Beyond One Third Byzantine Failures

Cheng Wang*, Carole Delporte-Gallet†, Hugues Fauconnier‡
Rachid Guerraoui§, Anne-Marie Kermarrec¶

Abstract

The Byzantine agreement problem requires a set of n processes to agree on a value sent by a transmitter, despite a subset of b processes behaving in an arbitrary, i.e. Byzantine, manner and sending corrupted messages to all processes in the system. It is well known that the problem has a solution in a (an eventually) synchronous message passing distributed system iff the number of processes in the Byzantine subset is less than one third of the total number of processes, i.e. iff $n > 3b + 1$. The rest of the processes are expected to be correct: they should never deviate from the algorithm assigned to them and send corrupted messages. But what if they still do?

We show in this paper that it is possible to solve Byzantine agreement even if, beyond the b ($< n/3$) Byzantine processes, some of the other processes also send corrupted messages, as long as they do not send them to all. More specifically, we generalize the classical Byzantine model and consider that Byzantine failures might be partial. In each communication step, some of the processes might send corrupted messages to a subset of the processes. This subset of processes - to which corrupted messages might be sent - could change over time. We compute the exact number of processes that can commit such faults, besides those that commit classical Byzantine failures, while still solving Byzantine agreement. We present a corresponding Byzantine agreement algorithm and prove its optimality by giving resilience and complexity bounds.

This paper is a regular submission.
The paper is a student paper.

*École Polytechnique Fédérale de Lausanne,Switzerland, Email: cheng.wang@epfl.ch
†LIAFA-Université Paris-Diderot, Paris, France, Email: cd@liafa.univ-paris-diderot.fr
‡LIAFA-Université Paris-Diderot, Paris, France, Email: hf@liafa.univ-paris-diderot.fr
§École Polytechnique Fédérale de Lausanne, Switzerland, Email: rachid.guerraoui@epfl.ch
¶INRIA Rennes Bretagne-Atlantique, France, Email: anne-marie.kermarrec@inria.fr

1. Introduction

Pease, Shostak and Lamport introduced the Byzantine model in their landmark papers [1,2]. A Byzantine process is defined as a process that can arbitrarily deviate from the algorithm assigned to it and send corrupted messages to other processes. They considered a synchronous model and proved that agreement is achievable with a fully connected network if and only if the number of Byzantine processes is less than one third of the total number of processes. Dolev extended this result to general networks, in which the connectivity number is more than twice the number of faulty processes [3]. The early work on Byzantine agreement is well summarized in the survey by Fischer [4].

Several approaches have been proposed to circumvent the impossibility of reaching Byzantine agreement in an asynchronous context [5]. The eventually synchronous model was presented in [6]: an intermediate model between synchronous and asynchronous models, allowing some limited periods of asynchrony. Eventual synchrony is considered weak enough to model real systems and strong enough to make Byzantine agreement solvable. Alternative approaches rely on randomized algorithms [7–10]. As Karlin and Yao showed in [11], the *one third* bound is still a tight bound for randomized Byzantine agreement algorithms.

We show in this paper that it is possible to solve Byzantine agreement deterministically even if, beyond the b ($< n/3$) Byzantine processes, some of the other processes also send corrupted messages, as long as they do not send them to all. We show that this is possible deterministically, and even in an eventually synchronous model. We compute the exact number of processes that can commit such partial Byzantine faults, besides those that commit classical Byzantine failures, while still solving Byzantine agreement. For pedagogical purposes, we mainly focus in the main paper on the synchronous context and non-signed messages [1,12]. We discuss signed messages and the eventually synchronous context in Section 4Resilience Lower Boundssection.4 and the Appendices.

We generalize the classical Byzantine model and consider that Byzantine failures might be partial. This generalization is, we believe, interesting in its own right. In each communication step, some of the processes might send corrupted messages to a subset of the processes. The classical Byzantine failure model corresponds to the extreme case where this subset is the entire system. So we consider a system of n processes, of which m can be partially faulty. The processes communicate with each other directly through a complete network. We assume that each partially faulty process p is associated with up to d ($< n-1$) Byzantine communication links. Such a process p is said to be *d-faulty*. The d Byzantine links are *dynamic*: they may be different in different communication rounds. A d-faulty process somehow means that the local computation of the processes remains correct: only the communication links related to the faulty processes are controlled by the adversary - during specific rounds. This captures practical situations where processes experience possibly temporary bugs in specific parts of their code or communication links. From the component failure model's view, our generalization is orthogonal to those of [13–15].

We establish tight bounds on Byzantine agreement in terms of (a) the number of processes to which corrupted messages can be sent and (b) time complexity, i.e. the number of rounds needed to reach agreement. Besides basic distributed computing tools like full information protocols and scenario arguments, we also introduce and make use of a new technique we call "*View-Transform*" which basically enables processes to locally correct partial Byzantine failures and transform a classical Byzantine agreement algorithm into one that tolerates more than $1/3$ failures. Interestingly, this transformation only requires adding a couple more rounds to a classical Byzantine agreement algorithm, i.e., its time complexity does not grow with the number of partial Byzantine faults tolerated. In fact, by tolerating more than $1/3$ Byzantine failures, our algorithm can be faster than classical algorithms in the following sense. In situation where $1/3$ processes are Byzantine, a deterministic Byzantine algorithm [1] need to wait for all correct processes to communicate, even if some of the communication links between processes have

very large delays. In our case, these highly delayed links will be viewed as partial failures, and can be totally tolerated.

For a system with b Byzantine processes and m "d-faulty" processes, Byzantine agreement can be solved among n processes iff $n > \max\{2m + d, 2d + m, b\} + 2b$. There is thus a clear trade-off between the number b of Byzantine failures we can tolerate, the number m of partial Byzantine failures and d. For instance, the system could tolerate $1/6$ fraction of "1-faulty" processes in addition to $(1/3 - \epsilon)$ Byzantine processes. Tolerating fewer classical Byzantine failures would enable us to tolerate many more partial Byzantine ones. For example, if $b = 0$, we can tolerate up to $n/2$ "1-faulty" processes.

The rest of the paper is organized as follows. Section 2Model and Definitionssection.2 describes our partial Byzantine failure model and recalls the Byzantine agreement problem. Section 3The Byzantine Agreement Protocolsection.3 presents our Byzantine agreement algorithm in the synchronous context. Section 4Resilience Lower Boundssection.4 proves the resilience optimality of our algorithm and also discusses the case where messages are signed. Section 5Time Optimalitysection.5 discusses the time optimality of the algorithm. We conclude by reviewing related work in Section 6Concluding Remarkssection.6. For space limitations, we defer the discussion on early decision and eventual synchrony, as well as some correctness proofs to the optional appendices.

2. Model and Definitions

2.1. Synchronous computations

We first consider a synchronous message passing distributed system P of n processes. Each process is identified by a unique id $p \in \{0, 1, \ldots, n-1\}$. As in [1,16], a synchronous computation proceeds in a sequence of rounds.* The processes communicate by exchanging messages round by round within a fully connected point-to-point network. In each round, each process p first sends at most one message to every other process, possibly to all processes, and then p receives the messages sent by other processes. The communication channels are authenticated, i.e. the sender is known to the recipient. Following [1], we consider oral messages† with the following properties: (a) every message sent is delivered; (b) the absence of a message can be detected. In the system, there is a designated process called *transmitter* which has an initial input value from some domain $\mathcal{V}$ to transmit to all processes.

We model an algorithm as a set of deterministic automata, one for each process in the system. Thus, the actions of a process are entirely determined by the algorithm, the initial value of the transmitter and the messages it receives from others.

2.2. Failure model

In short, a d-faulty process p may lie to other processes: in each round, p can send to a subset of d processes Byzantine messages, i.e., messages that differ from those that p has to send following its algorithm. We assume that up to m ($\geqslant 0$) of the processes are partial controlled by the adversary (these processes can send Byzantine messages to d ($< n - 1$) processes) and up to b ($\geqslant 0$) are fully controlled by the adversary. By convention, if $m = 0$, we assume $d = 0$ to make our condition simpler to state.

In each round, the adversary chooses up to d communication links from each partial controlled process that could carry Byzantine messages, while the fully controlled processes could send Byzantine messages. We call an instance of our system of n processes with m d-faulty processes and b Byzantine processes as a *(n, m, d, b)-system*. We refer to the correct processes as well as the d-faulty ones as *non-Byzantine* processes in this paper.

∗. We consider eventually synchronous computations in Appendix IVAppendix 2.The Eventually Synchronous Casesection*.6.

†. We discuss the impact of signed messages in Section 4Resilience Lower Boundssection.4.

2.3. Full information algorithms

We consider full information algorithms in the sense of [1,18,19]. Every process transmits to all processes its entire state in each round, including everything it knows about all values sent by other processes in the previous round. We introduce in the following a collection of notations (a slight extension of [18]) to establish and prove our results.

We use $P^{l:k}$ to denote the set of strings of process identifiers in P of length at least l and at most k, and P^k to denote the set of strings of length k. An empty string has length 0. We use P^+ to denote non-empty strings of symbols in P and P^* to denote all strings including the empty one. We always refer to p_0 as the transmitter in the Byzantine agreement problem, and $\mathcal{V}$ as the domain of values which processes wish to agree on. For convenience, we assume that $\{\perp, 0, 1\} \in \mathcal{V}$ where $\perp$ refers to the empty value.

A *k-round scenario* σ (in a (n, m, d, b)-system P) describes an execution of the algorithm. Intuitively σ describes a communication scheme admissible for the (n, m, d, b)-system. The scenario is determined by the initial value of each process and the communication scheme. Given scenario σ, $\sigma(p_0 p_1 \dots p_k)$ represents the value p_{k-1} tells p_k that p_{k-2} tells p_{k-1} ... that p_0 tells p_1 is p_0's initial value. Formally, a *k-round scenario* σ is a mapping $\sigma : p_0 P^{0:k} \to \mathcal{V}$, such that:

- $\sigma(p_0)$ is the initial value of transmitter p_0.
- There are sets $B(\sigma)$ and $D(\sigma)$ of processes (denoting the set of Byzantine and d-faulty processes, respectively) such that:
  - $|B(\sigma)| \leqslant b$ and $|D(\sigma)| \leqslant m$,
  - for every process $p \notin (B(\sigma) \cup D(\sigma))$: $\sigma(wpq) = \sigma(wp)$ for all $q \in P$ and $w \in p_0 P^{0:k-2}$,
  - for every process $p \in D(\sigma)$ and round l ($\leqslant k$), there is a set T of at most d processes such that for every $q \in P \setminus T$ and every $w \in p_0 P^{l-2}$ we have $\sigma(wpq) = \sigma(wp)$.

Note that $\sigma(wpq) \neq \sigma(wp)$ for some strings w of length l and process q means that q receives a Byzantine message from p in round $l + 1$.

Throughout this paper, we use σ to represent a k-round scenario for a (n, m, d, b)-system with transmitter p_0, d-faulty processes in $D(\sigma)$ and Byzantine processes in $B(\sigma)$. Let $\sigma_p(s) = \sigma(sp)$ for every $s \in p_0 P^{0:k-1}$. σ_p is called the *view* of p. Let $\sigma_{q_1 \dots q_i}(s) = \sigma(s q_1 \dots q_i)$ for every $s \in p_0 P^{0:k-i}$. $\sigma_{q_1 \dots q_i}$ is q_i's view of q_{i-1}'s view ... of q_1's view, or in short q_i's view from $q_1 \dots q_i$. Let $\sigma^{p_0 \dots p_i}$ denote the $(k-i)$-round scenario with transmitter p_i such that $\sigma^{p_0 \dots p_i}(p_i s) = \sigma(p_0 \dots p_i s)$ for every $s \in P^{0:k-i}$. Naturally, $\sigma_p^{p_0 \dots p_i}$ denotes the view of p with respect to scenario $\sigma^{p_0 \dots p_i}$, and $\sigma_{q_1 \dots q_j}^{p_0 \dots p_i}$ denotes the view of q_j from $q_1 \dots q_j$ with respect to scenario $\sigma^{p_0 \dots p_i}$.

Let $\mathcal{U}^k$ be the set of mappings from $p_0 P^{k-1}$ into $\mathcal{V}$. Any k-round algorithm F defined in a (n, m, d, b)-system may be defined on the set of all views; namely as a function F: $\mathcal{U}^k \to \mathcal{V}$.

2.4. The Byzantine agreement problem

We address in this paper the problem of *Byzantine agreement* (also called the Byzantine generals problem in [1]). Each process has an output register which records the outcome of the computation. We assume that the initial value of this register is $nil \notin \mathcal{V}$ and that this output register can be written at most once.

Let F be a k-round algorithm and the output is a value in $\mathcal{V}$. Then we say that F solves Byzantine agreement if, for each k-round scenario σ and every process $p \in P$, the following conditions hold:

- *Termination*: Every non-Byzantine process p outputs value $F(\sigma_p)$.
- *Validity*: If the transmitter p_0 is non-Byzantine, then every non-Byzantine process p outputs the initial value of p_0, i.e. $F(\sigma_p) = \sigma(p_0)$ if $p, p_0 \notin B(\sigma)$.

- *Agreement*: Any two non-Byzantine processes p and q have the same output, i.e. $F(\sigma_p) = F(\sigma_q)$ if $p, q \notin B(\sigma)$.

3. The Byzantine Agreement Protocol

In this section, we present an algorithm we call BA++ (Algorithm 3The Byzantine Agreement Protocolalgocfline.3) for solving Byzantine agreement within a (n, m, d, b)-system. We adopt the description style of [18] for our algorithm. The main theorem is as follows.

Theorem 1. *BA++ is a $(b+3)$-round algorithm that solves Byzantine agreement for a (n, m, d, b)-system if $n > \max\{2m + d, 2d + m, b\} + 2b$.*

At a very high level (Figure 1High-level view of Algorithm BA++figure.1), the idea underlying algorithm BA++ is the following. The processes exchange their messages in a full information manner during $b + 3$ rounds.‡ According to our model, the views obtained at each process contains both partial failures and Byzantine failures. The first step of BA++ is to correct the partial failures. This is challenging because the partial faults introduced in the early rounds would still exist in the subsequent rounds. We address this problem by an algorithm we call *View-Transform* (Algorithm 2The Byzantine Agreement Protocolalgocfline.2): this transforms a view with partial failures into a view without partial failures. Another challenge is to ensure that the views (that resulted from a same scenario) still belong to a same scenario after View-Transform. This is addressed by iterations of *Local-Majority* (Algorithm 1The Byzantine Agreement Protocolalgocfline.1). After applying View-Transform to the original view, the majority algorithm (OM) of Lamport [1] (or any $(b + 1)$-round simultaneous Byzantine agreement algorithm) can be employed to compute a output.

Lemma 1. *Suppose $n > \max\{2m + d, 2d + m, b\} + 2b$. In LM_3 (Algorithm 1The Byzantine Agreement Protocolalgocfline.1), if $\sigma_{sp}^{p_0p_1\dots p_i}(p_ip_{i+1}p_{i+2}) = \sigma_{s'p'}^{p_0p_1\dots p_i}(p_ip_{i+1}p_{i+2})$ for all p_{i+1} and p_{i+2}, then $LM_3(\sigma_{sp}^{p_0p_1\dots p_i}) = LM_3(\sigma_{s'p'}^{p_0p_1\dots p_i})$. If p_i is non-Byzantine, then $LM_3(\sigma_p^{p_0p_1\dots p_i}) = \sigma(p_0p_1\dots p_i)$.*

Proof. The first part of the lemma follows directly from the algorithm, so we only need to show the second part.

If $m = d = 0$ and $b = 0$, the lemma follows directly since there are no failures. In the following, we prove the lemma in the case that $m \neq 0$ or $b \neq 0$.

If p_{i+1} is correct, then there are at least $n-m-b-1$ elements in $\{\sigma_p(p_0\dots p_{i+1}p_{i+2}) : p_{i+2} \in P\backslash p_{i+1}\}$ equal to $\sigma(p_0\dots p_{i+1})$, which implies $\sigma(p_0\dots p_{i+1})$ is added to S.

If p_{i+1} is d-faulty, then there are at most $m + d + b - 1$ values different from $\sigma(p_0\dots p_{i+1})$ in $\{\sigma_p(p_0\dots p_{i+1}p_{i+2}) : p_{i+2} \in P\backslash p_{i+1}\}$. Since $n - m - b - 1 > m + d + b - 1$, only $\sigma(p_0\dots p_{i+1})$ might be added to S.

Now consider p_i. If p_i is correct, then all correct processes will contribute a value $\sigma(p_0\dots p_i)$ to S. So there are at least $n - 1 - m - b$ values equal to $\sigma(p_0\dots p_i)$ in S and at most b values in S different from $\sigma(p_0\dots p_i)$ (contributed by b Byzantine processes). If $m \neq 0$, then $n > 2m + d + 2b \geqslant m + 1 + d + 2b$. If $m = 0$ but $b \neq 0$, then $n > 3b \geqslant m + 1 + d + 2b$. So $n - 1 - m - b$ is always greater than b, the majority value of S is $\sigma(p_0\dots p_i)$, i.e. $LM_3(\sigma_p) = \sigma(p_0\dots p_i)$.

If p_i is d-faulty, then all correct processes except the ones that receive wrong values from p_i will contribute a value $\sigma(p_0\dots p_i)$ to S. So there are at least $n - m - d - b$ values equal to $\sigma(p_0\dots p_i)$ in S, and at most $d + b$ values different from $\sigma(p_0\dots p_i)$ in S. Since $n > m + 2d + 2b$, the majority value of S is still $\sigma(p_0\dots p_i)$, i.e. $LM_3(\sigma_p^{p_0\dots p_i}) = \sigma(p_0\dots p_i)$. □

‡. We discuss how to reduce that number of rounds in Section 5Time Optimalitysection.5.

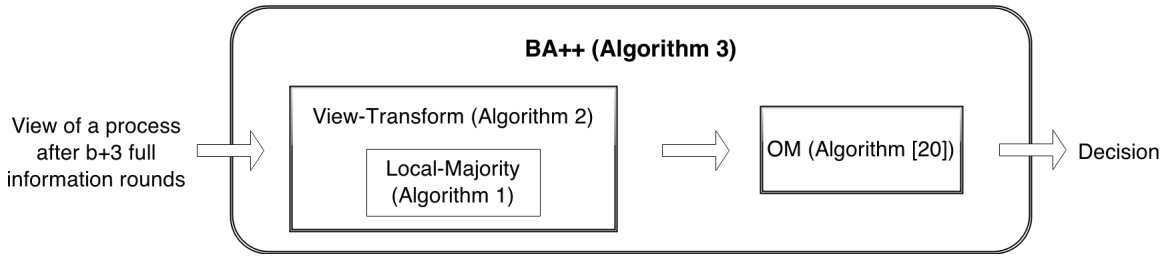

Figure 1. High-level view of Algorithm BA++

Algorithm 1: 3-round Local-Majority (LM_3)

Assume: σ_p is a k-round view of process p for a (n, m, d, b)-system with $k \geqslant 3$ and transmitter p_0.
Code for p:
For every string $p_0p_1 \dots p_i$ and string s with $0 \leqslant |s| \leqslant k-3-i$:

1) p initializes an empty multiset S.
2) For every process $p_{i+1} \in P \backslash p_i$, if at least $n-m-b-1$ elements of $\{\sigma_{sp}^{p_0p_1\dots p_i}(p_ip_{i+1}p_{i+2}) : p_{i+2} \in P \backslash p_{i+1}\}$ have the same value v, p adds v to S.
3) If more than half of S have the same value v', then p sets $LM_3(\sigma_{sp}^{p_0p_1\dots p_i})$ to v'. Otherwise p sets $LM_3(\sigma_{sp}^{p_0p_1\dots p_i})$ to $\perp$.

Algorithm 2: View-Transform VT^p with respect to LM_3

Assume: σ_p is a k-round view of process p for a (n, m, d, b)-system with $k \geqslant 3$ and transmitter p_0. LM_3 is Algorithm 1The Byzantine Agreement Protocolalgocfline.1.
Code for p:
Loop from $i = k-3$ to $i = 0$: (denote the following ith iteration as transform VT_i^p.)

1) Let σ'_p be a copy of σ_p.
2) p changes $\sigma'_p(p_0p_1 \dots p_is)$ to be $LM_3(\sigma_{sp}^{p_0p_1\dots p_i})$ for every $p_1 \dots p_i$ and every string s with $0 \leqslant |s| \leqslant k-3-i$.
3) Let $\sigma_p = \sigma'_p$. (σ'_p is the output of VT_i^p.)

After the loop, p outputs the first $(k-2)$-round view of σ_p.

Algorithm 3: BA++ with respect to LM_3

Assume: σ_p is a $(b+3)$-round view of process p for a (n, m, d, b)-system and transmitter p_0. VT^p is Algorithm 2The Byzantine Agreement Protocolalgocfline.2.
Code for p:

1. Let $\sigma'_p = VT^p(\sigma_p)$ with respect to LM_3.
2. Then p outputs $OM(\sigma'_p)$. Here OM is the Byzantine agreement algorithm in [1].

We show that the output of *View-Transform* for different processes actually comes from a single scenario of a (n, 0, 0, b)-system for which the OM algorithm guarantees Byzantine agreement in $(b+1)$ rounds. We prove this by introducing the following *Scenario-Transform*.

Assume: σ is a k-round scenario for a (n, m, d, b)-system with $k \geqslant 3$ and transmitter p_0. Let $B = B(\sigma)$ be the set of Byzantine processes. VT_i^p is the i-th iteration in Algorithm 2The Byzantine Agreement Protocolalgocfline.2.

Transform:

Loop from $i = k-3$ to $i = 0$: (denote the following ith iteration as transform ST_i)

1) Let σ' be a copy of σ.
2) For each $p \notin B$, apply VT_i^p to σ', i.e. $\sigma'(p_0 \dots p_i s p) = VT_i^p(\sigma_p)(p_0 \dots p_i s)$ for every $s \in P^{0:k-i-1}$. (This Line makes sense because view transforms are independent for different processes.)
3) For every $p \notin B$, $q \in B$, $s \in P^{0:k-i-2}$, set $\sigma'(p_0 \dots p_i s p q)$ to $\sigma'(p_0 \dots p_i s p)$.
4) Let $\sigma = \sigma'$.

After the loop, output σ.

Figure 2: Scenario-Transform (ST) with respect to LM_3

Lemma 2. *Consider a k-round scenario σ for a (n, m, d, b)-system with $k \geqslant 3$ and transmitter p_0. The output scenario of Scenario-Transform (Figure 2The Byzantine Agreement Protocolfigure.2) is a scenario of a (n, 0, 0, b)-system. Moreover, this output scenario satisfies $(ST(\sigma))_p = VT^p(\sigma_p)$ for any non-Byzantine process p. If p_0 is a non-Byzantine transmitter for σ, then p_0 is a correct transmitter for $ST(\sigma)$ such that $ST(\sigma)(p_0) = \sigma(p_0)$.*

Proof. For a non-Byzantine process p, $(ST(\sigma))_p = VT^p(\sigma_p)$ follows immediately from Line 2 which uses VT_i^p as in algorithm VT^p. We now prove that the output scenario is a scenario of a (n, 0, 0, b)-system.

Let ith-σ be the scenario just after the ith loop iteration inside ST. We prove by induction this claim: if $i \leqslant v \leqslant k-3$, then ith-$\sigma(p_0 \dots p_v p_{v+1}) = i$th-$\sigma(p_0 \dots p_v)$ for every non-Byzantine process p_v. Note that if $p_{v+1} \in B$, the claim follows by Line 3 of ST. Thus we only need to prove the claim for the case p_{v+1} is non-Byzantine.

First consider $i = k-3$. In this case, v could only be $k-3$. Suppose p_{k-3} and p_{k-2} are non-Byzantine. Then $(k-3)$th-$\sigma(p_0 \dots p_{k-3} p_{k-2}) = VT_{k-3}^{p_{k-2}}(\sigma_{p_{k-2}})(p_0 \dots p_{k-3})$. According to Line 2 of $VT_{k-3}^{p_{k-2}}$, $VT_{k-3}^{p_{k-2}}(\sigma_{p_{k-2}})(p_0 \dots p_{k-3}) = LM_3(\sigma_{p_{k-2}}^{p_0 \dots p_{k-3}})$. Since $p_{k-3} \notin B$, by Lemma 1lemma.1 $LM_3(\sigma_{p_{k-2}}^{p_0 \dots p_{k-3}}) = \sigma^{p_0 \dots p_{k-3}}(p_{k-3}) = \sigma(p_0 \dots p_{k-3})$. Since $(k-3)$th-$\sigma(p_0 \dots p_{k-3}) = \sigma(p_0 \dots p_{k-3})$, the claim for $k-3$ is proved.

Now suppose the claim is true for $i+1$. Let us prove it for i. We need to show the claim for all $i \leqslant v \leqslant k-j$. First, consider $v = i$ and suppose p_i and p_{i+1} are non-Byzantine. Then according to Line 2 of $VT_i^{p_{i+1}}$, ith-$\sigma(p_0 \dots p_i p_{i+1}) = LM_3\left((i+1)\text{th-}\sigma_{p_{i+1}}^{p_0 \dots p_i}\right)$. Since p_i is non-Byzantine, according to Lemma 1lemma.1, $LM_3\left((i+1)\text{th-}\sigma_{p_{i+1}}^{p_0 \dots p_i}\right) = (i+1)\text{th-}\sigma^{p_0 \dots p_i}(p_i) = (i+1)\text{th-}\sigma(p_0 \dots p_i)$. Hence, ith-$\sigma(p_0 \dots p_i p_{i+1}) = (i+1)$th-$\sigma(p_0 \dots p_i)$. Because the value for $p_0 \dots p_i$ is not changed in the ith loop of ST, ith-$\sigma(p_0 \dots p_i) = (i+1)$th-$\sigma(p_0 \dots p_i)$. Thus ith-$\sigma(p_0 \dots p_i) = i$th-$\sigma(p_0 \dots p_i p_{i+1})$, the claim is true for $v = i$. Now consider $v > i$. According to $VT_i^{p_{v+1}}$ and $VT_i^{p_v}$, ith-$\sigma(p_0 \dots p_v p_{v+1}) = LM_3\left((i+1)\text{th-}\sigma_{p_{i+1} \dots p_{v+1}}^{p_0 \dots p_i}\right)$ and ith-$\sigma(p_0 \dots p_v) = LM_3\left((i+1)\text{th-}\sigma_{p_{i+1} \dots p_v}^{p_0 \dots p_i}\right)$. Since p_v is correct and $v > i$, by induction hypothesis $(i+1)$th-$\sigma_{p_{i+1} \dots p_{v+1}}^{p_0 \dots p_i}$ is equal to $(i+1)$th-$\sigma_{p_{i+1} \dots p_v}^{p_0 \dots p_i}$. Therefore ith-$\sigma(p_0 \dots p_v p_{v+1}) = i$th-$\sigma(p_0 \dots p_v)$, and the claim is proved.

From the claim, we see that in $ST(\sigma)$ every non-Byzantine process always sends correct messages to other processes. So $ST(\sigma)$ is a scenario of (n, 0, 0, b)-system with Byzantine processes $B(\sigma)$. Therefore,

if p_0 is non-Byzantine in σ then p_0 is also correct in $ST(\sigma)$. Because the value of $\sigma(p_0)$ for non-Byzantine process p_0 is never changed in ST, $ST(\sigma)(p_0) = \sigma(p_0)$. □

With all the lemmas above, now we can give a proof of Theorem 1theorem.1.

Proof of Theorem 1theorem.1. Suppose σ is a $(b+3)$-round scenario for (n, m, d, b)-system. By Lemma 2lemma.2 above, $\sigma' = ST(\sigma)$ with respect to LM_3 is a $(b+1)$-round scenario of (n, 0, 0, d)-system. Since $VT^p(\sigma_p) = \sigma'_p$ for every non-Byzantine process p, $OM(VT^p(\sigma_p))$ are equal for all non-Byzantine process which proves the agreement property. Moreover, if p_0 is non-Byzantine, then $OM(VT^p(\sigma_p)) = ST(\sigma)(p_0)$. This shows the validity property. Therefore, the theorem is proved. □

4. Resilience Lower Bounds

We show here that our BA++ algorithm is optimal with respect to resilience; namely, $n > \max\{2m + d, 2d + m, b\} + 2b$ is a tight bound to reach Byzantine agreement. If $m = d = 0$, this bound is $n > 3b$ which is tight by [1]. So in this section, we assume that $m, d > 0$ and show that it is impossible to achieve Byzantine agreement if $n \leqslant 2m + d + 2b$ or $n \leqslant 2d + m + 2b$.

Lemma 3. *If $n \leqslant 2m + d + 2b$, then there is no Byzantine agreement algorithm in a (n, m, d, b)-system.*

Proof. Consider a Byzantine agreement algorithm F for a (n, m, d, b)-system. Since $n \leqslant 2m+d+2b$, P can be partitioned into five non-empty sets G, H, I, J and K, with $|G| \leqslant m$, $|H| \leqslant m$, $|I| \leqslant b$, $|J| \leqslant b$, $|K| \leqslant d$. Select an arbitrary process in G as transmitter p_0. We define scenarios α and β recursively as follows:

i. For every $p \in P$, $k \in K$, $q \in P \backslash K$, let

$$\alpha(p_0) = 0, \alpha(p_0 p) = 0,$$

$$\beta(p_0) = 1, \beta(p_0 k) = 0, \beta(p_0 q) = 1,$$

ii. For every $g \in G$, $h \in H$, $i \in I$, $j \in J$, $k \in K$, $p \in P$, $q \in P \backslash K$, $w \in p_0 P^*$, define the following values recursively on the length of w:

$$\alpha(wgp) = \alpha(wg), \alpha(wip) = \alpha(wi), \alpha(wkp) = \alpha(wk),$$

$$\beta(whp) = \beta(wh), \beta(wjp) = \beta(wj), \beta(wkp) = \beta(wp),$$

$$\alpha(whk) = \beta(whk), \alpha(whq) = \alpha(wh), \alpha(wjp) = \beta(wjp),$$

$$\beta(wgk) = \alpha(wgk), \beta(wgq) = \beta(wg), \beta(wip) = \alpha(wip).$$

It is easy to check that α is a scenario of a (n,m,d,b)-system with d-faulty processes in H and Byzantine processes in J, and that β is a scenario of a (n,m,d,b)-system with d-faulty processes in G and Byzantine processes in I.

In the construction, $\alpha_k = \beta_k$ for all $k \in K$. Thus, $F(\alpha_k) = F(\beta_k)$ for all $k \in K$. Since p_0 is a non-Byzantine process in both α and β, according to Byzantine agreement we have

$$F(\alpha_k) = \alpha(p_0) = 0,$$

$$F(\beta_k) = \beta(p_0) = 1.$$

However, it is a contradiction to that $F(\alpha_k) = F(\beta_k)$ for all $k \in K$. The lemma is proved. □

Lemma 4. *If $n \leqslant 2d + m + 2b$, then there is no Byzantine agreement algorithm in a (n, m, d, b)-system.*

The proof for Lemma 4lemma.4 is similar to the proof of Lemma 3lemma.3. Due to space limitation, we defer the proof to Appendix IAppendix 4.Proof of Lemma 4lemma.4section*.3.

Taking together the algorithm in Section 3The Byzantine Agreement Protocolsection.3 and the lemmas above, we have the following theorem.

Theorem 2. *Byzantine agreement can be solved in a (n, m, d, b)-system if and only if* $n > \max\{2m + d, 2d + m, b\} + 2b$.

Signed messages

So far we have assumed oral message. We now discuss the case where processes could send signed messages [1]. In this case, we also have a tight bound on the number of processes for reaching Byzantine agreement. Following [1], a signed message satisfies the following two properties:

1) The signature of a non-Byzantine process cannot be forged and any alteration of its content can be detected.
2) Every process can verify the authenticity of a signature.

Formally, suppose σ is a k-round scenario for a (n, m, d, b)-system with signed messages. Let $\sigma(p_0 p_1 \dots p_i p)$ $(i < k)$ be a message received by process p. If process p_j $(j \leqslant i)$ is non-Byzantine, then either $\sigma(p_0 \dots p_i p) = \sigma(p_0 \dots p_j)$, or the signature of p_j is forged.

Algorithm 4: Algorithm SBA++

Assume: σ_p is a $(b + 2)$-round view of process p for a (n, m, d, b)-system with signed messages, and p_0 is the transmitter.

Code for p:

1. p initializes an empty set S.
2. For every string $p_0 \dots p_i$ $(0 \leqslant i \leqslant b + 1$, and $p_0, \dots, p_i$ are different processes): if the signatures attached to value $\sigma_p(p_0 \dots p_i)$ are correct, then p adds $\sigma(p_0 p_1 \dots p_i)$ into S.
3. p outputs the majority value of S.

We present an algorithm called SBA++ (Algorithm 4Signed messagesalgocf.4) for solving Byzantine agreement with signed message. Due to space limitation, we move the proof of Algorithm SBA++ and the following theorem into Appendix IIAppendix 4.Byzantine Agreement with Signed Messagessection*.4.

Theorem 3. *Byzantine agreement can be solved for a (n, m, d, b)-system with signed messages if and only if* $n > m + d + b$.

5. Time Optimality

In this section, we investigate the time complexity of reaching Byzantine agreement for a (n, m, d, b)-system. If $m = 0$, the communication rounds needed to reach Byzantine agreement is $b + 1$ by [18]. So in this section, we assume $m > 0$. We show that in some cases ($n \geqslant \max\{2m + 2d, b + 1\} + 2b$) the lower bound of the number of rounds for reaching Byzantine agreement is $b + 2$, and in other cases (e.g. $b = 0$) the lower bound is $b + 3$.

We first show that a $(b + 2)$-round algorithm is available if $n \geqslant \max\{2m + 2d, b + 1\} + 2b$. In this case we have the following 2-round Local-Majority algorithm.

Algorithm 5: 2-round Local-Majority (LM_2)

Assume: σ_p is a k-round view of process p for a (n, m, d, b)-system with $k \geqslant 3$ and p_0 is the transmitter.

Code for p:

For every string $p_0p_1 \dots p_i$ and string s with $0 \leqslant |s| \leqslant k-3-i$:

1) If more than half of $\{\sigma_{sp}^{p_0p_1\dots p_i}(p_ip_{i+1}) : p_{i+1} \in P \backslash p_i\}$ have the same value v, then p sets $LM_2(\sigma_{sp}^{p_0p_1\dots p_i})$ to v. Otherwise p sets $LM_2(\sigma_{sp}^{p_0p_1\dots p_i})$ to $\perp$.

Lemma 5. *Suppose $n \geqslant 2m+2d+2b$ and $n > 2b+1$. In LM_2 (Algorithm 5Time Optimalityalgocf.5), if $\sigma_{sp}^{p_0p_1\dots p_i}(p_ip_{i+1}) = \sigma_{s'p'}^{p_0p_1\dots p_i}(p_ip_{i+1})$ for all p_{i+1}, then $LM_2(\sigma_{sp}^{p_0p_1\dots p_i}) = LM_2(\sigma_{s'p'}^{p_0p_1\dots p_i})$. If p_i is non-Byzantine, then $LM_2(\sigma_p^{p_0p_1\dots p_i}) = \sigma(p_0p_1\dots p_i)$.*

Proof. The first part of the lemma follows directly from the algorithm. So we only need to show the second part.

If p_i is correct, then in $\{\sigma_p(p_0\dots p_{i+1}) : p_{i+1} \in P\backslash p_i\}$ there are at least $n-1-m-b$ values equal to $\sigma(p_0\dots p_i)$ and at most $m+b$ values different from $\sigma(p_0\dots p_i)$ of which b values are contributed by $B(\sigma)$ and m values are contributed by $D(\sigma)$. If $m=d=0$, then $n > 2b+1 = 2m+2b+1$. If $m \neq 0$ and $d \neq 0$, then $n \geqslant 2m+2d+2b > 2m+2b+1$. So n is always greater than $m+2b+1$, the majority values of $\{\sigma_p(p_0\dots p_{i+1}) : p_{i+1} \in P\backslash p_i\}$ are equal to $\sigma(p_0\dots p_i)$, i.e. $LM_2(\sigma_p^{p_0p_1\dots p_i}) = \sigma(p_0p_1\dots p_i)$.

If p_i is d-faulty, then in $\{\sigma_p(p_0\dots p_{i+1}) : p_{i+1} \in P\backslash p_i\}$ there are at least $n-m-d-b$ values equal to $\sigma(p_0\dots p_i)$ and at most $m-1+d+b$ values different from $\sigma(p_0\dots p_i)$ of which b values are contributed by $B(\sigma)$ and $m-1+d$ values are contributed by $D(\sigma)$. Since $n \geqslant 2m+2d+2b$, we have $n-1 > 2(m-1+d+b)$, the majority values are equal to $\sigma(p_0\dots p_i)$, i.e. $LM_2(\sigma_p^{p_0p_1\dots p_i}) = \sigma(p_0p_1\dots p_i)$. □

Lemma 6. *If $n \geqslant \max\{2m+2d, b+1\}+2b$, then Byzantine agreement can be solved in $b+2$ rounds for a (n, m, d, b)-system.*

Proof. Section 3The Byzantine Agreement Protocolsection.3 uses 3-round algorithm LM_3 (Algorithm 1The Byzantine Agreement Protocolalgocfline.1) to implement the sub-algorithm *View-Transform* (Algorithm 2The Byzantine Agreement Protocolalgocfline.2), and then get a $(b+3)$-round Byzantine agreement algorithm. When $n \geqslant \max\{2m+2d, b+1\}+2b$, we have a 2-round Local-Majority algorithm LM_2. Thus if we replace LM_3 with LM_2, we obtain a $(b+2)$-round Byzantine agreement algorithm. □

In the following, we prove that $b+2$ is also a lower bound of rounds for reaching Byzantine agreement. Specially, $b+2$ is a tight bound for the case $n \geqslant \max\{2m+2d, b+1\}+2b$.

Theorem 4. *Byzantine agreement for a (n, m, d, b)-system ($m, d > 0$) requires at least $b+2$ rounds.*

Proof. Suppose in contrary that there is a $(b+1)$-round Byzantine agreement algorithm F. For any string w, we use $\bar{w}$ to denote the number corresponding to w with radix n.

Select an arbitrarily process p_0 in the system as a fixed transmitter. For $0 \leqslant x \leqslant n^{b+1}+1$, define $\alpha_x : p_0P^{0:b} \to \{0,1\}$ as

$$\text{for } w \in p_0P^{0:b}, \alpha_x(w) = \begin{cases} 0 & \text{if } \overline{w} < x, \\ 1 & \text{otherwise.} \end{cases}$$

It is easy to see that $\alpha_0(w)$ is always equal to 1, so $F(\alpha_0) = 1$. For the same reason, $F(\alpha_{n^{b+1}+1}) = 0$. We claim: α_x and α_{x+1} are views derived from a same scenario for all $1 \leqslant x \leqslant n^{b+1}$. If so, by the agreement property of F we have $F(\alpha_x) = F(\alpha_{x+1})$. Then, we have $F(\alpha_0) = F(\alpha_1) = \dots = F(\alpha_{n^{b+1}+1})$. This is a contradiction to $F(\alpha_0) = 1$ and $F(\alpha_{n^{b+1}+1}) = 0$. Now it remains to prove the claim.

For $1 \leqslant x \leqslant n^{b+1}$, let $x = \overline{q_0q_1\dots q_b}$. Since $n > b+3$, there exists two different processes q_{b+1} and q_{b+2} (assume $q_{b+1} > q_{b+2}$ without loss of generality) in $P\backslash\{q_0\dots q_b\}$. Define a function $\sigma : p_0P^{0:b+1} \to \{0,1\}$

as

$$\text{for } w \in p_0P^{0:b+1}, \sigma(w) = \begin{cases} 0 & \text{if } p_0 < q_0, \\ 0 & \text{if } p_0 = q_0 \text{ and } w = q_0 \dots q_i q s, \text{with } 0 \leqslant i \leqslant b, q < q_{i+1}, \\ 1 & \text{otherwise.} \end{cases}$$

It is easy to check that $\sigma_{q_{b+1}} = \alpha_x$ and $\sigma_{q_{b+2}} = \alpha_{x+1}$. If $p_0 < q_0$, then $\sigma(w)$ is always equal to 0. So α_x and α_{x+1} come from an admissible scenario σ. If $p_0 > q_0$, for the similar reason the claim is correct. If $q_0 = p_0$, then for every process p in $P\backslash\{q_0, \dots, q_b\}$ we always have $\sigma(wpq) = \sigma(wp)$. If the set $\{q_0, \dots, q_b\}$ has less than b elements, then let $B(\sigma) = \{q_0, \dots, q_b\}$ and σ is a $(b+1)$-round scenario. Thus α_x and α_{x+1} come from an admissible scenario σ. If the set $\{q_0, \dots, q_b\}$ has $b+1$ different elements, then let ϕ be as follows:

$$\text{for } w \in p_0P^{0:b+1}, \phi(w) = \begin{cases} 1 & \text{if } w = q_0 \dots q_b q \text{ with } q < q_{b+1} \text{ and } q \neq q_{b+2}, \\ \sigma(w) & \text{otherwise.} \end{cases}$$

ϕ is a $(b+1)$-round scenario with Byzantine processes $\{q_0, \dots, q_{b-1}\}$ and d-faulty processes $\{q_b\}$. Also we have $\phi_{q_{b+1}} = \sigma_{q_{b+1}} = \alpha_x$ and $\phi_{q_{b+2}} = \sigma_{q_{b+2}} = \alpha_{x+1}$. Thus α_x and α_{x+1} come from an admissible scenario ϕ. Hence, the claim we mentioned is always correct. So the theorem follows. □

Now we show that $b+3$ could be lower bound in certain cases. Specifically, suppose $b = 0$, we prove that 3 rounds is a lower bound.

Lemma 7. *Suppose $m, d > 0$ and $max\{2m + d, 2d + m\} < n < 2m + 2d$, then there is no 2-round Byzantine agreement algorithm for a (n, m, d, 0)-system.*

Proof. Let F be a 2-round Byzantine agreement algorithm. Select an arbitrarily process p_0 in P as transmitter. By the assumption of the lemma, $P\backslash p_0$ can be partitioned into four sets G, H, I and J such that $|G| \leqslant m-1$, $|H| \leqslant m-1$, $0 < |I| \leqslant d$, $0 < |J| \leqslant d$. We define two 2-round scenarios α (with d-faulty processes in $G \cup \{p_0\}$) and β (with d-faulty processes in $H \cup \{p_0\}$) as follows.

i. For every $i \in I$, $j \in J$, $q_i \in P\backslash I$, $q_j \in P\backslash J$ let

$$\alpha(p_0) = 0, \alpha(p_0 i) = 1, \alpha(p_0 q_i) = \alpha(p_0),$$

$$\beta(p_0) = 1, \beta(p_0 j) = 0, \beta(p_0 q_j) = \beta(p_0),$$

ii. For every $g \in G$, $h \in H$, $i \in I$, $q_g \in P\backslash(G \cup \{p_0\})$, $q_h \in P\backslash(H \cup \{p_0\})$, $q_i \in P\backslash I$, $p \in P$ let

$$\alpha(p_0 p_0 i) = \beta(p_0 p_0 i) = 1,$$

$$\alpha(p_0 g i) = 1, \alpha(p_0 g q_i) = \alpha(p_0 g), \alpha(p_0 q_g p) = \alpha(p_0 q_g),$$

$$\beta(p_0 h i) = 0, \beta(p_0 h q_i) = \beta(p_0 h), \beta(p_0 q_h p) = \beta(p_0 q_h).$$

In the construction, $\alpha_i = \beta_i$ for all $i \in I$. Thus for any $i \in I$,

$$0 = \alpha(p_0) = F(\alpha_i) = F(\beta_i) = \beta(p_0) = 1,$$

giving a contradiction. □

6. Concluding Remarks

There have been several attempts to overcome the need for three-times redundancy in Byzantine agreement [20–24]. Several researchers considered stronger communication models such as broadcast channels. In the synchronous setting, Rabin and Ben-Or [20] introduced the notion of global broadcast channel and showed that any multiparty computation could be achieved with two-times redundancy only. A partial broadcast channel was defined by Fitzi and Maurer [21], and corresponding lower bounds for reaching Byzantine agreement were presented in [22–24]. Problems of secure communication and computation in the presence of a Byzantine adversary within an [3,25] incomplete network have also been studied [3,25].

Accounting for the fact that communication failures sometimes dominate computation ones (due to the high reliability of hardware and operating systems), some models focused on communication failures [26,27] or hybrid failures [12,28]. These include models where the Byzantine components are the communication channels instead of (or in addition to) the processes. For instance, in [14,29], Santoro and Widmayer showed that agreement cannot be achieved with $\lceil \frac{n-1}{2} \rceil$ Byzantine communication faults. Our Theorem 2theorem.2 generalizes this result. Actually in Theorem 2theorem.2, taking $m = \lceil \frac{n-1}{2} \rceil$, $d = 1$ and $b = 0$ would force $n < 2m + d + b$, which implies the impossibility of Byzantine agreement.

References

[1] Leslie Lamport, Robert Shostak, and Marshall Pease. The Byzantine generals problem. *TOPLAS*, 4(3):382–401, 1982.

[2] Marshall Pease, Robert Shostak, and Leslie Lamport. Reaching agreement in the presence of faults. *JACM*, 27(2):228–234, 1980.

[3] Danny Dolev. The Byzantine generals strike again. *Journal of Algorithms*, 3(1):14–30, 1982.

[4] Michael J Fischer. The consensus problem in unreliable distributed systems (a brief survey). In *FCT*, pages 127–140. Springer, 1983.

[5] Michael J Fischer, Nancy A Lynch, and Michael S Paterson. Impossibility of distributed consensus with one faulty process. *JACM*, 32(2):374–382, 1985.

[6] Cynthia Dwork, Nancy Lynch, and Larry Stockmeyer. Consensus in the presence of partial synchrony. *JACM*, 35(2):288–323, 1988.

[7] Nicolas Braud-Santoni, Rachid Guerraoui, and Florian Huc. Fast Byzantine agreement. In *PODC*, pages 57–64. ACM, 2013.

[8] Cynthia Dwork, David Peleg, Nicholas Pippenger, and Eli Upfal. Fault tolerance in networks of bounded degree. *Journal on Computing*, 17(5):975–988, 1988.

[9] Valerie King, Steven Lonargan, Jared Saia, and Amitabh Trehan. Load balanced scalable Byzantine agreement through quorum building, with full information. In *ICDCN*, pages 203–214. Springer, 2011.

[10] Michael O Rabin. Randomized Byzantine generals. In *FOCS*, pages 403–409. IEEE, 1983.

[11] Anna Karlin and Andrew Yao. Probabilistic lower bounds for Byzantine agreement. *Unpublished document*, 1986.

[12] Patrick Lincoln and John Rushby. A formally verified algorithm for interactive consistency under a hybrid fault model. In *FTCS*, pages 402–411. IEEE, 1993.

[13] Lewis Tseng and Nitin Vaidya. Iterative approximate byzantine consensus under a generalized fault model. In *DCN*, pages 72–86. Springer, 2013.

[14] Nicola Santoro and Peter Widmayer. Agreement in synchronous networks with ubiquitous faults. *Theoretical Computer Science*, 384(2):232–249, 2007.

[15] S Amitanand, I Sanketh, K Srinathant, V Vinod, and C Pandu Rangan. Distributed consensus in the presence of sectional faults. In *PODC*, pages 202–210. ACM, 2003.

[16] Sam Toueg, Kenneth J Perry, and TK Srikanth. Simple and efficient Byzantine general algorithms with early stopping. Technical report, Cornell University, 1984.

[17] TK Srikanth and Sam Toueg. Optimal clock synchronization. *JACM*, 34(3):626–645, 1987.

[18] Michael J Fischer and Nancy A Lynch. A lower bound for the time to assure interactive consistency. *Information Processing Letters*, 14(4):183–186, 1982.

[19] Nancy A Lynch. *Distributed algorithms*. Morgan Kaufmann, 1996.

[20] Tal Rabin and Michael Ben-Or. Verifiable secret sharing and multiparty protocols with honest majority. In *PODC*, pages 73–85. ACM, 1989.

[21] Mattias Fitzi and Ueli Maurer. From partial consistency to global broadcast. In *STOC*, pages 494–503. ACM, 2000.

[22] DVS Ravikant, V Muthuramakrishnan, V Srikanth, K Srinathan, and C Pandu Rangan. On Byzantine agreement over (2, 3)-uniform hypergraphs. *Distributed Computing*, pages 450–464, 2004.

[23] Jeffrey Considine, Matthias Fitzi, Matthew Franklin, Leonid A Levin, Ueli Maurer, and David Metcalf. Byzantine agreement given partial broadcast. *Journal of Cryptology*, 18(3):191–217, 2005.

[24] Alexander Jaffe, Thomas Moscibroda, and Siddhartha Sen. On the price of equivocation in Byzantine agreement. In *PODC*, pages 309–318. ACM, 2012.

[25] Matthew Franklin and Rebecca N Wright. Secure communication in minimal connectivity models. In *EUROCRYPT'98*, pages 346–360. Springer, 1998.

[26] Kenneth J Perry and Sam Toueg. Distributed agreement in the presence of processor and communication faults. *Software Engineering*, 12(3):477–482, 1986.

[27] Ulrich Schmid, Bettina Weiss, and John Rushby. Formally verified Byzantine agreement in presence of link faults. In *DCS*, pages 608–616. IEEE, 2002.

[28] Li Gong, Patrick Lincoln, and John Rushby. Byzantine agreement with authentication: Observations and applications in tolerating hybrid and link faults. *DCFTS*, 10:139–158, 1998.

[29] Nicola Santoro and Peter Widmayer. Time is not a healer. In *STACS 89*, pages 304–313. Springer, 1989.

[30] Danny Dolev and H Raymond Strong. Polynomial algorithms for multiple processor agreement. In *SOTC*, pages 401–407. ACM, 1982.

[31] Axel W Krings and Thomas Feyer. The Byzantine agreement problem: optimal early stopping. In *HICSS-32*, pages 12–pp. IEEE, 1999.

[32] Miguel Castro and Barbara Liskov. Practical Byzantine fault tolerance. In *OSDI*, pages 173–186, 1999.

Appendix I. Proof of Lemma 4lemma.4

Proof. Consider a Byzantine agreement algorithm F for a (n, m, d, b)-system. Since $n \leqslant 2d+m+2b$, P can be partitioned into five non-empty sets G, H, I, J and K, with $|G| \leqslant m$, $|H| \leqslant d$, $|I| \leqslant d$, $|J| \leqslant b$, $|K| \leqslant b$. Select an arbitrarily process in G as transmitter p_0. We define scenarios α and β recursively as follows:

i. For every $h \in H$, $i \in I$, $q_\alpha \in P\backslash H$, $q_\beta \in P\backslash I$ let

$$\alpha(p_0) = 0, \alpha(p_0 h) = 1, \alpha(p_0 q_\alpha) = 0,$$
$$\beta(p_0) = 1, \beta(p_0 i) = 0, \beta(p_0 q_\beta) = 1,$$

ii. For every $g \in G$, $h \in H$, $i \in I$, $j \in J$, $k \in K$, $p \in P$, $q_\alpha \in P\backslash H$, $q_\beta \in P\backslash I$, $w \in p_0 P^*$, define the following values recursively on the length of w:

$$\alpha(whp) = \alpha(wg), \alpha(wip) = \alpha(wi), \alpha(wkp) = \alpha(wk),$$
$$\beta(whp) = \beta(wh), \beta(wip) = \beta(wi), \beta(wjp) = \beta(wj),$$
$$\alpha(wgq_\alpha) = \alpha(wg), \alpha(wgh) = \beta(wg), \alpha(wjp) = \beta(wjp),$$
$$\beta(wgq_\beta) = \beta(wg), \beta(wgi) = \alpha(wg), \beta(wkp) = \alpha(wkp).$$

It is easy to check that α is a scenario of a (n,m,d,b)-system with d-faulty processes in G and Byzantine processes in J, and that β is a scenario of a (n,m,d,b)-system with d-faulty processes in G and Byzantine processes in K.

In the construction, $\alpha_h = \beta_h$ and $\alpha_i = \beta_i$ for all $h \in H$ and $i \in I$. Thus for any $h \in H$,

$$0 = \alpha(p_0) = F(\alpha_h) = F(\beta_h) = \beta(p_0) = 1,$$

giving a contradiction. □

Appendix II. Byzantine Agreement with Signed Messages

We consider that processes send signed messages. Following [1], a *signed message* satisfies the following two properties:

1) A non-Byzantine process's signature cannot be forged and any alteration of the content of its signed messages can be detected.
2) Any process can verify the authenticity of a process's signature.

Formally, suppose σ is a k-round scenario for a (n, m, d, b)-system with signed messages. Let $\sigma(p_0 p_1 \dots p_i p)$ $(i < k)$ be a message received by process p. If process p_j $(j \leqslant i)$ is non-Byzantine, then either

1. $\sigma(p_0 \dots p_i p) = \sigma(p_0 \dots p_j)$, or
2. the signature of p_j is forged.

In this new setting, we have the following main result:

Theorem 5. *Byzantine agreement can be solved for a (n, m, d, b)-system with signed messages if and only if* $n > m + d + b$.

Lemma 8. SBA++ *(Algorithm 6Appendix II. Byzantine Agreement with Signed Messagesalgocf.6) solves Byzantine agreement for a (n, m, d, b)-system with signed messages if* $n > m + d + b$.

Proof. First suppose the transmitter p_0 is non-Byzantine. By definition of a signed message, every message $\sigma_p(p_0 \dots p_i)$ $(i \leqslant b+1)$ is either equal to $\sigma(p_0)$, or is detected as forged message. So set S contains

Algorithm 6: Algorithm SBA++

Assume: σ_p is a $(b+2)$-round view of process p for a (n, m, d, b)-system with signed messages, and p_0 is the transmitter.

Code for p:

1. p initializes an empty set S.
2. For every string $p_0 \dots p_i$ ($0 \leqslant i \leqslant b+1$, and $p_0, \dots, p_i$ are different processes): if the signatures attached to value $\sigma_p(p_0 \dots p_i)$ are correct, then p adds $\sigma(p_0 p_1 \dots p_i)$ into S.
3. p outputs the majority value of S.

at most $\sigma(p_0)$. If p_0 is correct, then $\sigma_p(p_0) = \sigma(p_0)$ and $\sigma(p_0)$ is added into S. If p_0 is d-faulty, then there must be at least one correct process such that $\sigma(p_0 q) = \sigma(p_0)$ since $n > m + d + b$. And then we have $\sigma(p_0 q p) = \sigma(p_0 q) = \sigma(p_0)$. According to Line 2 of SBA++, $\sigma_p(p_0 q) = \sigma(p_0)$ is added into S. Therefore, S contains a single value $\sigma(p_0)$. Consequently, if p is non-Byzantine, p outputs $\sigma(p_0)$.

Now assume the transmitter p_0 is Byzantine. Let S_p and $S_{p'}$ be the corresponding set S initiated by p and p' in Line 1 of SBA++. We show that $S_p = S_{p'}$ for any two non-Byzantine processes p and p'. Suppose $\sigma_p(p_0 \dots p_k)$ is included in S_p. Let p_l ($l \leqslant k$) be the non-Byzantine process in $p_0 \dots p_k$ with the smallest subscript. Then all processes in $p_0 \dots p_{l-1}$ are Byzantine, which implies $l \leqslant b$. Since the signatures attached to $\sigma_p(p_0 \dots p_k)$ are correct, $\sigma(p_0 \dots p_l) = \sigma_p(p_0 \dots p_k)$. Since $n - m - b > d$, p_l sends $\sigma(p_0 \dots p_l)$ to at least one correct process q. Then $\sigma_{p'}(p_0 \dots p_l q)$ is equal to $\sigma(p_0 \dots p_l)$. According to Line 2 of SBA++, $\sigma_{p'}(p_0 \dots p_l q) = \sigma(p_0 \dots p_l) = \sigma_p(p_0 \dots p_k)$ is added into $S_{p'}$. Therefore, we have $S_p \subset S_{p'}$. Since p and p' are two arbitrary non-Byzantine process, we also have $S_{p'} \subset S_p$. That is to say $S_p = S_{p'}$. According to Line 3 of SBA++, all non-Byzantine processes output a same value. □

Proof of Theorem 5theorem.5. From the lemma above, we know that if $n > m + d + b$ then Byzantine agreement is solvable. Now we show that if $n \leqslant m + d + b$ then Byzantine agreement is impossible.

Suppose by contradiction that F is a Byzantine agreement algorithm for a (n, m, d, b)-system with signed messages and $n \leqslant m + d + b$. We separate the processes into three sets G, H and I such that $|G| \leqslant m$, $|H| \leqslant b$, $|I| \leqslant d$. Select an arbitrarily process in G as transmitter p_0. We define the scenarios α and β (both with Byzantine processes in H and d-faulty processes in G) recursively as follows:

i. For every $i \in I$, $q \in P \backslash I$ let

$$\alpha(p_0) = 0, \alpha(p_0 i) = \perp, \alpha(p_0 q) = \alpha(p_0),$$

$$\beta(p_0) = 1, \beta(p_0 i) = \perp, \beta(p_0 q) = \beta(p_0).$$

ii. For every $g \in G$, $h \in H$, $i \in I$, $p \in P$, $q \in P \backslash I$, $w \in p_0 P^*$, define the following values recursively on the length of w:

$$\alpha(wgi) = \beta(wgi) = \perp, \alpha(wgq) = \alpha(wg), \beta(wgq) = \beta(wg),$$

$$\alpha(whp) = \alpha(wip) = \beta(whp) = \beta(wip) = \perp.$$

Moreover, $\alpha_i = \beta_i$ for all $i \in I$ since $\alpha_i(w) = \beta_i(w) = \perp$ for all string $w \in p_0 P^*$. Thus for any $i \in I$,

$$0 = \alpha(p_0) = F(\alpha_i) = F(\beta_i) = \beta(p_0) = 1,$$

giving a contradiction. □

Appendix III. Early Decision

The work in [30,31] showed that processes could make an early decision if the number of actual Byzantine failures is less than the maximal number of failures it can tolerate. We show here how we can achieve early decision with partial Byzantine failures.

Theorem 6. *Consider a (n, m, d, b)-system ($m, d > 0$) and f denotes the number of actual Byzantine processes during an execution. Then Byzantine agreement can be solved in the following number of rounds:*

- $min\{2(f+2), 2(d+1)\}$, *if* $n \geqslant max\{2m+2d, b+1\} + 2b$,
- $min\{3(f+2), 3(d+1)\}$, *if* $n > max\{2m+d, 2d+m, b\} + 2b$.

Proof. First consider $n > \max\{2m+d, 2d+m, b\} + 2b$. From Lemma 1lemma.1, for any scenario σ we have $LM_3(\sigma_p^{p_0}) = \sigma(p_0)$ provided that p_0 is non-Byzantine. By definition $\sigma_p^{p_0} = \sigma_p$, so we have $LM_3(\sigma_p) = \sigma(p_0)$. This means every non-Byzantine process could get the initial value of the non-Byzantine transmitter p_0 despite that p_0 might be partial faulty. Thus by applying LM_3 to a 3-round scenario σ, we could obtain a 3-round reliable broadcast algorithm. If we use this reliable broadcast algorithm as a broadcast primitive in the early deciding algorithms in [30,31], then we could get early deciding Byzantine agreement for a (n, m, d, b)-system as well. The time complexity of algorithms in [30,31] is $min\{f+2, b+1\}$. Since we replace one round broadcast with three rounds broadcast, the time complexity of early deciding algorithm with 3-round reliable broadcast is $min\{3(f+2), 3(b+1)\}$.

The result for $n \geqslant max\{2m+2d, b+1\} + 2b$ follows from the same idea. □

Appendix IV. The Eventually Synchronous Case

We considered so far synchronous computations. However, it is also possible to tolerate partial failures in eventually synchronous systems. In this section, we first present a reliable broadcast implementation that tolerates partial Byzantine failures. Here, reliable broadcast ensures that if a non-Byzantine process broadcasts a message then other processes will receive the same message eventually (no such guarantee for Byzantine processes). This broadcast primitive thus can be plugged into an algorithm like [32].

We assume here that after an unknown but finite time the system become synchronous [19]. Within an eventually synchronous system, the processes could not distinguish message delay from the absence of a message. We consider a *static* (n, m, d, b)-system which includes up to b Byzantine processes and up to m partial faulty processes each of which is associated with up to d *fixed* Byzantine links. We first show that the algorithm LM_2 and LM_3 can be modified to achieve reliable broadcast in an eventually synchronous (n, m, d, b)-system.

Algorithm 7: 2-round Reliable-Broadcast (RB_2)

Assume: σ_p is a 2-round view of process p for a static (n, m, d, b)-system with $k \geqslant 2$ and p_0 is the transmitter.

Code for p:

1. Waits until receiving more than $n-m-d-b$ values for $\{\sigma_p(p_0p_1) : p_1 \in P \backslash p_0\}$ with a same value v, then output v.

Lemma 9. *In RB_2 (Algorithm 7Appendix IV. The Eventually Synchronous Casealgocf.7), if $n \geqslant 2m + 2d + 2b$ and p_0 is non-Byzantine, then $RB_2(\sigma_p) = \sigma(p_0)$.*

Proof. As in Lemma 5lemma.5, p_0 will receive $n-m-d-b$ $\sigma_p(p_0p_1)$ that equal to $\sigma(p_0)$ from $n-m-d-b$ correct processes. Since $2(n-m-d-b) > n-1$, the lemma follows. □

Algorithm 8: 3-round Reliable-Broadcast (RB_3)

Assume: σ_p is a 3-round view of process p for a static (n, m, d, b)-system with $k \geqslant 3$ and p_0 is the transmitter.

Code for p:

1. p initializes an empty set S.
2. Waits until receiving $n-m-b-1$ values for $\{\sigma_p(p_0p_1p_2) : p_2 \in P\backslash p_1\}$ with a same value v, then p adds v to S.
3. Waits until $n-m-d-b$ values in S have a same value v', then p outputs v'.

Lemma 10. *In RB_3 (Algorithm 8Appendix IV. The Eventually Synchronous Casealgocf.8), if $n > \max\{2m+d, 2d+m, b\} + 2b$ and p_0 is non-Byzantine, then $RB_3(\sigma_p) = \sigma(p_0)$.*

Proof. If p_1 is correct, there are at least $n-m-b-1$ values equal to $\sigma(p_0p_1)$ in $\{\sigma_p(p_0p_1p_2) : p_2 \in P\backslash p_1\}$ from correct processes, which implies $\sigma(p_0p_1)$ will be added to S eventually.

If p_1 is d-faulty, there are at most $m+d+b-1$ values different from $\sigma(p_0p_1)$ in $\{\sigma_p(p_0p_1p_2) : p_2 \in P\backslash p_1\}$. Since $n-m-b-1 \geqslant m+d+b-1$, only $\sigma(p_0p_1)$ might be added to S.

Now consider the transmitter. If p_0 is non-Byzantine, all correct processes except the one receiving wrong values from p_0 will contribute a value $\sigma(p_0)$ to S. So S will eventually include at least $n-m-d-b$ values equal to $\sigma(p_0)$ and at most $d+b$ values different from $\sigma(p_0)$. Since $n > m+2d+2b$, $RB_3(\sigma_p)$ can only be $\sigma(p_0)$. □

If RB_2 or RB_3 is employed as a broadcast primitive, i.e. a process broadcasts a message by executing an instance of RB_2 or RB_3, then the messages broadcast by non-Byzantine processes will be received by other non-Byzantine processes as if there are no partial failures. In this way, RB_2 and RB_3 could play the role of reliable broadcast for a (n, m, d, b)-system. We could then use our reliable broadcast primitive (either RB2 or RB3) within an algorithm such as PBFT [32]. We obtain the following theorem.

Theorem 7. *Byzantine agreement can be solved assuming eventually synchronous computation of a static (n, m, d, b)-system ($m, d > 0$) if and only if $n > \max\{2m+d, 2d+m, b\} + 2b$.*

Proof. The sufficiency follows from the above discussion. The necessity comes from Lemma 3lemma.3 and Lemma 4lemma.4. □