. In the model proposed in [1], some processors may prefer a certain outcome of the protocol, i.e. that all processors eventually agree on the value 1 or that all agree on the value 0.

Afek et al. propose in [1] an algorithm providing a $n - 1$-strong-equilibrium for the problem of distributed consensus with rational agents in a synchronous ring. The algorithm is presented in IV.

In this paper we show that this algorithm fails in case $n$ is even. Moreover, we show that a $n - 1$-strong-equilibrium is unattainable when $n$ is even, in every topology (II). Nonetheless, we prove that it is the only algorithm that provides a $n - 1$-strong-equilibrium in a synchronous ring, when $n$ is odd (III), and that the algorithm does provide a $n - 2$-strong-equilibrium in a synchronous ring (IV).
II. IMPOSSIBILITY OF $n-1$-STRONG-EQUILIBRIUM WHEN $n$ IS EVEN

**Definition 4** (Input Cheaters). Let $A$ be an algorithm for distributed consensus and let $C$ be a coalition of cheaters, then $C$ are input cheaters iff $C$ run $A$ but may act as if they have a different set of inputs than they are actually given.

**Theorem 1** (Impossibility of $n−1$-Strong-Equilibrium). Assuming uniform distribution over the inputs, there is no algorithm for distributed consensus, in every topology, providing a $n−1$-strong-equilibrium when $n$ (the number of nodes) is even.

**Proof.** Assume by contradiction that there is an algorithm $A$ that provides a $n−1$-strong-equilibrium.

Given a coalition of size $n−1$, denote the single node which is not part of this coalition by $ν$, and the group of nodes in the coalition (i.e. $V/\{ν\}$) by $C$.

We will show that even when we bound $C$ to be input cheaters, $A$ will fail to provide a $n−1$-strong-equilibrium.

We will use the following notations:

1. $D(ν)$ – The value decided by a node $ν$ when it finishes to run $A$.
2. $J(ν)$ – The input that a node $ν ∈ C$ pretends to have.
3. $I(ν)$ – The input a node $ν$ is given (before it starts running $A$).
4. $I(ν) = (I(ν))_{C}$, i.e. the set of inputs given to the nodes of $C$.
5. $J(ν) = (J(ν))_{C}$, i.e. the set of inputs that the nodes of $C$ pretend to have.
6. For $j : C → \{0,1\}$ define $ones(j) = |\{ν ∈ C | J(ν) = 1\}|$

Assume $I(ν) = 1$ and $∀ ν ∈ C : J(ν) = 1$, then by validity attribute of consensus $D(ν) = 1$ is deduced (since $ν$ cares only about $J(ν)$ and cannot know that $I(ν) ≠ J(ν)$).

Formally,

$$\mathbb{P}(D(ν) = 1|I(ν) = 1, ones(J(ν)) = |C|) = 1$$

And similarly,

$$\mathbb{P}(D(ν) = 0|I(ν) = 0, ones(J(ν)) = 0) = 1$$

Define $α(j)$ the parity of $ones(j)$, i.e. $α(j) ≡ ones(j) mod 2$.

**Lemma 1.1.** Assume $C$ runs $A$ with $J(ν) = j$, then $D(ν) = ¬(I(ν) \oplus α(j))$.

**Proof.** We will prove the lemma by backwards induction on $ones(j)$. If $ones(j) = n−1$, then $α(j) = 1$.

If $I(ν) = 1$ then using (1) we conclude $D(ν) = 1$, and indeed $1 = ¬(1 \oplus 1)$.

Otherwise, $I(ν) = 0$. Assume that $I(ν) = (0,...,0)$ and that $∀ p ∈ C : u_{p}(0) = 0, u_{p}(1) = 1$ i.e. every player in $C$ prefers that $D(ν) = 1$.

If $C$ did not cheat, their expected utilization was:

$$\mathbb{E}[u_{p}|J(ν)| I(ν) = (0)_{C}] = \mathbb{P}(D(ν) = 1|J(ν) = (0)_{C}) = \mathbb{P}(I(ν) = 0) = 1 \cdot \mathbb{P}(I(ν) = 1) = \frac{1}{2}$$

Where $=_{1}$ is a result of (2).

On the other hand, if $C$ cheated with $ones(j) = n−1$, i.e. $j = (1)_{C}$, then their expected utilization was:
(4) $\mathbb{E}[u_p|I(C) = (1)_c] = \mathbb{P}(D(v) = 1|I(C) = (1)_c) = \mathbb{P}(D(v) = 1|I(C) = (1)_c, I(v) = 1) \cdot \mathbb{P}(I(v) = 1) + \mathbb{P}(D(v) = 1|I(C) = (1)_c, I(v) = 0) \cdot \mathbb{P}(I(v) = 0) = \frac{1}{2} + \frac{1}{2} \mathbb{P}(D(v) = 1|I(C) = (1)_c, I(v) = 0)$

Since $A$ provides a $n - 1$-strong-equilibrium then $C$ should not benefit from cheating. Formally,

(5) $\forall p \in C: \mathbb{E}[u_p|I(C) = j(C)] \geq \mathbb{E}[u_p|I(C) = (1)_c]$  
Combining (3), (4) and (5) we have

(6) $\forall p \in C: \frac{1}{2} \geq \mathbb{E}[u_p|I(C) = j(C)] \geq \mathbb{E}[u_p|I(C) = (1)_c] = \frac{1}{2} + \frac{1}{2} \mathbb{P}(D(v) = 1|I(C) = (1)_c, I(v) = 0)$

Hence, $\mathbb{P}(D(v) = 1|I(C) = (1)_c, I(v) = 0) = 0$, thus $D(v) = 0$. Indeed $0 = -\alpha(0 \oplus 1)$.

We will assume the statements hold for ones($j$) = $k$ and prove it for ones($j$) = $k - 1$.

Consider the first case $\alpha(j) = 0$, i.e. $k = 0$ mod 2 = 1.

If $I(v) = 0$ then we can assume that $I(C) = (1)_c$ and that $\forall p \in C: u_p(0) = 1, u_p(1) = 0$ i.e. every player in $C$ prefers that $D(v) = 0$. If $C$ did not cheat, their expected utilization was:

$\mathbb{E}[u_p|I(C) = I(C)] = \mathbb{P}(D(v) = 0|I(C) = (1)_c) = \mathbb{P}(D(v) = 0|I(C) = (1)_c, I(v) = 0) \cdot \mathbb{P}(I(v) = 0)$

$\leq \mathbb{P}(I(v) = 0) = \frac{1}{2}$

Where $=_{1}$ is a result of (1).

Notice that from the case $I(v) = 1$ above, $\mathbb{P}(D(v) = 0|I(C) = j, I(v) = 1) = 1$ follows. Therefore, if $C$ did cheat, the expected utilization of $C$ was:

$\mathbb{E}[u_p|I(C) = j] = \mathbb{P}(D(v) = 0|I(C) = j) = \mathbb{P}(D(v) = 0|I(C) = j, I(v) = 1) \cdot \mathbb{P}(I(v) = 1) + \mathbb{P}(D(v) = 0|I(C) = j, I(v) = 0) \cdot \mathbb{P}(I(v) = 0) = \frac{1}{2} + \frac{1}{2} \mathbb{P}(D(v) = 0|I(C) = j, I(v) = 0)$

Since $A$ provides a $n - 1$-strong-equilibrium then $C$ should not benefit from cheating. Formally,

$\forall p \in C: \frac{1}{2} \geq \mathbb{E}[u_p|I(C) = j(C)] \geq \mathbb{E}[u_p|I(C) = j] = \frac{1}{2} + \frac{1}{2} \mathbb{P}(D(v) = 0|I(C) = j, I(v) = 0)$

Thus, we have $\mathbb{P}(D(v) = 0|I(C) = j, I(v) = 0) = 0$, so $D(v) = 1$. Indeed $1 = -\alpha(0 \oplus 0)$.

The second case $\alpha(j) = 1$, is symmetric.

In case ones($j$) = 0 and $I(v) = 0$, by Lemma 1.1 we have that $D(v) = -\alpha(0 \oplus 0) = 1$, contradicting (2).

III. SINGULARITY OF THE SOLUTION ALGORITHM WHEN n IS ODD

Assume $A$ is an algorithm providing a $n - 1$-strong-equilibrium when $n$ is odd. We can repeat the proof of Lemma 1.1, replacing the claim with $D(v) = I(v) \oplus \alpha(j)$.

The proof works exactly the same, but the result of this lemma doesn't contradict (2).
Instead, we have that even number of nodes with an input equal to 1 results in \( D(v) = 0 \) and odd number of nodes with an input equal to 1 results in \( D(v) = 1 \). Thus, if we look at \( A \) as a function that maps between the nodes’ inputs and the output of the distributed consensus, \( A \) is exactly equal to the algorithm given in [1].

IV. A \( n - 2 \)-Strong-Equilibrium

**Theorem 2** (A \( n - 2 \)-Strong-Equilibrium Algorithm) Assuming uniform distribution over the inputs and a synchronous ring of even size, the algorithm proposed in [1] works and provides a \( n - 2 \)-strong-equilibrium.

**Proof.** To simplify, we shall start by assuming the ring is unidirectional. A method to discard this assumption regardless of the cheating coalition is shown in [2].

We shall present the algorithm proposed in [1]:

**Algorithm 1** (Afek et al. [1]):

```latex
\begin{algorithm}
do_consensus(input, id):
\begin{algorithmic}[1]
\State my_rand := rand()
\State ids_array.append(<id, input>)
\State rand_sum := my_rand
\State input_sum := input
\State Send(<input, id, my_rand>, out_interface)
\For {i = 1,...,n-1:}
\State tmp := Recv(in_interface)
\If {tmp = null or not (tmp.input \in \{0, 1\})}:
\State REPORT CHEATER
\State ids_array.append(<tmp.id, tmp.input>)
\State rand_sum += tmp.rand
\State input_sum += tmp.input
\State Send(tmp, out_interface)
\EndFor
\EndFor
\State tmp := Recv(in_interface)
\If {tmp.input \neq input or tmp.id \neq id or tmp.rand \neq my_rand:}
\State REPORT CHEATER
\State sort ids_array by id
\If {\exists id \in ids_array with ids_array.count(id) \neq 1:}
\State REPORT CHEATER
\State leader := ids_array[rand_sum % n]
\State input_sum += leader.input
\If {input_sum % 2 == 1:}
\State return 1
\Else:
\State return 0
\EndIf
\EndIf
\EndIf
\Enddo_consensus
\end{algorithmic}
\end{algorithm}
```

**Lemma 2.1** If all nodes in the ring comply with the algorithm, it maintains both validity and agreement, and \( \mathbb{P}(\text{ring decides } 0) = \mathbb{P}(\text{ring decides } 1) = \frac{1}{2} \).

**Proof.** Addressing validity – if every node got 0 as an input, then for every node in the ring \( u.\text{input}_\text{sum} \) will be 0, meaning that \( u.\text{input}_\text{sum} \text{ mod } 2 = 0 \) resulting in all nodes agreeing on 0. If every node got 1 as input, then for every node in the ring \( u.\text{input}_\text{sum} \) will be \( n + 1 \), and since \( n \) is even – it will result in 1 as consensus. If some nodes got 0 and others got 1, every outcome is acceptable in terms of validity.

Addressing agreement – if all nodes comply then every node after line 18 will have the exact same values for \( \text{rand}_\text{sum}, \text{input}_\text{sum} \) and \( \text{ids}_\text{array} \) (follows from a simple induction), which will result in all nodes agreeing on the same value.
Addressing $\mathbb{P}(\text{ring decides 0}) = \mathbb{P}(\text{ring decides 1}) = \frac{1}{2}$ - we shall prove the following claims by induction on the number of nodes:

1. Given nodes $v_1, v_2, \ldots, v_{2k}$: $\mathbb{P}\left(\sum_{i=1}^{2k} v_i. \text{input } \text{mod} 2 = 0\right) = \mathbb{P}\left(\sum_{i=1}^{2k} v_i. \text{input } \text{mod} 2 = 1\right) = \frac{1}{2}$

2. Given nodes $v_1, v_2, \ldots, v_{2k}$:
   $$\mathbb{P}\left(\sum_{i=1}^{2k} v_i. \text{input} + \text{leader. input } \text{mod} 2 = 0\right) = \mathbb{P}\left(\sum_{i=1}^{2k} v_i. \text{input} + \text{leader. input } = 1\right) = \frac{1}{2}$$

If $n = 2$ (we shall denote the nodes as $v_1, v_2$): Addressing the first claim - there are 4 possible inputs that can be given to the nodes - $<0,0>$, $<0,1>$, $<1,0>$, $<1,1>$. In half of them the sum of all inputs is even, in the other half - odd.

Addressing the second claim - if $v_1$ and $v_2$ both got 0 as an input – the sum (including the leader) is even. If both got 1 - the sum (including the leader) is odd. If $v_1$ got 0 and $v_2$ got 1 – then if $v_1$ is elected leader they will decide on 1, else they will decide on 0. The exact opposite result will occur if $v_1$ got 1 and $v_2$ got 0. In conclusion – in exactly half of the cases the sum (including leader) is even, and in the other half – odd.

Now, assume the above claims applies to all rings of size $n = 2(k - 1)$. We shall prove it applies to a ring of size $n = 2k$.

Addressing claim 1 - By applying the same analysis done in the base case for nodes $n_{2k}, n_{2k-1}$ and by using the induction hypothesis, the claim is proven.

Addressing claim 2 – Denote the nodes as $v_1, v_2, \ldots, v_{2k}$.

If $v_{2k}$ and $v_{2k-1}$ aren’t elected as leader, then their contribution to input_sum is odd (exactly 1) half of the times and even (0 or 2) the other half. By induction hypothesis (claim 2), we know that if one of the nodes $v_1, v_2, \ldots, v_{2k-2}$ was elected leader, then in half of the cases their sum of inputs (including leader) is 0, and in the other half – it is 1. Thus the contribution of $v_{2k-1}, v_{2k}$ maintains the probability of 0 or 1 to be decided.

If $v_{2k}$ or $v_{2k-1}$ are elected as leader, then by applying claim 1 on $v_1, v_2, \ldots, v_{2k-2}$ and claim 2 on $v_{2k-1}, v_{2k}$, the claim is proven.

From claim 2, it is clear that $\mathbb{P}(\text{ring decides 0}) = \mathbb{P}(\text{ring decides 1}) = \frac{1}{2}$.

Each node in the ring is initialized with 3 values - $<\text{id, input, random}>$. Denote Trip($u$) - the triplet of values with which $u$ is initialized in the beginning of the algorithm; and Trip($v, u$) - a triplet of values received by $u$ during the algorithm in which Trip($v, u$).id = $v$. Note that input and random may or may not be the actual values with which $v$ was initialized in the beginning. Furthermore, the id $v$ may or may not be an id of an actual node in the ring.

Let $v \in C, u \notin C$ such that $v$ is the closest member of the coalition to $u$ from $u$’s in_interface. Also, let $m$ be the number of nodes between $v$ and $u$, including $u$. It is clear that $v$ (as a member of the coalition) may lie regarding $n - m$ of the triplets sent to $u$ (thus, giving the coalition control over $n - m$ messages) but cannot affect $m$ of the triplets received by $u$ (Trip($u$) and the triplets of the $m - 1$ nodes between $v$ and $u$). Further, $v$ receives these $m$ triplets after it had sent all $n - m$ triplets controlled by him. Of course, $v$ cannot change the triplets it sends after it had sent them, which implies that any information that $u$ received from $v$ is independent of any information that $v$ received from $u$.

In order to reflect this conclusion, we define the following:

1. $R(v, u)$ – the set of nodes between $v$ and $u$, including $u$.
2. Lie($u$) = {Trip($w, u$)$|w \notin R(v, u)$}. These triplets are fully controlled by $C$.
3. Truth($u$) = {Trip($w, u$)$|w \in R(v, u)$}. These aren’t controlled by $C$.
4. InputT($u$) = $\sum_{\text{Trip} \in \text{Truth}(u)} \text{Trip}. \text{input}$
5. InputL($u$) = $\sum_{\text{Trip} \in \text{Lie}(u)} \text{Trip}. \text{input}$
6. RandomT($u$) = $\sum_{\text{Trip} \in \text{Truth}(u)} \text{Trip}. \text{random}$
7. RandomL($u$) = $\sum_{\text{Trip} \in \text{Lie}(u)} \text{Trip}. \text{random}$
Lemma 2.2 Given a coalition of cheaters denoted C, and a node $u \notin C$, then every id in $u$.ids_array has the same probability of $\frac{1}{n}$ to be elected as leader by $u$, regardless of the actions taken by C.

Proof. This lemma has been explored and proved in [1], [2]. In short – $\text{Trip}(u) \in \text{Truth}(u) \neq \phi$ which means that $\text{RandomT}(u)$ is uniformly distributed. Since $u$.rand_sum includes $\text{RandomT}(u)$ in the sum, $u$.rand-sum mod $n$ is uniformly distributed as well, which means that every value between 0 to $n-1$ has an equal chance to be selected as an index to $u$.ids_array.

Lemma 2.3 Given a coalition of cheaters denoted C, and a node $u \notin C$, if the leader elected by $u$ is $p \neq u$ then $P(u.\text{consensus} = 1) = P(u.\text{consensus} = 0) = \frac{1}{2}$

Proof. From the algorithm, after line 22 we have:

$u$.input_sum = $\text{InputT}(u) + \text{InputL}(u) + \text{Trip}(u.\text{leader},u).\text{input}$

Since we have assumed a uniform distribution over the inputs, the probability that $\text{inputT}(u)$ is odd is equal to the probability that it is even, i.e. $\frac{1}{2}$. This implies that $\text{input_sum}$ has the same probability to be odd and even, regardless of the parity of $\text{InputL}(u) + \text{Trip}(u.\text{leader},u)$. This proves Lemma 2.3.

Lemma 2.4 Assume a coalition of cheaters denoted C, and a node $u \notin C$, where $u$.input_interface \in C. If the leader elected by $u$ is $u$, then C has full control over the decision of $u$.

Proof. Since $u$.input_interface is a cheater, $\text{InputT}(u) = u$.input. After line 22:

$u$.input_sum = $u$.input + $\text{InputL}(u) + \text{Trip}(u.\text{leader},u).\text{input}$ = $2 \cdot u$.input + $\text{InputL}(u)$

Hence the result of the consensus algorithm is:

$u.\text{consensus} = u$.input_sum mod 2 = $(2 \cdot u$.input + $\text{InputL}(u))$ mod 2 = $\text{InputL}(u)$ mod 2

Which means that $u$'s decision depends only on the values sent to $u$ from $u$.input_interface.

Lemma 2.5 Assume a coalition of cheaters denoted C, with $|C| = n-2$. Also, assume $u_1, u_2 \notin C$ aren't neighbors in the ring. In this case, C has no incentive to cheat.

Proof. Assume w.l.o.g, that $C$ prefers the consensus to be 1. For convenience, we shall divide our analysis to the following three cases:

Case 1: $u_1, u_2$ received different random or id values during the course of the algorithm. Then we can analyze this case as if $u_1$ and $u_2$ were on different rings, thus their decisions are independent.

By 2.2:

1. $P(u_1.\text{leader} = u_1) = \frac{1}{n}, P(u_1.\text{leader} \neq u_1) = \frac{n-1}{n}$

It follows from 2.3 and 2.4 that:

2. $P(u_1.\text{consensus} = 1 | u_1.\text{leader} \neq u_1) = \frac{1}{2}$

3. $P(u_1.\text{consensus} = 1 | u_1.\text{leader} = u_1) = 1$

Combining (1), (2) and (3):
\[
\mathbb{P}(u_1, \text{consensus} = 1) = \frac{1}{n} + \frac{n-1}{2n}
\]

The exact same result applies to \(u_2\). As we explained, in this case \(u_1\), \text{consensus} is independent of \(u_2\), \text{consensus}, therefore the probability for a consensus to be reached over 1 is:

\[
\mathbb{P}(u_1, \text{consensus} = 1 \land u_2, \text{consensus} = 1) = \left(\frac{1}{n} + \frac{n-1}{2n}\right)^2 = \frac{1}{4} \left(1 + \frac{1}{n}\right)^2 \leq \frac{1}{4} \left(1 + \frac{1}{4}\right)^2 < \frac{1}{2}
\]

Where \(\leq\) is a result of \(n \geq 4\) (there are 2 non cheaters and an even non-zero number of coalition members). Therefore \(C\) has no incentive to cheat in Case 1.

**Case 2:** The random and id values received by \(u_1\) and \(u_2\) during the course of the algorithm are identical, but the input values are different. In this case, the leader elected by \(u_1\) is the same as the one elected by \(u_2\). For convenience, we shall denote it as \textit{leader}, since both nodes choose the same leader in this case.

By Lemma 2.2:

\[
\mathbb{P}(\text{leader} \neq u_1, u_2) = \frac{n-2}{n}, \quad \mathbb{P}(\text{leader} = u_1) = \frac{1}{n}, \quad \mathbb{P}(\text{leader} = u_2) = \frac{1}{n}
\]

From Lemmas 2.3 and 2.4 we have:

\[
\mathbb{P}(u_1, \text{consensus} = u_2, \text{consensus} = 1 \mid \text{leader} \neq u_1, u_2) = \frac{1}{4}
\]

\[
\mathbb{P}(u_1, \text{consensus} = u_2, \text{consensus} = 1 \mid \text{leader} = u_1) = \frac{1}{2}
\]

\[
\mathbb{P}(u_1, \text{consensus} = u_2, \text{consensus} = 1 \mid \text{leader} = u_2) = \frac{1}{2}
\]

By combining (6) – (9), we can derive:

\[
\mathbb{P}(u_1, \text{consensus} = 1 \land u_2, \text{consensus} = 1) = \frac{1}{4} + \frac{n-2}{n} + \frac{1}{2} + \frac{1}{2} + \frac{1}{n} = \frac{1}{4} + \frac{1}{2} + \frac{1}{n} \leq \frac{1}{4} + \frac{1}{8} < \frac{1}{2}
\]

Thus \(C\) has no incentive to cheat in Case 2 as well.

**Case 3:** Nodes \(u_1\) and \(u_2\) receive the exact same triplets during the course of the algorithm.

In this case at least one of the nodes will not choose itself as the leader. By Lemma 2.3 the probability of this node deciding on 1 is \(\frac{1}{2}\), and this bounds the probability of a consensus over this value. Hence, \(C\) has no incentive to cheat in Case 3.

By cases 1 – 3, Lemma 2.5 is proved.

**Lemma 2.6** Assume a coalition of cheaters denoted \(C\), and two nodes \(u_1, u_2 \notin C\) such that \(u_1 = u_2, \text{input}_\text{interface}\). Then \(\mathbb{P}(u_2, \text{consensus} = 1) = \mathbb{P}(u_2, \text{consensus} = 0) = \frac{1}{2}\).

**Proof.** By Lemma 2.3, if \(u_2\) chooses a leader other than itself, then:

\[
\mathbb{P}(u_2, \text{consensus} = 1) = \mathbb{P}(u_2, \text{consensus} = 0) = \frac{1}{2}
\]

If \(u_2\) chooses itself as a leader, then \(u_2, \text{input}_\text{sum} = 2 \cdot u_2, \text{input} + u_1, \text{input} + \text{InputL}(u)\)
Since $\mathbb{P}(u_1.\text{input} = 1) = \mathbb{P}(u_2.\text{input} = 1) = \frac{1}{2}$, from the exact same arguments of Lemma 2.3 we have $\mathbb{P}(u_2.\text{consensus} = 1) = \mathbb{P}(u_2.\text{consensus} = 0) = \frac{1}{2}$.

**Lemma 2.7** Assume a coalition of cheaters denoted $C$ with $|C| = n - 2$, and two nodes $u_1, u_2 \in C$ such that $u_1 = u_2.\text{input}_{\text{interface}}$. Then there is no incentive for $C$ to cheat.

**Proof.** Follows from Lemma 2.6.

**Proof of Theorem 2** (A $n - 2$-Strong-Equilibrium Algorithm)

**Proof.** From Lemma 2.1 we know that the algorithm maintains both validity and agreement when every node complies, and also that $\mathbb{P}(\text{ring decides 0}) = \mathbb{P}(\text{ring decides 1}) = \frac{1}{2}$. From Lemmas 2.5 and 2.7 we know that regardless of the position of the two non-cheaters (neighbors or not) there is no incentive for cheaters to cheat. Thus, the algorithm provides a $n - 2$-strong-equilibrium.

V. CONCLUSION

In this paper, we have proved that problem of distributed consensus with a coalition of $n - 1$ rational agents cannot be solved when $n$ is even, regardless of the network topology. Furthermore, we have proved that when $n$ is odd, there can be only one method to reach consensus – the parity of the sum of all inputs.

Also, we have proved that the algorithm from [1] solves the problem of distributed consensus in a synchronous ring with a coalition of $n - 2$ rational agents, when $n$ is even. Thus, we gave both an upper and a lower bound on the maximal size of a coalition supported by a correct protocol for the problem of distributed consensus with rational agents in this topology.

VI. ACKNOWLEDGEMENT

We would like to express our gratitude to Prof. Yehuda Afek, for his support, patience, and encouragement throughout our time as his students. We also thank Rhea Chowers for his comments on an earlier version of this paper.

**REFERENCES**

[1] Y. Afek, Y. Ginzeberg, S. Landau Feibish, and M. Sulamy. “Distributed Computing Building Blocks for Rational Agents”. Available: http://mcg.cs.tau.ac.il/papers/podc091f-afek.pdf.

[2] I. Abraham, D. Dolev, and J. Y. Halpern. “Distributed protocols for leader election: A game-theoretic perspective”. In DISC, pages 61–75, 2013.

[3] Davies, and Wakerly. "Synchronization and Matching in Redundant Systems." IEEE Transactions on Computers C-27.6 (1978): 531-39.

[4] Pease, M., R. Shostak, and L. Lamport. "Reaching Agreement in the Presence of Faults." Journal of the ACM 27.2 (1980): 228-34.

[5] Dolev, Danny, and H. Raymond Strong. "Polynomial algorithms for multiple processor agreement." Proceedings of the fourteenth annual ACM symposium on Theory of computing - STOC '82 (1982).

[6] Fischer, Michael J. "The consensus problem in unreliable distributed systems (a brief survey)." Foundations of Computation Theory Lecture Notes in Computer Science (1983): 127-40.

[7] Fischer, Michael J., Nancy A. Lynch, and Michael S. Paterson. "Impossibility of distributed consensus with one faulty process." Proceedings of the 2nd ACM SIGACT-SIGMOD symposium on Principles of database systems - PODS 83 (1983).

[8] I. Abraham, L. Alvisi, and J. Y. Halpern. Distributed computing meets game theory: combining insights from two fields. SIGACT News, 42(2):69–76, 2011.

[9] I. Abraham, D. Dolev, and J. Y. Halpern. Distributed protocols for leader election: A game-theoretic perspective. In DISC, pages 61–75, 2013.

[10] I. Abraham, D. Dolev, R. Gonen, and J. Y. Halpern. Distributed computing meets game theory: robust mechanisms for rational secret sharing and multiparty computation. In PODC, pages 53–62, 2006.

[11] A. S. Aiyer, L. Alvisi, A. Clement, M. Dahlin, J.-P. Martin, and C. Porth. Bar fault tolerance for cooperative services. In SOSP, pages 45–58, 2005.

[12] J. Y. Halpern and V. Teague. Rational secret sharing and multiparty computation: extended abstract. In STOC, pages 623–632, 2004.