Title	Tight Bound on Mobile Byzantine Agreement
Author(s)	Bonnet, Francois; Defago, Xavier; Nguyen, Thanh Dang; Potop-Butucaru, Maria
Citation	Lecture Notes in Computer Science, 8784: 76-90
Issue Date	2014
Type	Journal Article
Text version	author
URL	http://hdl.handle.net/10119/12944
Rights	This is the author-created version of Springer, Francois Bonnet, Xavier Defago, Thanh Dang Nguyen, Maria Potop-Butucaru, Lecture Notes in Computer Science, 8784, 2014, 76-90. The original publication is available at www.springerlink.com, http://dx.doi.org/10.1007/978-3-662-45174-8_6
Description	

Tight Bound on Mobile Byzantine Agreement

François Bonnet⋆, Xavier Défago⋆, Thanh Dang Nguyen⋆, and Maria Potop-Butucaru†

⋆School of Information Science, JAIST, Japan
†Université Pierre & Marie Curie (UPMC) – Paris 6, France

Abstract. This paper investigates the problem of Byzantine Agreement in a synchronous system where malicious agents can move from process to process, corrupting their host. Earlier works on the problem are based on biased models which, as we argue in the paper, give an unfair advantage either to the correct processes or to the adversary controlling the malicious agents. Indeed, the earlier studies of the problem assume that, after a malicious agent has left a process, that process, said to be cured, is able to instantly and accurately detect the fact that it was corrupted in earlier rounds, and thus can take local actions to recover a valid state (Garay's model). We found no justification for that assumption which clearly favors correct processes. Under that model, an algorithm is known for $n > 4t$, where n is the number of processes and t the maximum number of malicious agents. The tightness of the bound is unknown. In contrast, more recent work on the problem remove the assumption on detection and assume instead that a malicious agent may have left corrupted messages in the send queue of a cured process. As a result, the adversary controlling the malicious agents can corrupt the messages sent by cured processes, as well as those sent by the newly corrupted ones, thus doubling the number of effective faults. Under that model, which favors the malicious agents, the problem can be solved if and only if $n > 6t$. In this paper, we refine the latter model to avoid the above biases. While a cured process may send messages (based on a state corrupted by the malicious agent), it will behave correctly in the way it sends those messages: i.e., send messages according to the algorithm. Surprisingly, in this model we could derive a new non-trivial tight bound for Byzantine Agreement. We prove that at least $5t+1$ processors are needed in order to tolerate t mobile Byzantine agents and provide a time optimal algorithm that matches this lower bound, altogether with a formal specification of the problem.

1 Introduction

New emergent distributed systems such as P2P, overlay networks, social networks or clouds are inherently vulnerable to faults, insider attacks, or viruses. Faults and attacks cannot be predicted accurately, may affect different parts of a system, and may occur at any moment of its execution. In this work, we investigate the case where transient state corruptions, which can be abstracted as malicious "agents," can move through the network and corrupt the nodes they occupy.

This models the situation where, as soon as a faulty node is repaired (e.g., by software rejuvenation), another one becomes compromised. For more than two decades, the main case study problem in this context was Byzantine Agreement. Briefly stated, it requires processors, some of which malicious, that start the computation with an initial value to decide on the same value. When faults are mobile the problem is known as Mobile Byzantine Agreement and requires special attention for preserving agreement once it has been reached.

Related work. Byzantine Agreement, introduced by Lamport *et al.* [12, 16], has been studied for decades in static distributed systems under different aspects (e.g., *possibility*, *complexity*, *cost*) in various models (from synchronous [12,16,17] to asynchronous [5,13], from authenticated [8] to anonymous [14]) with different methodologies (deterministic [12, 16], probabilistic [3, 9]). In all these works, faults are stationary. That is, they do not change their original location during the computation.

Santoro *et al.* [19, 20], and later Schmid *et al.* [22], investigate the agreement problem in dynamic transmission failure models for both complete and arbitrary networks. These models assume that different communication links may randomly fail at different times. Santoro and Widmayer [19] study the k-agreement problem, where the system reaches a k-agreement if, in finite time, k processes choose the same value, either 0 or 1, with $k > \lceil n/2 \rceil$,[1] where n is the total number of processes.

Based on the bivalent argument of Fischer *et al.* [10], they state that ($\lceil n/2 + 1 \rceil$)-agreement is impossible in a synchronous system if at each time there is one processor whose messages may be corrupted. Although not explicitly stated, the impossibility applies to the mobile Byzantine model. Thus, work on Mobile Byzantine Agreement typically rely on the assumption that at least one process remains uncorrupted for $\Omega(n)$ rounds of communication.

Mobile Byzantine Agreement, introduced by Reischuk [18], has regained much attention recently. Research on the problem, in synchronous systems, follows two main directions: constrained or unconstrained mobility.

Constrained mobility. This direction, studied by Buhrman *et al.* [4], considers that malicious agents move from one node to another only when protocol messages are sent (similar to how viruses would propagate). In that model, they prove a tight bound for Mobile Byzantine Agreement ($n > 3t$, where t is the maximal number of simultaneously faulty processes) and propose a time optimal protocol that matches this bound.

Unconstrained mobility. In this direction, which includes the work in this paper, the mobility of malicious agents is *not* constrained by message exchanges [1, 11, 15, 18, 21].

Reischuk [18] proposed a first sub-optimal solution under an additional hypothesis on the stability/stationarity of malicious agents for a given period of time. Later, Ostrovsky and Yung [15] introduced the notion of an adversary that can inject and distribute faults in the system at a constant rate in every round

[1] If $k \leq \lceil n/2 \rceil$ the k-agreement problem is trivial.

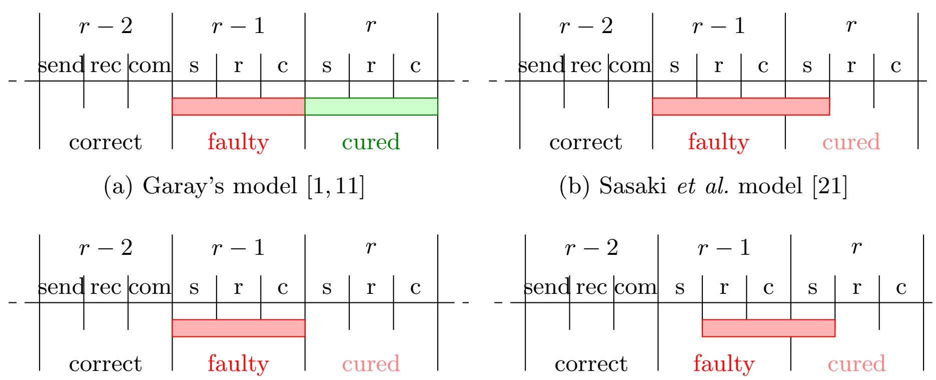
and proposed solutions (mixing randomization and self-stabilization) for tolerating the attacks of mobile viruses. Then, Garay [11] and, more recently, Banu *et al.* [1] and Sasaki *et al.* [21] consider, in their model, that processes execute synchronous rounds composed of three phases: *send*, *receive*, *compute*. Between two consecutive rounds, malicious agents can move from one host to another, hence the set of faulty processes has a bounded size although its membership can change from one round to the next. Garay's model is particular in that, a process has a limited ability to detect its own infection after the fact. More precisely, during the first round following the leave of the malicious agent, a process enters a state, called *cured*, during which it can take preventive actions to avoid sending messages that are based on a corrupted state. Under this assumption, Garay [11] proposes an algorithm that solves Mobile Byzantine Agreement provided that $n > 6t$.

Notice that Garay's model advantages the cured processes since they have the possibility of miraculously detecting the leave of malicious agents. In the same model, Banu *et al.* [1] propose a Mobile Byzantine Agreement algorithm for $n > 4t$. However, to the best of our knowledge, the tightness of the bound remains an open question.

Sasaki *et al.* [21] investigate the problem in a different model where processes do not have this ability to detect when malicious agents move. This is similar to our model with the subtle difference that cured processes have *no control* on the messages they send. That is, messages are computed in the previous round (i.e., when the process was still faulty) and the cured process cannot control the buffer where these messages are stored, even though the process is no longer faulty. It follows that a cured process may behave as a malicious one for one additional round. They propose tight bounds for Mobile Byzantine Agreement in arbitrary networks if $n > 6t$ and the degree of the network is $d > 4t$. This work extends the tight bounds ($n > 3t$ and $d > 2t$) for Byzantine Agreement of Dolev [7] in arbitrary networks with static faults.

Motivation. Analyzing the results proposed in [1, 11, 21], it is clear that there is a gap between how these models capture the power of malicious agents or cured processes. Garay's model [11] is biased toward the cured processes, whereas the model of Sasaki *et al.* [21] favors the malicious agent, as it can control the send buffer of a cured process even though it is no longer hosted by the process. Our research fills the gap by avoiding these biases; similarly to Sasaki's model [21], a cured process may send corrupted messages, but only computed based on the corrupted state left by a malicious agent. In particular, a malicious agent can corrupt neither the code nor the identity of the process it occupies, and a cured process always executes a correct code which ensures, for instance, that it will send the same message to all of its neighbors.

The difference between the three models are subtle (see Fig. 1) but they have important consequences (Table 1). Figure 1 depicts the effects of a malicious agent on a process. Red areas correspond to the steps controlled by the malicious agent. In Sasaki's model [21] (Fig. 1b), a single malicious agent can corrupt a process for two rounds even though it occupies the process only for a single round. In Garay's model [11] (Fig. 1a) a cured process is aware of its current state (cured), which is represented in green. In our model (Fig 1c; defined in Sect. 2) malicious nodes have the same power as in Garay's model, but the cured processes may send messages with corrupted content as in Sasaki's model.



(a) Garay's model [1, 11]

(b) Sasaki *et al.* model [21]

(c) Our model

(d) Buhrman *et al.* model [4]

Fig. 1. Graphical representation of the various fault models

Table 1. Lower and upper bounds for Byzantine Agreement with mobile faults.

Model	Impossibility result	Possibility result	Byzantine vs Cured Game
Garay [11]	open question	$n > 6t$	Advantaged Cured
Banu *et al.* [1]	open question	$n > 4t$	Advantaged Cured
Sasaki *et al.* [21]	$n \leq 6t$	$n > 6t$	Advantaged Byzantine Agent
This paper	$n \leq 5t$	$n > 5t$	No one advantaged
Buhrman *et al.* [4]	$n \leq 3t$	$n > 3t$	Virus like propagation

Contribution. In this model we prove a tight bound for the agreement problem. We prove in Section 3 that the problem has no solution if the size of the network is $n < 5t$ (where t is an upper bound on the number of faulty agents) and propose an algorithm that matches this bound in Section 4. We also formalize the Mobile Byzantine Agreement problem in Section 2.2. Following the results proved in [11], our solution is also asymptotically time optimal.

2 Model and definitions

2.1 System model

Processes. We consider a synchronous message-passing system consisting of n processes p_0, p_1, ..., p_{n-1} where $\Pi = \{0, \ldots, n-1\}$ denotes the set of process indices. Each process is an automaton whose state evolves following the execution of its local algorithm. All processes execute the same algorithm.

The network is fully connected: all pairs of processes are directly linked with a reliable bidirectional channel; *i.e.* there is no loss, duplication, or alteration of messages. The system evolves in synchronous rounds and all processes start simultaneously at round 0. There is a round counter accessible to the algorithm executed by each process. Each round consists of three steps; *send*, *receive*, and *compute*. Based on its current local state, a process (1) computes and *sends* a message to all processes (including itself); (2) *receives* messages sent by all processes (including itself); and (3) *computes* its new state based on its current state and the set of received message.

Mobile malicious agents. Faults are represented by malicious mobile agents that can move from process to process between rounds. There are at most t malicious agents, with $t < n$, and any process can be occupied by an agent. A process is said to be *faulty* in a given round if it is occupied by an agent in that round. A process which is not occupied by a malicious agent, but was occupied in the previous round is called a *cured* process. A process which is neither faulty nor cured is called a *correct* process. $\mathcal{F}_r$, $\mathcal{C}o_r$, and $\mathcal{C}u_r$ denote respectively the set of faulty, correct, and cured processes at round r. For ease of writing, we also consider the combined sets of correct/cured processes as the set of non-faulty processes $\mathcal{C}_r = \mathcal{C}o_r \cup \mathcal{C}u_r = \Pi \setminus \mathcal{F}_r$.

Malicious agents are mobile and can move between the compute step of a round and the send step of the next round (Figure 1c). The behavior of a faulty process is controlled by the malicious agent. In particular, the agent can corrupt the local state of its host process, and force it to send arbitrary messages (potentially different messages to different processes). However, a malicious agent cannot corrupt the identify of that process (*i.e.*, it cannot send messages using another identity), and is unable to modify the code of the algorithm (*i.e.*, the process resumes executing the correct algorithm after the malicious agent moves away). So, as suggested in [4], we assume a secure, tamper-proof read-only memory where the identity and the code are stored.

While it is possible for each non-faulty process to rejuvenate its code at the beginning of each round, local variables may still be corrupted (and of course cannot be recovered). Therefore, in the case of cured processes the computation may be performed using a corrupted state.

Comparison with previous models. As explained in Section 1 and graphically depicted in Figure 1, the above model differs from Garay's [11] and Sasaki's [21] as follows. In Sasaki's model [21], a single malicious agent can corrupt a process for more than a round although occupying this process only for a round. In our model, once the malicious agent leaves a process, that process will execute the correct code even though the computation will be performed on a corrupted state. Differently from the Garay's model [11], where a cured process has the knowledge of its cured state and exploits it in the algorithm, in our model processes can not access and exploit this knowledge.

Notation. In the formal definitions and proofs, var_i^r denotes the value of variable var in process p_i at the end of round r. We also use the notation $\#_w(\mathcal{W})$ to refer to the number of occurrences of w in tuple $\mathcal{W}$.

2.2 Mobile Byzantine Agreement problem

We now formally define the Mobile Byzantine Agreement problem introduced first by Garay *et al.* [11] and refined most recently by Sasaki *et al.* [21]. The definition presented here is stronger than the definition proposed by Sasaki [21] (see discussion below).

Each initially-correct process p_i has an initial value w_i. All processes must *decide*[2] a value *dec* such that the following properties hold:

1. *BA-Termination*: Eventually, all non-faulty processes during a round terminate the round with a non-bottom decided value.

$$\exists r, \quad \forall r' > r \quad \forall i \in \mathcal{C}_{r'} \quad dec_i^{r'} \neq \bot$$

2. *BA-Agreement*: No two non-faulty processes decide different values:

$$\forall r, r' \quad \forall i \in \mathcal{C}_r \quad \forall j \in \mathcal{C}_{r'} \quad \left(dec_i^r \neq \bot \ \wedge \ dec_j^{r'} \neq \bot\right) \Rightarrow \left(dec_i^r = dec_j^{r'}\right)$$

3. *BA-Validity*: If all initially-correct processes propose the same value w, correct processes can decide only w.

$$\forall w \quad (\forall i \in \mathcal{C}o_0 \quad w_i = w) \Rightarrow (\forall r \quad \forall i \in \mathcal{C}_r \quad dec_i^r \in \{\bot, w\})$$

Note that specification of Mobile Byzantine Agreement given in this section is actually stronger than the definition proposed by Sasaki *et al.* [21]. They differ in two important aspects. Firstly, where we require that, after some time, all non-faulty processes decide a value at every round, their definition requires a decision only from processes that are not faulty infinitely often. Secondly, where we allow non-faulty processes to decide only on a unique non-bottom value, Sasaki's algorithm [21] allows the variable storing the decision to take arbitrary values for a finite number of rounds. In other words, our specification requires *perpetual* consistency whereas Sasaki's algorithm ensures *eventually* consistency.

[2] We use a terminology consistent with the classical definition of Byzantine agreement. However, the action "decide" does not in itself guarantee a permanent decision. Indeed, due to the mobility of the malicious agents, non-faulty processes must *re-decide* the decision at the end of each round.

We now state two lemmas, proved in earlier models [11,19], which also apply to our model. The first lemma states a necessary condition. That condition is however not sufficient; as explained previously, a bound on the number of faults is also required.

Lemma 1 (stated in [11]; formal proof derivable from [19]). *Mobile Byzantine Agreement requires that at least one process remains uncorrupted for $\Omega(n)$ rounds of communication.*

Lemma 2 (from [11]). *Every Mobile Byzantine Agreement protocol requires $\Omega(n)$ rounds in its worst case execution.*

3 Upper bound on the number of faulty processes

In this section, we prove that, in the presence of t malicious mobile agents, Mobile Byzantine Agreement cannot be solved with $5t$ processes or less, even if some process remains uncorrupted forever.

Sasaki *et al.* [21] proved a similar result by reduction from a well-known existing bound. From the classical bound ($n \leq 3t$) on synchronous Byzantine agreement, they could obtain their bound ($n \leq 6t$) by considering both faulty and cured processes as Byzantine.

However, we cannot use the same approach because, in sharp contrast with Sasaki's model [21] and as explained in Section 2, in our model, the adversary cannot entirely control cured processes.

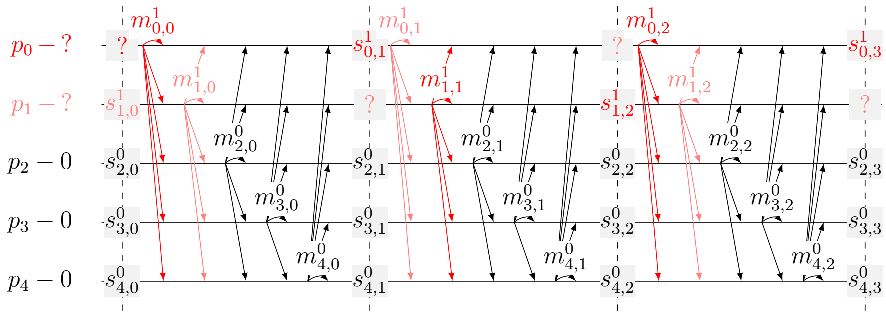
Theorem 1. *There is no deterministic algorithm that solves Mobile Byzantine Agreement in a synchronous five-process system in the presence of a single mobile Byzantine agent (even with a permanently correct process).*

Proof. The proof is by contradiction. Given a system consisting of five processes $\{p_0, \ldots, p_4\}$, where at least one is permanently correct, let us suppose that there exists an algorithm that can solve the BA problem in the presence of a single malicious mobile agent. Suppose that, in this algorithm, processes send the same message to all processes.[3] Note that, during an execution, nothing prevents a faulty processes from sending different messages to other processes.

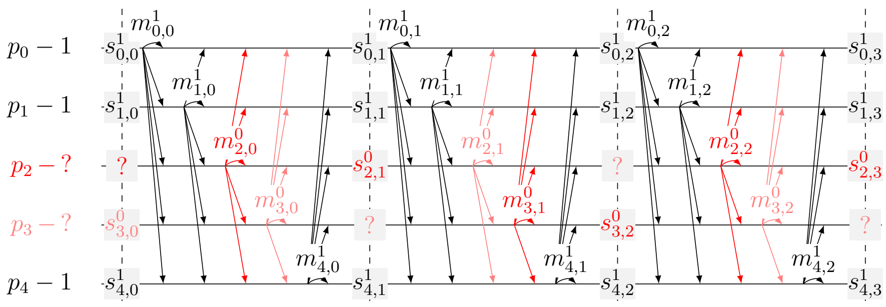
General idea. We consider three executions of this algorithm. In executions E^0 and E^1, all correct processes propose the same value; 0 and 1 respectively. The BA properties imply that, eventually, non-faulty processes respectively decide 0 and 1 in these two executions. The third execution, called E^{01}, brings a contradiction: some processes decide 0 while others decide 1.

The three executions are represented on Figure 2. Red (resp. light red) arrows correspond to corrupt messages sent by faulty (resp. cured) processes. The values

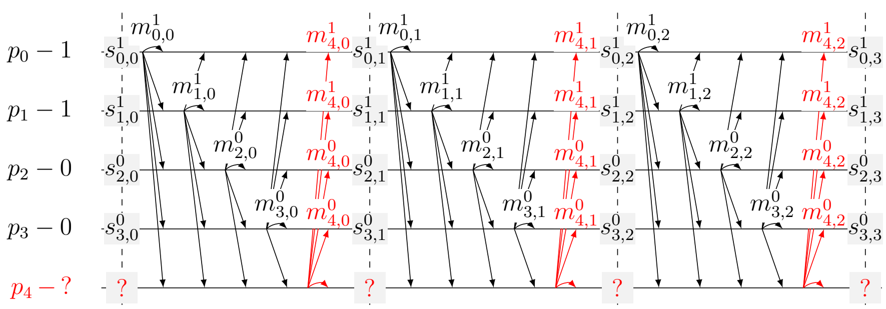
[3] If not the case, we can trivially define an algorithm that satisfies this property by combining the set of sent messages into a single message.



(a) Execution E^0 where initially-correct processes p_2, p_3, and p_4 propose value 0.



(b) Execution E^1 where initially-correct processes p_0, p_1, and p_4 propose value 1.



(c) Execution E^{01} where initially-correct processes p_0 and p_1 propose value 1 while initially-correct processes p_2 and p_3 propose value 0. Process p_4 is faulty and sends different messages to each process.

Fig. 2. Three executions leading to a contradiction of the existence of a BA protocol in a 5-process system with one mobile malicious agent. (Legend: Arrows correspond to messages exchanged between processes. Gray boxes contain the new local state computed by each process at the end of each round, which is then used to send message in the following round. Red indicates actions taken by the faulty processes while light red refers to actions taken by cured processes. Vertical dashed line separate successive rounds.)

proposed by correct processes appear on the left. Non-correct processes do not have proposed values since they may have been corrupted by the malicious agent. Vertical dashed lines separate successive rounds.

For each execution, we choose the process occupied by the single malicious agent. As required, there is at least one process which is permanently non-faulty in each execution.

Executions E^0 and E^1. In execution E^0, the malicious agent alternates between processes p_0 and p_1. In execution E^1, it alternates between processes p_2 and p_3. Processes p_2, p_3, and p_4 are initially correct and propose 0 in E^0, while processes p_0, p_1, and p_4 are initially correct and propose 1 in E^1.

For non-faulty processes, the messages sent during these executions are computed by the algorithm based on the local states of processes. For correct processes (*i.e.*, excluding cured ones), let us denote by $s^0_{i,r}$ (resp. $s^1_{i,r}$) the local state of process p_i at the beginning of the round r in execution E^0 (resp. E^1). Based on this local state, let $m^0_{i,r}$ (resp., $m^1_{i,r}$) denote the message computed and sent by a correct process p_i at round r in execution E^0 (resp., E^1).

We now define the behavior of the malicious agent. For the faulty process p_i (either p_0 or p_1) at round r of execution E^0, we choose that p_i sends the message $m^1_{i,r}$ (*i.e.*, the message it would have sent at the same round in E^1) and we choose that p_i updates its local state to $s^1_{i,r+1}$ at the end of the round (*i.e.*, the same state it would have computed in E^1). Similarly we choose that the faulty process p_i (either p_2 or p_3) at round r of execution E^1 sends the message $m^0_{i,r}$ and updates its state to $s^0_{i,r+1}$.

Execution E^{01}. In execution E^{01}, the malicious agent always occupies process p_4. The four other processes are initially (and forever) correct. As in E^0, processes p_2 and p_3 propose 0. As in E^1, processes p_0 and p_1 propose 1. In this execution, the faulty process p_4 does not send the same message to all processes. At any round, p_4 sends the message $m^1_{4,r}$ to p_0 and p_1, but sends $m^0_{4,r}$ to p_3 and p_4.

Indistinguishability. In the sequel, we prove the following claim: *E^0 and E^{01} are indistinguishable for p_2 and p_3, and similarly E^1 and E^{01} for p_0 and p_1.* This can be proven by induction on the round number, using the following predicate $\mathcal{P}(r)$ for $r \geq 0$:

$$\mathcal{P}(r) = \begin{cases} p_0 \text{ starts round } r \text{ in } E^1 \text{ and } E^{01} \text{ with the same local state} \\ p_1 \text{ starts round } r \text{ in } E^1 \text{ and } E^{01} \text{ with the same local state} \\ p_2 \text{ starts round } r \text{ in } E^0 \text{ and } E^{01} \text{ with the same local state} \\ p_3 \text{ starts round } r \text{ in } E^0 \text{ and } E^{01} \text{ with the same local state} \end{cases}$$

The proof is only for p_0. The proofs for p_1, p_2, and p_3 are identical.

- *Case $r = 0$.* p_0 proposes the same value in E^1 and E^{01} and therefore starts round 0 with the same initial local state, namely $s^0_{0,0}$.
- *Case $r \geq 0$.* Let us suppose that predicate $\mathcal{P}(r)$ is true.

- p_0 is correct in E^1 and E^{01} and, by induction hypothesis, starts round r with the same local state. Therefore p_0 necessarily sends the same message, namely $m^1_{0,r}$, to all processes in round r of both E^1 and E^{01}.
  Similarly, p_1 sends the same message $m^1_{1,r}$ to all processes in round r of both E^1 and E^{01}.
- p_2 is correct in E^0 and E^{01} and, by induction hypothesis, starts round r with the same local state. Therefore p_2 necessarily sends the same message, namely $m^0_{2,r}$, to all processes in round r of both E^0 and E^{01}. Considering execution E^1, there are two cases to consider; (1) p_2 is faulty during round r and then, by construction, the malicious agent forces p_2 to send the message $m^0_{2,r}$; (2) p_2 is cured during round r, which means that it was faulty in the previous round and the malicious agent forced p_2 to start round r in the local state $s^0_{2,r}$ which implies that p_2 still sends the message $m^0_{2,r}$. In all cases, p_2 sends the same message in round r of both E^1 and E^{01}.
  Similarly, p_3 sends the same message $m^0_{3,r}$ to all processes in round r of both E^1 and E^{01}.
- p_4 is faulty in E^{01}. By construction, in each round, it sends to p_0 the same message as in E^1. It means that p_4 sends the same message, namely $m^0_{4,r}$, to p_0 in round r of both E^1 and E^{01}.

Process p_0 receives the same messages from all processes in round r of E^1 and E^{01}. Since p_0 is correct in both executions, it computes the same new local state and starts round $r+1$, which prove $\mathcal{P}(r+1)$.

Thus by induction, the predicate $\mathcal{P}(r)$ is true for all rounds and therefore the claim holds. Since p_0 and p_1 eventually decide 1 in E^1, they also decide 1 in E^{01}. Similarly, since p_2 and p_3 eventually decide 0 in E^0, they also decide 0 in E^{01}. Contradiction.

When $n \leq 5t$, the proof of Theorem 1 can be generalized by replacing any process appearing in the proof by a group of processes of size at most t.

Corollary 1. *There is no deterministic algorithm that solves the Mobile Byzantine Agreement problem in a synchronous n-process system in the presence of t mobile byzantine agent if $n \leq 5t$ (even with a permanently correct process).*

4 Algorithm for Mobile Byzantine Agreement

Given a system with t malicious mobile agents, we introduce an algorithm that solves Mobile Byzantine Agreement under the following two conditions: (1) there are at least $5t+1$ processes in total, and (2) at least one process remains uncorrupted for $3n$ consecutive rounds (see Lemma 1).

Algorithm 1: BA algorithm (code for p_i with proposed value w_i)

```
Function MBA(w_i):
    v_i ← w_i;
    for s = 0 to n − 1 do
        begin round // proposing round r = 3s
            v_i ← propose(v_i);
            dec_i ← ⊥;
        end round

        begin round // collecting round r = 3s + 1
            SV_i ← collect(v_i);
            dec_i ← ⊥;
        end round

        begin round // deciding round r = 3s + 2
            v_i ← decide(s, SV_i);
            dec_i ← ⊥;
        end round
    end for

    dec_i ← v_i;

    for r = 3n to ∞ do
        begin round // maintaining round
            send dec_i to all processes;
            dec_i ← the value received at least n − 2t times;
        end round
    end for

Function propose(v):
    PV[1..n] ← [⊥, ..., ⊥];
    send v to all processes;
    foreach j ∈ Π do
        if v_j received from j then PV[j] ← v_j;
    if ∃w ≠ ⊥, #_w(PV) ≥ n − 2t then return w;
    return ⊥;

Function collect(v):
    SV[1..n] ← [⊥, ..., ⊥];
    send v to all processes;
    foreach j ∈ Π do
        if v_j received from j then SV[j] ← v_j;
    return SV;

Function decide(s, SV):
    EV[1..n][1..n] ← [[⊥, ..., ⊥], ..., [⊥, ..., ⊥]];
    send SV to all processes;
    foreach j ∈ Π do
        if SV_j received from j then EV[j] ← SV_j;

    RV[1..n] ← [⊥, ..., ⊥];
    foreach j ∈ Π do
        if ∃w ≠ ⊥, #_w(EV[·][j]) > 2t then RV[j] ← w;

    if ∃w ≠ ⊥, #_w(RV) > 3t then return w;
    else
        c ← s mod n;
        if ∃w ≠ ⊥, #_w(EV[c][·]) > 2t then return w;
        return 0;
```

4.1 Description of the algorithm

The algorithm builds upon earlier ones [1, 11, 21] but contains some important improvements; (i) a clear separation between the deciding and the maintaining parts, (ii) a simplification of the code of the algorithm, and (iii) additional code in order to satisfy our stricter BA-Agreement property. The algorithm (lines $1-23$) consists of two main parts:

1. Deciding part: processes execute $3n$ rounds to agree on a value.
2. Maintaining part: processes execute the same round forever to keep the decided value.

Maintaining part (lines $18-23$) This part is simple and repeats forever from round $3n$. The goal is to allow cured processes to recover the decided value from correct ones, since that value may have been corrupted by the malicious agent. All processes exchange their current decided values *dec* and update their variable *dec* to the value that has been received at least $n-2t$ times. During each of these rounds, there must be at least $n-2t$ correct processes according to the model. If all of them send the same value (which is guaranteed by the algorithm), all non-faulty processes receive $n-2t$ messages containing this same value and thus decide accordingly.

Deciding part (lines $3-16$) This part is complex and consists of n phases of 3 rounds each. The goal is to guarantee that, at the end of round $3n-1$, all non-faulty processes have the same value v and therefore decide it (line 17). During the first $3n$ rounds, v may take different non-bottom values, which is why processes cannot decide in earlier rounds.[4]

This part uses the rotating coordinator paradigm. Recall that, in each round, there are at least $n-t$ non-faulty processes, and at least $n-2t$ correct ones. Each of the n phases are divided into 3 rounds:

- Proposing round; all non-faulty processes (at least $n-t$) end the round with at most one non-bottom value v. Consequently, it guarantees that the (at least $n-2t$) correct processes of the next round start with at most one non-bottom value v.
- Collecting round; processes exchange the values computed in the previous round and store them in array SV (the set of received values).
- Deciding round; processes try to agree on the same value v using the rotating coordinator paradigm. If the coordinator of the current round is correct during the entire phase, non-faulty processes are guaranteed to terminate the phase with the same value. Such a coordinating round exists since, by assumption, there is one process which is correct for at least $3n$ rounds.
  In the deciding round, processes exchange the array SV computed during the previous round. Based on the arrays they received, each process computes a new[5] array RV (the vector of reconstructed values). For each non-faulty process, both SV and RV contain "almost" the same values ($SV = RV$ if all processes are correct), but, as it appears in the proof, these two arrays are necessary to guarantee the correctness of our algorithm.

[4] This is different from previous papers as already mentioned in Section 2.
[5] Technically, as in [21], it is possible to use the same variable for both SV and RV. We choose to use two different names for the clarity of the proof.

After the phase corresponding to a correct coordinator, all non-faulty processes have the same value v. This property will continue during all subsequent phases even if the corresponding coordinators are faulty (in fact lines $46-49$ will not be executed anymore as shown in the proof).

Additional code (lines 6*,* 10*,* 14*)* Usually, the variable *dec* is initialized to $\perp$ at the beginning of an algorithm. However, this value may be corrupted for any process that becomes faulty during the execution. To satisfy the BA-Agreement property, it is therefore necessary for each non-faulty process to re-initialize its variable *dec* to $\perp$ at the end of each of the first $3n$ round.

4.2 Proof of the algorithm

Due to page limitations, the proof of the algorithm appears in [2]. We only state here the final theorem.

Theorem 2. *Algorithm 1 solves Mobile Byzantine Agreement in a synchronous n-process system in the presence of t mobile Byzantine agents provided that $n \geq 5t+1$ and that at least one process remains uncorrupted.*

5 Conclusion and Discussion

We proposed a new model for Mobile Byzantine Agreement, that balances the power of correct and malicious agents. In our model, a process cannot detect its own infection and cannot instantly recover its state after the malicious agent moves away. Hence, our model gives less power to correct processes than Garay's model [11]. Recall that, in this model, a cured process can magically detect the leave of the malicious agent. In contrast, in our model, a cured process (a process that has been infected by a malicious agent) will not behave maliciously after the agent left it. That is, a cured process may send corrupted messages (computed based on a corrupted state) but it will send the same corrupted message to all neighbors. In this respect, our model gives less power to the Byzantine agents than Sasaki's model [21] where a Byzantine agent can prepare messages and control the sending of these messages even after it left that process. In our model, we prove that there is no protocol for Mobile Byzantine Agreement in synchronous networks with $n \leq 5t$. We propose then a tight algorithm which can tolerate t mobile Byzantine agents with at least $5t+1$ processes.

In the following, we list several open questions and non trivial research directions in this area. The next step in our research is the study on the feasibility of Mobile Byzantine Agreement on arbitrary topologies. Another interesting direction would be to decrease, via randomization, the time complexity of the algorithm.

Notice that, even though our model has a self-stabilization flavor, our work is different in several aspects from the self-stabilizing Byzantine agreement of [6]. Note that in the case of self-stabilizing Byzantine agreement the studied model assumes that the Byzantine set is fixed. That is, it does not change during the execution. Also it is assumed, as in all self-stabilizing algorithms, that the system eventually becomes coherent (i.e. the communication network and a sufficient fraction of nodes is not faulty for sufficient long time period for the pre-conditions for convergence of the protocol to hold). More specifically, in self-stabilization it is assumed that during the convergence period the system does not suffer additional perturbations. In our case the system is permanently stressed due to the mobility of the Byzantine nodes. Note also that the problem solved in [6] is different since it allows the output of inconsistent decision values during transient periods.

In our model, a malicious agent can move anywhere in the network, and likely most work on the subject, we considered a fully connected topology. Sasaki *et al.* [21] have considered the case of different topologies. An interesting line of work is to generalize to arbitrary topologies, and also to consider when the mobility of the malicious agents is constrained by a, possibly different, topology.

Finally, to the best of our knowledge, so far no investigation of Mobile Byzantine Agreement has been done in anonymous settings or networks where node identities are not unique. In these contexts, algorithms based on a coordinator are not applicable.

Acknowledgments

This research was supported in part by JSPS KAKENHI Grant Number 26330020 and 26870228.

References

1. Banu, N., Souissi, S., Izumi, T., Wada, K.: An improved byzantine agreement algorithm for synchronous systems with mobile faults. International Journal of Computer Applications 43(22), 1–7 (April 2012)
2. Bonnet, F., Défago, X., Nguyen, T.D., Potop-Butucaru, M.: Tight bound on mobile byzantine agreement. Research Report IS-RR-2014-004, Japan Advanced Institute of Science and Technology (JAIST) (May 2014)
3. Bracha, G.: An o(log n) expected rounds randomized byzantine generals protocol. Journal of the ACM 34(4), 910–920 (Oct 1987)
4. Buhrman, H., Garay, J.A., Hoepman, J.H.: Optimal resiliency against mobile faults. In: Proceedings of the 25th International Symposium on Fault-Tolerant Computing (FTCS'95). pp. 83–88 (1995)
5. Correia, M., Veronese, G.S., Lung, L.C.: Asynchronous byzantine consensus with $2f + 1$ processes. In: Proceedings of the 25th ACM Symposium on Applied Computing (SAC'10). pp. 475–480 (2010)
6. Daliot, A., Dolev, D.: Self-stabilizing Byzantine agreement. In: Proc. 25th ACM Symp. on Principles of Distributed Computing (PODC'06). pp. 143–152 (2006)
7. Dolev, D.: The byzantine generals strike again. Journal of Algorithms 3(1), 14–30 (March 1982)

8. Dolev, D., Fischer, M.J., Fowler, T.R., Lynch, N.A., Strong, H.R.: An efficient algorithm for byzantine agreement without authentication. Information and Control 52(3), 257–274 (March 1982)
9. Feldman, P., Micali, S.: An optimal probabilistic protocol for synchronous byzantine agreement. SIAM Journal on Computing 26(4), 873–933 (Aug 1997)
10. Fischer, M.J., Lynch, N.A., Paterson, M.S.: Impossibility of distributed consensus with one faulty process. Journal of the ACM 32(2), 374–382 (Apr 1985)
11. Garay, J.A.: Reaching (and maintaining) agreement in the presence of mobile faults. In: Proceedings of the 8th International Workshop on Distributed Algorithms. vol. 857 (1994)
12. Lamport, L., Shostak, R., Pease, M.: The byzantine generals problem. ACM Transactions on Programming Languages and Systems 4(3), 382–401 (Jul 1982)
13. Martin, J.P., Alvisi, L.: Fast byzantine consensus. IEEE Transactions on Dependable and Secure Computing 3(3), 202–215 (Jul 2006)
14. Okun, M., Barak, A.: Efficient algorithms for anonymous byzantine agreement. Theory of Computing Systems 42(2), 222–238 (Jan 2008)
15. Ostrovsky, R., Yung, M.: How to withstand mobile virus attacks (extended abstract). In: Proceedings of the 10th Annual ACM Symposium on Principles of Distributed Computing (PODC'91). pp. 51–59 (1991)
16. Pease, M., Shostak, R., Lamport, L.: Reaching agreement in the presence of faults. Journal of the ACM 27(2), 228–234 (Apr 1980)
17. Raynal, M.: Fault-tolerant Agreement in Synchronous Message-passing Systems. Synthesis Lectures on Distributed Computing Theory, Morgan & Claypool Publishers (2010)
18. Reischuk, R.: A new solution for the byzantine generals problem. Information and Control 64(1-3), 23–42 (January-March 1985)
19. Santoro, N., Widmayer, P.: Time is not a healer. In: Proceedings of the 6th Annual Symposium on Theoretical Aspects of Computer Science (STACS'89). pp. 304–313 (1989)
20. Santoro, N., Widmayer, P.: Majority and unanimity in synchronous networks with ubiquitous dynamic faults. In: Proceedings of the 12th International Conference on Structural Information and Communication Complexity (SIROCCO'05). pp. 262–276 (2005)
21. Sasaki, T., Yamauchi, Y., Kijima, S., Yamashita, M.: Mobile byzantine agreement on arbitrary network. In: Proceedings of the 17th International Conference on Principles of Distributed Systems (OPODIS'13). pp. 236–250 (December 2013)
22. Schmid, U., Weiss, B., Keidar, I.: Impossibility results and lower bounds for consensus under link failures. SIAM Journal on Computing 38(5), 1912–1951 (Jan 2009)