Multidimensional Approximate Agreement with Asynchronous Fallback

Diana Ghinea
ghinead@ethz.ch
ETH Zürich
Zürich, Switzerland

Chen-Da Liu-Zhang
chen-da.liuzhang@ntt-research.com
NTT Research
Sunnyvale, USA

Roger Wattenhofer
wattenhofer@ethz.ch
ETH Zürich
Zürich, Switzerland

ABSTRACT

Multidimensional Approximate Agreement considers a setting of n parties, where each party holds a vector in $\mathbb{R}^D$ as input. The honest parties are required to obtain very close outputs in $\mathbb{R}^D$ that lie inside the convex hull of their inputs.

Existing Multidimensional Approximate Agreement protocols achieve resilience against $t_s < n/(D+1)$ corruptions under a synchronous network where messages are delivered within some time Δ, but become completely insecure as soon as a single message is further delayed. On the other hand, asynchronous solutions do not rely on any delay upper bound, but only achieve resilience up to $t_a < n/(D+2)$ corruptions.

We investigate the feasibility of achieving Multidimensional Approximate Agreement protocols that achieve simultaneously guarantees in both network settings: We want to tolerate t_s corruptions when the network is synchronous, and also tolerate $t_a \leq t_s$ corruptions when the network is asynchronous. We provide a protocol that works as long as $(D+1) \cdot t_s + t_a < n$, and matches several existing lower bounds.

KEYWORDS

Approximate Agreement, Multidimensional Approximate Agreement, hybrid protocols

1 INTRODUCTION

Agreement primitives constitute fundamental building blocks in the distributed systems literature. They allow a set of n parties with possibly distinct input values to agree on a value subject to some validity condition, even when t of these parties are Byzantine and exhibit malicious behavior.

We need to distinguish the synchronous and the asynchronous network model. In the synchronous model, it is guaranteed that the messages sent are delivered within a known amount of time Δ. In contrast, the asynchronous model does not assume any upper bound on the network delay (only that the messages get delivered eventually). From a practical perspective, both models make sense, as they reflect stable network conditions (synchronous) or networks in distress and possibly under attack (asynchronous).

In many agreement protocols, asynchronous networks pose significant limitations. For example, the seminal result of Fischer, Lynch and Paterson [17] shows that the problem of Byzantine Agreement, where parties must agree on the exactly same output value, cannot be achieved deterministically even under the presence of a single crash failure.

Because of this fundamental limitation, Dolev et al. [13] introduced the *Approximate Agreement* (AA) problem. AA relaxes the agreement requirement by allowing the parties to obtain ε-close outputs that lie within the range of their real-valued inputs. AA is of great interest in several practical applications, such as in clock synchronization [21, 23, 33], or for robots gathering on a line [8].

In this paper, we study the more general case of *Multidimensional Approximate Agreement*, or *D-dimensional Approximate Agreement* (D-AA) with $D > 1$, introduced independently by Mendes and Herlihy [26] and Vaidya and Garg [32]. Here, each party holds a vector in $\mathbb{R}^D$ as input, and the honest parties try to converge to ε-close outputs in $\mathbb{R}^D$ that lie in the convex hull of their inputs.

Considering higher dimensions turns out to be useful in several practical applications, including scenarios where robots need to converge to close locations in a 2 or 3-dimensional space [30], in distributed voting where the preferences are described by assigning weights, or in optimization problems, and maybe most prominently in machine learning [14, 31]: In federated machine learning, n parties (e.g., companies, hospitals) want to (or, must) keep their training data private, but they agree to improve their model based on the data of other parties. So each party runs its own machine learning model, and learns with its own data. From time to time, the parties exchange their learning parameters (in particular gradients, which are vectors). The parties try to approximately agree on a gradient, while being resilient to Byzantine faults. This is a D-AA problem, with D being the dimension of the model.

Previous works on D-AA have shown that when the network is synchronous, and when requiring the honest parties to agree on exactly the same values, the optimal resilience is $(D+1) \cdot t < n$ [32]. However, these protocols assume a stable synchronous network, and completely fail as soon as the synchrony assumption is violated. On the other hand, asynchronous protocols remain secure even when the network is unstable, but have to settle for the lower resilience of $(D+2) \cdot t < n$ [26, 32].

A natural question is to investigate whether there is a protocol that simultaneously achieves the best guarantees from both settings. That is, tolerating a high corruption threshold t_s when the network is synchronous, and at the same time a possibly lower number of corruptions $t_a \leq t_s$ when the network is asynchronous. Indeed, primitives with such hybrid security guarantees have received increased attention in the last few years [3, 5, 7, 12, 20, 28]. Concretely, we ask the following question:

> *Is there a D-dimensional Approximate Agreement (D-AA) protocol that tolerates up to t_s corruptions when the network is synchronous, and at the same time tolerates $t_a \leq t_s$ corruptions when the network is asynchronous?*

Our Contributions. This question was completely solved recently by Ghinea, Liu-Zhang and Wattenhofer [20] for the uni-dimensional

($D = 1$) case, showing that this is possible if and only if $2t_s + t_a < n$. We study the problem for higher dimensions in the affirmative by presenting a D-AA protocol that is secure as long as $(D + 1) \cdot t_s + t_a < n$. This trade-off matches previous lower bounds: By setting $t_a = 0$, this is an optimal-resilience synchronous D-AA protocol [32], by setting $t_s = t_a$, we match the necessary condition in the asynchronous model $n > (D + 2) \cdot t_a$ [26, 32], and when $D = 1$, the lower bound matches that of [20].

1.1 Related Work

Approximate Agreement. AA was introduced in the seminal work of Dolev et al. in [13] for the uni-dimensional case, and has been studied in both synchronous and asynchronous networks. The authors showed a synchronous protocol secure up to $t < n/3$ corruptions and an asynchronous variant secure up to $t < n/5$ corruptions. The asynchronous variant was later improved by Abraham et al. [1] to the optimal corruption threshold $t < n/3$. A sequence of works focused on improving the convergence speed of AA protocols [4, 15, 16].

The works by Mendes and Herlihy [26], and Vaidya and Garg [32] extended the problem of AA to the multidimensional case (see also the journal version [27]), where they showed that for dimension D, the resilience $(D + 1) \cdot t < n$ (resp. $(D + 2) \cdot t < n$) is optimal in synchronous (resp. asynchronous) networks. Other works, such as Függer et al. [19], focus on improving the convergence rate in the multidimensional asynchronous case.

AA has been generalized in different directions as well: a line of works [2, 24, 29] focuses on a variant where the input values are vertices of a finite graph G and the metric distance as well as the convex hull properties hold within the graph G.

The problem of uni-dimensional AA in the hybrid network setting was studied by Ghinea, Liu-Zhang and Wattenhofer in [20], where the protocol simultaneously achieves a resilience t_s for a synchronous network, and t_a for an asynchronous network. The authors show that the trade-off $2t_s + t_a < n$ is achievable and optimal.

Additional related work. Designing protocols that achieve simultaneously security guarantees in both synchronous and asynchronous networks has been a subject that attracted increased attention in the recent years. Indeed, there has been a line of works studying such protocols for Byzantine agreement [5, 12], state-machine replication [6] and also multi-party computation [3, 7, 12]. Another recent work [28] introduced a refinement of hybrid protocols for the problem of Byzantine Fault Tolerance, where they also split the security guarantees (safety and liveness) for different thresholds, see also [22, 25].

1.2 Comparison to previous works

Our result integrates techniques from previous works in a non-trivial manner.

The protocol proceeds in iterations, where at each iteration parties distribute their current values to all other parties. In order to ensure that the parties' values stay within the convex hull of the honest inputs, we follow the approach in [26], which computes a *safe area* of the values received: by intersecting the convex hulls of each $n - t$ subsets of values out of the values received. However, in contrast to their setting which only tolerates $t < n/(D + 2)$ corruptions, our protocol still needs to give guarantees when up to $t_s < n/(D + 1)$ parties may be corrupted and the network is synchronous. In this case, the safe area may be empty. We solve this using the so-called Overlap All-to-All Broadcast primitive, which was initially introduced in [20], in the context of uni-dimensional AA in the hybrid network setting.

Next, in order to prove that the values obtained by the honest parties in each iteration converge, we need to adapt the convergence arguments into the multidimensional setting. It turns out that achieving this requires a non-trivial set of novel combinatorial results (see Lemmas 5.8–5.12), which constitute one of our main technical contributions.

Another technical contribution of our work is to provide a mechanism that enables the honest parties to decide when to output a value, i.e., estimate a sufficient number of iterations. This is achieved by allowing the honest parties to estimate the honest inputs' convex hull, using a mechanism based on techniques initially introduced in [26], but with a novel witness technique that is adapted to the hybrid-network setting. This is in contrast to the protocol in [20], which assumes that some bounds on the honest inputs' range are initially known.

Finally, also note that in contrast to the work in [20] which needs to assume a public-key infrastructure to achieve optimal resilience in the uni-dimensional case, our protocols show that this is not required in the multidimensional case. Intuitively, this is because the necessary condition $n > (D + 1) \cdot t_s$ when $D > 1$ is enforced by the Helly number of $\mathbb{R}^D$ [29], and then, important building blocks, such as reliable broadcast, can be achieved without any setup.

2 MODEL AND DEFINITIONS

We consider a setting of n parties $P_1, P_2, \ldots, P_n$ running a protocol over a network. The parties are pair-wise connected through authenticated channels.

The network may be either synchronous or asynchronous, and the parties are not aware of the type of network in which they are running the protocol. In a synchronous network, every message is delivered within a publicly known amount of time Δ, and the parties have access to synchronized clocks. In an asynchronous network, the messages are only guaranteed to be delivered eventually, and no assumption is made about the clocks.

We consider an adaptive adversary that may corrupt at any point of the protocol's execution at most t_s parties if the network is synchronous, and at most t_a parties if the network is asynchronous. The corrupted parties become Byzantine, meaning that they may deviate arbitrarily from the protocol, and may even be malicious. Additionally, the adversary may schedule the delivery of the messages, with the condition that, if the network is synchronous, the messages are delivered within Δ time.

The messages sent over the network are provided with identification numbers ensuring that the parties can identify which messages correspond to which sub-protocol instances. For simplicity of presentation, we omit these identification numbers.

2.1 Useful Definitions and Notations

The first definition that we require is the euclidean distance between two values in $\mathbb{R}^D$.

Definition 2.1 (Euclidean Distance). For any $v, v' \in \mathbb{R}^D$, the euclidean distance between v and v' is $\delta(v, v') = \sqrt{\sum_{d=1}^{D}(v^d - v'^d)^2}$, where v^d is the projection of v on coordinate $1 \leq d \leq D$.

We use $\delta_{\max}(V) = \max\{\delta(v, v') : v, v' \in V\}$ to denote the diameter of $V \subseteq \mathbb{R}^D$.

Below we provide the definitions of a convex set and the convex hull of a set. Roughly, $V \subseteq \mathbb{R}^D$ is convex if, for any $v, v' \in V$, the segment between v and v' is also included in V.

Definition 2.2 (Convex Set). A set of values $V \subseteq \mathbb{R}^D$ is convex if for any $v_1, v_2, \ldots, v_k \in V$ and $\lambda_1, \lambda_2, \ldots, \lambda_k \geq 0$ such that $\sum_{i=1}^{k} \lambda_i = 1$, it holds that $\sum_{i=1}^{k} \lambda_i v_i \in V$.

Definition 2.3 (Convex Hull). The convex hull of $V \subseteq \mathbb{R}^D$, denoted by convex(V), is the smallest convex set V' such that $V \subseteq V$.

To avoid working with multisets instead of sets in our D-AA protocol, we will consider sets of value-party pairs $\mathcal{M} \subseteq \mathbb{R}^D \times \{P_1, P_2, \ldots P_n\}$, and we define val$(\mathcal{M}) = \{\{v : (v, P) \in \mathcal{M}\}\}$ (as a multiset). We also extend the definition of the convex hull to such sets: convex$(\mathcal{M})$ = convex(val$(\mathcal{M})$).

2.2 D-dimensional Approximate Agreement

We include the definition of D-AA, as presented in [26, 32].

Definition 2.4. (D-dimensional Approximate Agreement) Let Π be a protocol where initially each party P holds an input in $\mathbb{R}^D$ and may output $v_P \in \mathbb{R}^D$. We consider the following properties:

- t-Validity: If at most t of the parties involved are corrupted, and an honest party P obtains output v_P, then v_P is within the convex hull of the honest inputs.
- (t, ε)-Agreement: If at most t of the parties involved are corrupted, and two honest parties P and P' obtain outputs v_P and $v_{P'}$, then $\delta(v_P, v'_P) \leq \varepsilon$.
- t-Liveness: If at most t of the parties involved are corrupted, then every honest party eventually obtains an output v_P.

Then, Π is a t-secure D-dimensional Approximate Agreement protocol if it achieves t-Validity (t, ε)-Agreement for any given $\varepsilon > 0$, and t-Liveness.

3 LOWER BOUNDS ON RESILIENCE

In the synchronous setting, Vaidya and Garg [32] show that $n > (D + 1) \cdot t_s$ parties are necessary to achieve D-AA for $\varepsilon = 0$. We state their argument for any $\varepsilon > 0$.

Theorem 3.1. *There is no synchronous D-AA protocol that is t_s-secure when $n \leq (D + 1) \cdot t_s$.*

Proof. We assume that $n = (D + 1) \cdot t_s$, and that there is a protocol Π achieving D-AA in this setting. We fix an arbitrary $\varepsilon > 0$, and we partition the set of n parties into $D + 1$ sets of size t_s each: $S_0, S_1, \ldots S_D$. For $1 \leq d \leq D$, the parties in set S_d have input e_d such that $e_d = \varepsilon$ and all other components are 0. The parties in set S_0 have input $e_0 = \mathbf{0}^D$.

If the parties in set S_d are honest, they cannot distinguish between the following scenarios:

Scenario d.i, where $i \neq d$: The t_s parties in S_i are corrupted, and execute protocol Π correctly with input e_i. Then, since Π achieves t_s-Validity and t_s-Liveness, the honest parties in S_d obtain outputs in convex$(\{e_j : i \neq j\})$, as shown in Figure 1.

Then, the output of any honest party in S_d is in $\bigcap_{i \neq d}$ convex$(\{e_j : i \neq j\}) = \{e_d\}$, meaning that each honest party outputs its own input in Π. The diameter of the output set is $\delta_{\max}(\{e_d : 0 \leq d \leq D\}) = \sqrt{\varepsilon^2 + \varepsilon^2} = \varepsilon\sqrt{2} > \varepsilon$, hence Π does not achieve ε-Agreement. □

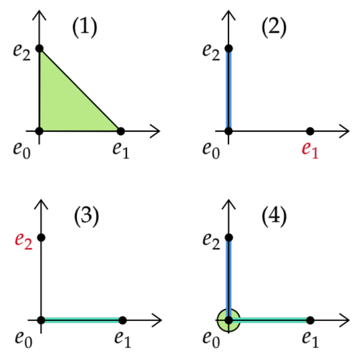

Figure 1: This figure shows an example $D = 2$, $t_s = 1$ and $n = (D + 1) \cdot t_s = 3$. The first picture shows the convex hull of the honest inputs e_0, e_1, e_2, as described in Theorem 3.1. The second and third pictures highlight the convex hull of the honest inputs in Scenarios 0.1 (where the party with input e_1 is corrupted) and resp. 0.2 (where the party with input e_2 is corrupted). The last figure shows that the intersection of the valid convex hulls is e_0.

The necessary condition for achieving D-AA in asynchronous networks was proven in previous works [26, 32] by using a similar argument to that of Theorem 3.1. The parties are split into $D + 2$ sets $S_0, S_1, \ldots, S_{D+1}$ of t_a parties each, and the parties in set S_d with $0 \leq d \leq D$ have inputs e_d as described in the proof of Theorem 3.1. Parties in S_{D+1} do not send any messages. Since the network is asynchronous, the honest parties cannot decide whether the parties in S_{D+1} are corrupted, or the parties in S_{D+1} are honest, but their messages are delayed, and any other t_a parties are corrupted. As they have to output a value, the honest parties in S_d with $0 \leq d \leq D$ simply output e_d, which breaks ε-Agreement.

Theorem 3.2. *There is no asynchronous D-AA protocol that is t_a-secure when $n \leq (D + 2) \cdot t_a$.*

4 COMMUNICATION PRIMITIVES

In this section, we provide formal definitions and constructions for the communication primitives we use to achieve D-AA.

4.1 Reliable Broadcast

Below we include the definition of Reliable Broadcast [9, 10]. Similarly to [20], we make the concrete running time and simultaneous termination properties explicit, as they will be used to keep the honest parties synchronized if the network is synchronous.

Definition 4.1. Let Π be a protocol where a designated party S (called the sender) holds a value v_S, and every party P may output a value v_P. We consider the following properties, where t denotes the number of corrupted parties involved:

- t-Validity: If an honest party obtains an output v_P, then $v_P = v_S$.
- t-Consistency: If P and P' are honest and output v_P and resp. $v_{P'}$, then $v_P = v_{P'}$.
- (t, c)-Honest Liveness: If S is honest, parties obtain outputs eventually. In addition,if the network is synchronous and the parties start executing the protocol at the same time τ, every honest party obtains output by time $\tau + c \cdot \Delta$.
- (t, c')-Conditional Liveness: If an honest party obtains output at time τ, then all honest parties obtain outputs eventually. In addition, if the network is synchronous and the honest parties start executing the protocol at the same time τ, then all honest parties obtain output by time $\tau + c' \cdot \Delta$.

We say that Π is a (t, c, c')-secure Reliable Broadcast protocol if it achieves t-Validity and t-Consistency even when not all honest parties join the execution of the protocol, (t, c)-Honest Liveness, and (t, c')-Conditional Liveness.

The following theorem is achieved from Bracha's Reliable Broadcast protocol [9] in a straightforward manner (see Appendix 6.1).

Theorem 4.2. *There is a (t, c_{rBC}, c'_{rBC})-secure Reliable Broadcast protocol Π_{rBC} for $n > 3t$, $c_{rBC} = 3$, and $c'_{rBC} = 2$.*

4.2 Overlap All-to-All Broadcast

Overlap All-to-All Broadcast [20] allows every party P to distribute its input v_P to every party. Each party obtains a set of value-party pairs $\mathcal{M}_P \subseteq I \times \{P_1, P_2, \ldots, P_n\}$, where I denotes the input space. Overlap All-to-All Broadcast provides some strong properties on the honest parties' outputs. We list these properties below, parameterized by t, denoting the number of corrupted parties involved.

Firstly, honest parties' inputs are distributed reliably:

- t-Validity: If an honest party P outputs $\mathcal{M}_P$ with $(v, P') \in \mathcal{M}_P$ for an honest party P', then $v = v_{P'}$.
- t-Consistency: If two honest parties P and P' output $\mathcal{M}_P$ and $\mathcal{M}_{P'}$ respectively, and $(v, P'') \in \mathcal{M}_P$ and $(v', P'') \in \mathcal{M}_{P'}$, then $v = v'$.

When the network is synchronous, we expect that honest values are always included in the output sets, since they are received within a known amount of time. In the asynchronous model, we only require that honest parties receive a minimum number of common values.

- t-Synchronized Overlap: If the network is synchronous and an honest party P obtains output $\mathcal{M}_P$, then $(v_{P'}, P') \in \mathcal{M}$ for every honest party P'.
- (T, t)-Overlap: If two honest parties P and P' obtain outputs $\mathcal{M}_P$ and $\mathcal{M}_{P'}$, then $|\mathcal{M}_P \cap \mathcal{M}_{P'}| \geq n - T$.

Then, if the network is synchronous, we expect the honest parties to obtain outputs within a known amount of time, so they can synchronize for further steps in a larger protocol.

- (t, c)-Synchronized Liveness: If the network is synchronous and the honest parties start executing the protocol at time τ, then all honest parties obtain output by time $\tau + c \cdot \Delta$.
- t-Liveness: All honest parties obtain output eventually.

We provide the definition of Overlap All-to-All Broadcast below.

Definition 4.3 (Overlap All-to-All Broadcast). Let Π be a protocol where every party P holds an input v_P and may output a set of value-sender pairs $\mathcal{M}_P$. Π is a (t_s, t_a, c)-secure Overlap All-to-All Broadcast protocol if:

- when running in a synchronous network, where the honest parties start executing Π simultaneously, it achieves: t_s-Validity, t_s-Consistency, t_s-Synchronized Overlap and (t_s, c)-Synchronized Liveness.
- when it runs in an asynchronous network, it achieves t_a-Liveness. In addition, even if not all honest parties join the execution of the protocol, t_a-Validity, t_a-Consistency, and (t_s, t_a)-Overlap must hold.

Overlap All-to-All Broadcast can be achieved by following the outline of [20], which makes use of a so-called *witness technique* [1]. At a high level, the protocol proceeds as follows:

The parties first share their inputs via Π_{rBC}. If the network is synchronous, the values sent by the honest parties are received within $c_{\text{rBC}} \cdot \Delta$ time, hence the honest parties collect at least the $n - t_s$ honest values by time $c_{\text{rBC}} \cdot \Delta$. Every value received is collected in a set of value-sender pairs $\mathcal{M}$.

After $c_{\text{rBC}} \cdot \Delta$ time has passed, if P has collected at least $n - t_s$ values via Π_{rBC}, P reports the values it has received so far by sending its set $\mathcal{M}$ to all the other parties. If the network is synchronous, this set is received within Δ time.

When receiving $\mathcal{M}_{P'}$ from P', P checks whether it has received every value in $\mathcal{M}_{P'}$ via Π_{rBC}. If this is the case, P marks P' as a witness. Note that any value in $\mathcal{M}_{P'}$ is eventually received by every party, and, if the network is synchronous, within $c'_{\text{rBC}} > 1$ communication rounds, hence by time $(c_{\text{rBC}} + c'_{\text{rBC}}) \cdot \Delta$. Therefore, P' can become a witness for every party and, if the network is synchronous, at time $(c_{\text{rBC}} + c'_{\text{rBC}}) \cdot \Delta$.

Then, at time $(c_{\text{rBC}} + c'_{\text{rBC}}) \cdot \Delta$, an honest party checks if has marked $n - t_s$ parties as witnesses. If this is the case, it can safely output its set $\mathcal{M}$. This ensures that every two honest parties have an honest witness in common even if the network is asynchronous, and hence at least $n - t_s$ common values in their output sets. We formally present the code below.

Protocol Π_{oBC}

Code for party P with input v

```
1: $\tau_{\text{start}} := \tau_{\text{now}}$
2: $\mathcal{M} := \emptyset$, $W := \emptyset$
3: Invoke $\Pi_{\text{rBC}}$ with input $v$
4: Upon receiving a value $v$ from $P'$ via $\Pi_{\text{rBC}}$, add $(v', P')$ to $\mathcal{M}$
5: When $\tau_{\text{now}} \geq \tau_{\text{start}} + c_{\text{rBC}} \cdot \Delta$ and $|\mathcal{M}| \geq n - t_s$:
6:     Send $\mathcal{M}$ to all the parties
7: When receiving $\mathcal{M}_{P'}$ from $P'$ such that $|\mathcal{M}_{P'}| \geq n - t_s$ and
   $\mathcal{M}_{P'} \subseteq \mathcal{M}$:
8:     Add $P'$ to $W$
9: When $\tau_{\text{now}} \geq \tau_{\text{start}} + (c_{\text{rBC}} + c'_{\text{rBC}}) \cdot \Delta$ and $|W| \geq n - t_s$:
        Output $\mathcal{M}$
```

The proof of the following theorem is enclosed in the appendix.

THEOREM 4.4. *Π_{oBC} is a (t_s, t_a, c_{oBC})-secure Overlap All-to-All Broadcast protocol for $c_{oBC} = c_{rBC} + c'_{rBC}$.*

5 ALGORITHM

Many AA protocols in the literature, whether they work with scalars [1, 4, 13, 20], multiple dimensions [26, 32], or even graphs or lattices [29], follow a similar structure: they operate in multiple iterations. In each iteration, the parties first distribute their current values. Then, the parties *discard the outliers* out of the values received, and compute a new value based on the values remaining.

To achieve AA, each iteration should satisfy a few properties. Firstly, the values obtained by the honest parties should be in the convex hull of the values they started the iteration with, to ensure Validity. Secondly, the values obtained by the honest parties should get closer, to ensure that ε-Agreement is eventually achieved. In our model, or even in the purely synchronous model, it is also essential to maintain the honest parties synchronized, similarly to [20]. These properties roughly ensure that, after a sufficient number of iterations, D-AA is achieved. For our AA protocol, we implement the steps taken in a single iteration in a subroutine called $\Pi_{\text{AA-it}}$.

It is also important to define the *sufficient* number of iterations. While some known AA protocols [20, 29] predefine this number (by assuming that some bounds on the input space are known), others allow the parties to estimate the range of honest inputs on their own [1, 26]. Our algorithm will extend this latter approach to the hybrid network model.

Then, our protocol proceeds as follows: the parties first run a subroutine Π_{init}, which enables each honest party to obtain a value v_0 (within the convex hull of the honest inputs) and an estimation T (computed accordingly to the convergence guarantees of $\Pi_{\text{AA-it}}$). Π_{init} ensures that T iterations of $\Pi_{\text{AA-it}}$, starting from the values v_0 it provides instead of the honest parties' inputs, are enough to achieve D-AA. This is the case even if T is the smallest honest estimation.

Hence, honest parties join the first iteration using v_0 as their initial values. In each iteration it, they run the subroutine $\Pi_{\text{AA-it}}$ with input $v_{\text{it}-1}$, defined below, and obtain a new value v_{it}.

When a party P reaches iteration $\text{it} = T$, matching its own estimation for when it is safe to output a value, it sends a halt message for iteration it. P outputs when it receives $t_s + 1$ halting messages for previous iterations, hence at least one honest halting message.

We formally present the code of our protocol below. The constant $c_{\text{AA-it}} = 5$ represents the number of communication rounds required by the subroutine $\Pi_{\text{AA-it}}$ when running in a synchronous network.

Protocol Π_{AA}

Code for party P with input v

```
1:  Run $\Pi_{\text{init}}$ with input $v$ and obtain $(T, v_0)$.
2:  it = 1
3:  while true do
4:      Join $\Pi_{\text{AA-it}}$ with input $v_{\text{it}-1}$
5:      After at least $c_{\text{AA-it}} \cdot \Delta$ time has passed within this iteration:
6:          Upon obtaining output $v_{\text{it}}$ from $\Pi_{\text{AA-it}}$:
7:              If it = $T$, send (halt, it) via $\Pi_{\text{rBC}}$ to all parties
8:          If $t_s + 1$ messages (halt, it') with it' < it were received:
9:              $\text{it}_h := (t_s+1)$-th smallest iteration number received
10:             Output $v_{\text{it}_h}$ and break
11:         it = it + 1
12: end while
```

We note that using this estimation-based approach for a sufficient number of iterations may force honest parties to output values at different times. That is, once the first honest party outputs a value, it does not participate in any further iterations. In our protocol, we make sure that this is not an issue. If the network is synchronous, we will show that $\Pi_{\text{AA-it}}$ does not need to offer any guarantees on these further iterations, since the honest parties end up with outputs obtained when every honest party was still participating. If the network is asynchronous, $\Pi_{\text{AA-it}}$ will still ensure that, if honest parties obtain new values, they will be in line with the sufficient requirements for obtaining D-AA.

In the following, we describe each of the pieces of our protocol Π_{AA} in detail: we first describe $\Pi_{\text{AA-it}}$, which focuses on a single iteration, and then Π_{init}. Afterwards, we prove that Π_{AA} indeed achieves (t_s, t_a)-secure D-AA.

Iterations. We describe the subroutine $\Pi_{\text{AA-it}}$, which is executed in each iteration of our D-AA protocol.

Honest parties first distribute their values via Π_{oBC}, similarly to [20]. To ensure validity, the parties *discard the outliers* out of the values received. In the pure asynchronous unidimensional setting (where $t_s = t_a = t$), discarding the lowest t and the highest t values received in each iteration is enough [1, 4, 13]: there is at least one value remaining, and any corrupted value remaining is in the range of honest values. Then, the parties compute the new values by taking the average between the lowest and the highest values remaining. The multidimensional case generalizes this approach: if a party obtains a set $\mathcal{M}$ of value-sender pairs of size at least $n-t$ via Π_{oBC} in some iteration, then at least $|\mathcal{M}| - t$ of the values in $\mathcal{M}$ are honest. Hence, one can obtain a *safe area* by computing the convex hull of each subset of $|\mathcal{M}| - t$ values, and then by intersecting these convex hulls, as shown in Figure 2. This intersection is guaranteed to be included in the convex hull of the honest values in $\mathcal{M}$. We formally define the *safe area* [26] below.

Definition 5.1 (Safe Area). Let $\mathcal{M}$ denote a set of value-sender pairs with values in $\mathbb{R}^D$. For a given t, the safe area of $\mathcal{M}$ is

$$\text{safe}_t(\mathcal{M}) = \bigcap_{M \in \text{restrict}_t(\mathcal{M})} \text{convex}(M),$$

where $\text{restrict}_t(\mathcal{M}) = \{M \subseteq \mathcal{M} : |M| = |\mathcal{M}| - t\}$.

In the pure asynchronous model, $\text{safe}_t(\mathcal{M}) \neq \emptyset$ even when $\mathcal{M}$ only contains $n - t$ values. In our model, however, we are facing the same challenge as [20]: when $n > (D+1) \cdot t_s + t_a$, it is possible that $\text{safe}_{t_s}(\mathcal{M}) = \emptyset$. As an example, consider the two-dimensional case, and let $n = 4$, $t_s = 1$ and $t_a = 0$. Assume that the network is synchronous and the Byzantine party does not distribute its value. Let the values of the three honest parties be $(0, 0)$, $(0, 1)$ and $(1, 0)$.

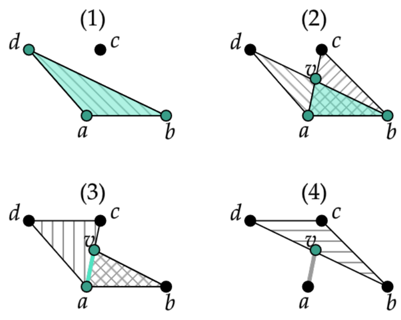

Figure 2: This figure shows an example for computing the safe area of four points a, b, c, d, given $t = 1$. We intersect the convex hulls of each subset of three points. In the first picture, we highlight the convex hull of a, b, and d. In the second image, we intersect convex($\{a, b, d\}$) with convex($\{a, b, c\}$) and we highlight their intersection convex($\{a, v, c\}$). Then, we add convex($\{a, c, d\}$) to the intersection and we obtain the highlighted segment $[a, v]$. Finally, when intersecting this segment with convex($\{b, d, c\}$), we obtain the safe area, which contains a single point v. Regardless of which of the four points is held by a corrupted party, v is included in the convex hull of the three honest points.

Then, an honest party that only receives the three honest values would compute its safe area as

$$\begin{aligned}\mathsf{safe}_1(\{(0,0),(0,1),(1,0)\}) = \mathsf{convex}(\{(0,0),(0,1)\}) \cap\\ \mathsf{convex}(\{(0,0),(1,0)\}) \cap\\ \mathsf{convex}(\{(0,1),(1,0)\}) = \emptyset.\end{aligned}$$

To overcome this issue, hence to ensure that the parties obtain non-empty safe areas, we use an approach similar to [20]. If the network is synchronous, and all honest parties start executing the iteration at the same time, then Π_{oBC} guarantees every honest value is added to the output set $\mathcal{M}$. This means that, if $\mathcal{M}$ has size $n - t_s + k$, where $0 \le k \le t_s$, only k of these values are sent by corrupted parties. If the network is asynchronous, Π_{oBC} still guarantees that $|\mathcal{M}| \ge n - t_s$. In this setting however, Π_{oBC} does not ensure that all honest values are received. Instead, at most t_a values in $\mathcal{M}$ are sent by corrupted parties. Then, to ensure that the newly computed values are within the convex hull of honest parties' inputs in $\Pi_{\text{AA-it}}$, we compute the safe area as $S := \mathsf{safe}_{\max(k,t_a)}(\mathcal{M})$. This way, in the previous example, the safe area obtained by the honest parties is convex($\{(0,0),(0,1),(1,0)\}$).

The new values are then computed as in [18]: each party picks the two points a and b realizing the diameter of its safe area. The two points are chosen deterministically (i.e. $\mathbb{R}^D$ is totally ordered, so, if there are multiple options, we can simply pick the pair with the lowest a, and, in case of equality, with the lowest b). Then, it computes its new value by taking their midpoint, which is guaranteed to be in the safe area, since it is convex.

With the help of a technical lemma, we will show that the safe areas obtained by any two honest parties intersect. This will enable us to prove that the values obtained by the honest parties in each iteration via $\Pi_{\text{AA-it}}$ get closer by a convergence factor of $\sqrt{\frac{7}{8}}$.

We formally present the code of $\Pi_{\text{AA-it}}$ below.

Protocol $\Pi_{\text{AA-it}}$

Code for party P with input v

```
1: Join $\Pi_{\mathrm{oBC}}$ with input $v$
2: Upon obtaining output $\mathcal{M}$ in $\Pi_{\mathrm{oBC}}$:
3:     $k := |\mathcal{M}| - (n - t_s)$
4:     $S := \mathsf{safe}_{\max(k,t_a)}(\mathcal{M})$
5:     $a, b := \mathrm{argmax}_{a,b \in S \times S}(\delta(a, b))$
6:     Output $(a + b)/2$
```

Estimating a sufficient number of iterations. We now present our subroutine Π_{init}, which allows the honest parties to obtain estimations on the sufficient number of iterations for Π_{AA}.

Similarly to Π_{oBC}, our protocol Π_{init} employs the *witness technique* [1]. Parties distribute their inputs via Π_{rBC}, and add any value received and its sender to a set of value-sender pairs $\mathcal{M}$. When at least $c_{\mathrm{rBC}} \cdot \Delta$ time has passed (and hence every honest value was received if the network is synchronous), and when $|\mathcal{M}| \ge n - t_s$ (since at most t_s parties are corrupted), the parties reliably broadcast their set $\mathcal{M}$. This step is almost identical to reporting values in Π_{oBC} – the main difference so far is that the sets of reported values are sent reliably and not through best-effort broadcast.

Then, as in Π_{oBC}, if P receives a set of reported values $\mathcal{M}_{P'}$ from P' such that each of the reported values was also received by P ($\mathcal{M}_{P'} \subseteq \mathcal{M}$), P marks P' as a witness.

At this point, Π_{init} starts to differ from Π_{oBC}. Similarly to the estimation mechanism of [1, 26], when P marks P' as a witness, P computes an estimated new input value for P', based on $\mathcal{M}_{P'}$. We ensure that these estimations are consistent – if a third party also marks P' as a witness, it estimate the same value for P'. This is ensured since the set of reports is sent via Π_{rBC}, and the estimated values are computed deterministically.

We also need to ensure that these estimated values are in the honest inputs' convex hull. Hence, the estimated values are computed identically to new values in $\Pi_{\text{AA-it}}$.

At this point, since the converge factor of $\Pi_{\text{AA-it}}$ is known, each party could derive a sufficient number of iterations T based on the set I_e containing the estimations it has obtained for its witnesses. This is the case in the pure asynchronous (when $t_s = t_a = t$) D-AA protocol of [26]. Note that the number of iterations T is sufficient for the estimated values, and not necessarily for the true honest inputs. Then, each honest party P should join the first iteration with a value that is in convex(I'_e) of any honest P'. If the network is asynchronous, it could be that P' has not marked P as a witness. However, the pure asynchronous setting gives a strong guarantee on the sets of witnesses W_P and $W_{P'}$ obtained by P and P' respectively: $W_P \setminus W_{P'} \le t$ (Lemma 4.14 of [26]). Then, P computes its own estimation by picking a point in the safe area of its own set I_e, which is in convex(I'_e).

Our setting, on the other hand, comes with a different challenge. If the network is asynchronous, and P and P' obtain $n - t_s$ witnesses each, meaning that $W_P \setminus W_{P'} \le t_s$. Then, using the same approach would imply that P has to pick its new initial value from $\mathrm{safe}_{t_s}(I_e)$, which may be an empty set. We overcome this issue by ensuring that every two honest parties have $n - t_s$ common estimations. This is already ensured in the synchronous setting, where parties wait long enough to mark each honest party as a witness. To obtain $n - t_s$ common estimations in the asynchronous setting as well, we introduce double-witnesses: each party sends its set of witnesses W to all the parties. If P receives $W_{P'}$ from P' such that $W_{P'} \subseteq W$, it marks P' as a double-witness. When P gathers $n - t_s$ double-witnesses, it has $n - t_s$ common estimations with every party. Then, P picks its new value v from the safe area of its set of estimations I_e. Since now P and any honest party P' have $n - t_s$ estimated values in common, P's safe area is included in the convex hull of these common estimations. This is enough to ensure that the estimated number of iterations is indeed sufficient.

We formally present the code of our subroutine Π_{init} below.

Protocol Π_{init}

Code for party P with input v

```
1: $\tau_{\mathsf{start}} := \tau_{\mathsf{now}}$; $\mathcal{M} = \emptyset$; $I_e = \emptyset$; $W := \emptyset$, $W_2 := \emptyset$
2: Send $v$ to every party via $\Pi_{\mathsf{rBC}}$
3: Upon receiving a value $v$ from $P'$ via $\Pi_{\mathsf{rBC}}$, add $(v', P')$ to $\mathcal{M}$
4: When $\tau_{\mathsf{now}} \geq \tau_{\mathsf{start}} + c_{\mathsf{rBC}} \cdot \Delta$ and $|\mathcal{M}| \geq n - t_s$:
5:     Send $\mathcal{M}$ to all the parties via $\Pi_{\mathsf{rBC}}$
6: When receiving $\mathcal{M}_{P'}$ from $P'$ such that $|\mathcal{M}_{P'}| \geq n - t_s$ and $\mathcal{M}_{P'} \subseteq \mathcal{M}$:
7:     $k_{P'} := |\mathcal{M}_{P'}| - (n - t_s)$
8:     $S_{P'} := \mathsf{safe}_{\max(t_a, k_{P'})}(\mathcal{M}_{P'})$
9:     $a, b := \mathrm{argmax}_{a,b \in S_{P'} \times S_{P'}}(\delta(a, b))$
10:    $v_{P'} := (a + b)/2$
11:    Add $(v_{P'}, P')$ to $I_e$ and $P'$ to $W$
12: When $\tau_{\mathsf{now}} \geq \tau_{\mathsf{start}} + 2c_{\mathsf{rBC}} \cdot \Delta$ and $|W| \geq n - t_s$:
13:    Send $W$ to every party
14: When receiving $W_{P'}$ from $P'$ such that $|W_{P'}| \geq n - t_s$ and $W_{P'} \subseteq W$:
15:    Add $P'$ to $W_2$
16: When $\tau_{\mathsf{now}} \geq \tau_{\mathsf{start}} + (2c_{\mathsf{rBC}} + c_{\mathsf{rBC}'}) \cdot \Delta$ and $|W_2| \geq n - t_s$:
17:    $k := |W| - (n - t_s)$
18:    $S := \mathsf{safe}_{\max(t_a, k)}(I_e)$
19:    $a, b := \mathrm{argmax}_{a,b \in S \times S}(\delta(a, b))$
20:    $v_{P'} := (a + b)/2$
21:    $T := \lceil \log_{\sqrt{7/8}}(\varepsilon / \delta_{\max}(I_e)) \rceil$
22:    Output $(T, v_0)$
```

5.1 Properties of the safe area

In order to prove that our protocol Π_{AA} indeed achieves D-AA, we need to explore the properties of the safe area.

Helly's Theorem. Since the safe area is computed as an intersection of convex hulls, our proofs heavily use the theorem included below, which comes from discrete geometry [11].

Theorem 5.2 (Helly's Theorem). *Consider any finite collection of closed convex sets $S = \{S_1, S_2, \dots, S_m\}$ on $\mathbb{R}^D$, with $m \geq D+1$. If the intersection of any $D + 1$ members of S is non-empty, $\bigcap_{S_i \in S} S_i \neq \emptyset$.*

Note that, in order to use Theorem 5.2 to prove that a collection of closed convex sets S indeed has a non-empty intersection, we need to ensure that S has size at least $D + 1$. Lemma 5.3, presented below, provides us with this guarantee in our proofs. For completeness, we have included its proof in the appendix.

Lemma 5.3. *Let $\mathcal{M}$ denote a set of $n - t_s + k$ value-party pairs, where $k \leq t_s$. Then, $|\mathit{restrict}_{\max(k,t_a)}(\mathcal{M})| \geq D + 1$.*

Lemma 5.4, presented below, is a more general form of one of the technical lemmas of [26] (Lemma 3.5), and gives us some lower bound on the size of the intersection of convex sets used to compute safe areas. It will help us show the main sufficient condition in Theorem 5.2 holds when proving that an honest party's safe area is non-empty. The additional set Y in the lemma's statement will be useful when showing that the honest parties' safe areas intersect, which will later imply that their values get closer. We include the proof of this lemma in the appendix.

Lemma 5.4. *Assume that X is a finite set, $m \leq D + 1$, and let $X_1, \dots, X_m \in \mathit{restrict}_t(X)$. Then, $|\bigcap_{i=1}^{m} X_i| \geq |X| - mt$.*

In addition, given a finite set Y, $|X \cap Y \cap \bigcap_{i=1}^{m} X_i| \geq |X \cap Y| - mt$.

Valid values. We can now use Theorem 5.2 to show that the safe areas obtained by the honest parties are non-empty.

Lemma 5.5. *Let $\mathcal{M}$ denote a set of $n - t_s + k$ value-sender pairs with values in $\mathbb{R}^D$, where $0 \leq k \leq t_s$. Then, $\mathit{safe}_{\max(k,t_a)}(\mathcal{M}) \neq \emptyset$.*

Proof. Since $|\mathrm{restrict}_{\max(k,t_a)}(\mathcal{M})| \geq D + 1$ by Lemma 5.3, we only need to show that any $D+1$ members of $\mathrm{restrict}_{\max(k,t_a)}(\mathcal{M})$, denoted by $M_1, M_2, \dots, M_{D+1}$, intersect.

Lemma 5.4 shows that $|\bigcap_{i=1}^{D+1} M_i| \geq n - t_s + k - (D+1) \cdot \max(k, t_a)$.

If $k \geq t_a$, we obtain that $|\bigcap_{i=1}^{D+1} M_i| \geq n - (D + 1) \cdot t_s \geq 1$. Otherwise, if $k < t_a$, we obtain that $|\bigcap_{i=1}^{D+1} M_i| \geq n - t_s + k - (D + 1) \cdot t_a \geq 1$, since $t_a \leq t_s$. Therefore, in both cases, $\bigcap_{i=1}^{D+1} M_i \neq \emptyset$. This result carries to the intersection of the convex hulls, and hence we can apply Theorem 5.2 and obtain that $\mathrm{safe}_{\max(k,t_a)}(\mathcal{M}) \neq \emptyset$. □

Now that the safe areas are guaranteed to be non-empty, we ensure that the values v computed by the honest parties are well-defined. The result below follows immediately from the convexity of the safe area.

Lemma 5.6. *Assume $\mathcal{M}$ is a set of $n - t_s + k$ value-party pairs, with $0 \leq k \leq t_s$, and let $S = \mathit{safe}_{\max(k,t_a)}(\mathcal{M})$ denote its safe area. Then, if $a, b = \mathit{argmax}_{a,b \in S \times S}(\delta(a, b))$, $v = (a + b)/2$ is well-defined and is in S.*

Finally, Lemma 5.7 will help us show that the values obtained by the honest parties are within the convex hull of their inputs.

Lemma 5.7. *Assume $\mathcal{M}$ is a set of $n - t_s + k$ value-party pairs, with $0 \leq k \leq t_s$, and let $I \subseteq \mathcal{M}$ denote an arbitrary subset of size $n - t_s + k - \max(k, t_a)$. Then, $\mathit{safe}_{\max(k,t_a)}(\mathcal{M}) \subseteq \mathit{convex}(I)$.*

Proof. Since $I \in \mathrm{restrict}_{\max(k,t_a)}(\mathcal{M})$, $\mathrm{convex}(I)$ is one of the members in the intersection defining $\mathrm{safe}_{\max(k,t_a)}(\mathcal{M})$. Therefore, $\mathrm{safe}_{\max(k,t_a)}(\mathcal{M}) \subseteq \mathrm{convex}(I)$.

□

Intersection of safe areas. We now focus on showing that the safe areas obtained by every pair of honest parties in each iteration intersect. This will be a sufficient argument when proving that the values obtained by the honest parties get closer. We note that this argument provides the same challenge as in the 1-dimensional case in the hybrid-network model [20] when compared to the pure asynchronous case [1]: in the pure asynchronous case, where $t = t_s = t_a$, every two honest parties receive at least $n - t$ common values. Then, the safe area of the common values is non-empty, and is included in the safe areas of the two honest parties. In our model, this argument does not necessarily hold, We show that the two honest parties' safe areas still intersect, using a different argument, based on Theorem 5.2.

Lemma 5.8. *Let $\mathcal{M}_1, \mathcal{M}_2$ denote two sets of value-sender pairs such that $|\mathcal{M}_1 \cup \mathcal{M}_2| \leq n$ and $|\mathcal{M}_1 \cap \mathcal{M}_2| \geq n - t_s$. Then,*

$$\textit{safe}_{\max(k_1,t_a)}(\mathcal{M}_1) \cap \textit{safe}_{\max(k_2,t_a)}(\mathcal{M}_2) \neq \emptyset,$$

where $k_1 = |M_1| - (n - t_s)$ and $k_2 = |M_2| - (n - t_s)$.

The proof of this lemma is split into three cases. The case when both $k_1 \geq t_s$ and $k_2 \geq t_s$ hold is covered by Lemma 5.9, where we show that the safe area of $\mathcal{M}_1 \cup \mathcal{M}_2$ is non-empty and included in both $\text{safe}_{k_1}(\mathcal{M}_1)$ and $\text{safe}_{k_2}(\mathcal{M}_2)$. Although the union contains more value-party pairs than $\mathcal{M}_1$ and $\mathcal{M}_2$, its safe area is intuitively smaller than $\text{safe}_{k_1}(\mathcal{M}_1)$ and $\text{safe}_{k_2}(\mathcal{M}_2)$. This is because the safe area of $\mathcal{M}_1 \cup \mathcal{M}_2$ takes into account in its intersection at least every convex hull considered in $\text{safe}_{k_1}(\mathcal{M}_1)$, and every convex hull considered in $\text{safe}_{k_2}(\mathcal{M}_2)$.

Afterwards, Lemma 5.11 covers the case when both $k_1 < t_a$ and $k_2 < t_a$. This case is similar to the convergence proofs for protocols achieving purely asynchronous D-AA [26]. Lemma 5.11 shows that the $\text{safe}_{t_a}(\mathcal{M}_1 \cap \mathcal{M}_2)$ is non-empty and included in both $\text{safe}_{t_a}(\mathcal{M}_1) \cap \text{safe}_{t_a}(\mathcal{M}_2)$.

The last case, when $k_1 < t_a \leq k_2$ is more difficult, and we cover it with the help of Lemma 5.12. In this case, Lemma 5.9 shows that the safe area of $\mathcal{M}_1 \cup \mathcal{M}_2$ is included in $\text{safe}_{k_2}(\mathcal{M}_2)$. Then, Lemma 5.12 guarantees that the safe area of $\mathcal{M}_1 \cup \mathcal{M}_2$ and $\text{safe}_{t_a}(\mathcal{M}_1)$ have a non-empty intersection, which concludes the proof of Lemma 5.8. We provide the formal statements and proofs of Lemmas 5.9, 5.11, and 5.12 below.

Lemma 5.9. *Let $\mathcal{M}_1, \mathcal{M}_2$ denote two sets of value-sender pairs such that $|\mathcal{M}_1 \cup \mathcal{M}_2| \leq n$, and $|\mathcal{M}_1 \cap \mathcal{M}_2| \geq n - t_s$. If $k_1 = |\mathcal{M}_1| - (n - t_s) \geq t_a$, then, $\textit{safe}_{k_1}(\mathcal{M}_1) \supseteq \textit{safe}_{k_\cup}(\mathcal{M}_1 \cup \mathcal{M}_2) \neq \emptyset$, where $k_\cup = |\mathcal{M}_1 \cup \mathcal{M}_2| - (n - t_s)$.*

Proof. Note that $t_a \leq k_\cup \leq t_s$. Lemma 5.5 immediately implies that $\text{safe}_{k_\cup}(\mathcal{M}_1 \cup \mathcal{M}_2) \neq \emptyset$.

Then, note that $\text{restrict}_{k_1}(\mathcal{M}_1) = \{M \subseteq \mathcal{M}_1 : |M| = n - t_s\} \subseteq \{M \subseteq \mathcal{M}_1 \cup \mathcal{M}_2 : |M| = n - t_s\} = \text{restrict}_{k_\cup}(\mathcal{M}_1 \cup \mathcal{M}_2)$. It follows that $\text{safe}_{k_1}(\mathcal{M}_1) \supseteq \text{safe}_{k_\cup}(\mathcal{M}_1 \cup \mathcal{M}_2)$. □

In the second case, we make use of a technical property which follows directly from [26]. We include its proof in the appendix.

Lemma 5.10. *Let m denote a value-party pair, and $\mathcal{M}$ a set of value-party pairs. Then, $\textit{safe}_t(\mathcal{M}) \subseteq \textit{safe}_t(\mathcal{M} \cup \{m\})$.*

Lemma 5.11. *Let $\mathcal{M}_1, \mathcal{M}_2$ denote two sets of value-party pairs such that $|\mathcal{M}_1 \cup \mathcal{M}_2| \leq n$, and $|\mathcal{M}_1 \cap \mathcal{M}_2| \geq n - t_s$. If $|\mathcal{M}_1| - (n - t_s) \leq t_a$, then, $\textit{safe}_{t_a}(\mathcal{M}_1) \supseteq \textit{safe}_{t_a}(\mathcal{M}_1 \cap \mathcal{M}_2) \neq \emptyset$.*

Proof. Since $n - t_s \leq |\mathcal{M}_1 \cap \mathcal{M}_2| \leq |\mathcal{M}_1| \leq n - t_s + t_a$, Lemma 5.5 implies that $\text{safe}_{t_a}(\mathcal{M}_1 \cap \mathcal{M}_2) \neq \emptyset$. Afterwards, Lemma 5.10 implies that $\text{safe}_{t_a}(\mathcal{M}_1 \cap \mathcal{M}_2) \subseteq \text{safe}_{t_a}(\mathcal{M}_1)$. □

Lemma 5.12. *Let $\mathcal{M}_1, \mathcal{M}_2$ denote two sets of value-party pairs such that $|\mathcal{M}_1 \cup \mathcal{M}_2| \leq n$, and $|\mathcal{M}_1 \cap \mathcal{M}_2| \geq n - t_s$. Assume that $|\mathcal{M}_1| - (n - t_s) < t_a$ and $|\mathcal{M}_2| - (n - t_s) \geq t_a$. Then, $\textit{safe}_{t_a}(\mathcal{M}_1) \cap \textit{safe}_{k_\cup}(\mathcal{M}_1 \cup \mathcal{M}_2) \neq \emptyset$, where $k_\cup = |\mathcal{M}_1 \cup \mathcal{M}_2| - (n - t_s)$.*

Proof. Note that $t_a \leq k_\cup \leq t_s$, hence $\text{safe}_{k_\cup}(\mathcal{M}_1 \cup \mathcal{M}_2) \neq \emptyset$ according to Lemma 5.5.

We show that $\text{safe}_{t_a}(\mathcal{M}_1) \cap \text{safe}_{k_\cup}(\mathcal{M}_1 \cup \mathcal{M}_2) \neq \emptyset$. This is the intersection of the members of a collection containing the convex sets $\text{convex}(X)$ with $X \in \text{restrict}_{t_a}(\mathcal{M}_1)$ and $\text{convex}(Y)$ with $Y \in \text{restrict}_{k_\cup}(\mathcal{M}_1 \cup \mathcal{M}_2)$. Lemma 5.3 implies that this collection contains at least $2D + 2$ convex sets, which enables us to apply Theorem 5.2.

Then, it suffices to show that for any $a, b \geq 0$ such that $a + b = D + 1$, the intersection of any $X_1, X_2, \ldots, X_a \in \text{restrict}_{t_a}(\mathcal{M}_1)$ and $Y_1, Y_2 \ldots, Y_b \in \text{restrict}_{k_\cup}(\mathcal{M}_1 \cup \mathcal{M}_2)$ is not empty, which also applies to the intersection of the corresponding convex hulls.

We provide a lower bound for $\mathcal{M}_1 \cap \bigcap_{i=1}^{a} X_i \cap \bigcap_{i=1}^{b} Y_i$, with the help of Lemma 5.4: we obtain that $|\mathcal{M}_1 \cap \bigcap_{i=1}^{b} Y_i \cap \bigcap_{i=1}^{a} X_i| \geq |\mathcal{M}_1 \cap \bigcap_{i=1}^{b} Y_i| - a \cdot t_a$, since $X_i \in \text{restrict}_{t_a}(\mathcal{M}_1)$.

Lemma 5.4 also implies that $|\bigcap_{i=1}^{b} Y_i| \geq |\mathcal{M}_1 \cup \mathcal{M}_2| - b \cdot k_\cup$. Then, since $|(\mathcal{M}_1 \cup \mathcal{M}_2) \setminus \mathcal{M}_1| \leq t_s$ and $\bigcap_{i=1}^{b} Y_i \subseteq (\mathcal{M}_1 \cup \mathcal{M}_2)$, we obtain that $|\bigcap_{i=1}^{b} Y_i \cap \mathcal{M}_1| \geq |\mathcal{M}_1 \cup \mathcal{M}_2| - b \cdot k_\cup - t_s$.

Since $n \geq (D + 1) \cdot t_s + t_a + 1$, we have obtained the lower bound below, and that the intersection of these sets, and hence the intersection of their corresponding convex hull is non-empty. Then, Theorem 5.2 guarantees that $\text{safe}_{t_a}(\mathcal{M}_1) \cap \text{safe}_{k_\cup}(\mathcal{M}_1 \cup \mathcal{M}_2) \neq \emptyset$.

$$\begin{aligned} \left|\bigcap_{i=1}^{a} X_i \cap \bigcap_{i=1}^{b} Y_i\right| \geq \left|\mathcal{M}_1 \cap \bigcap_{i=1}^{a} X_i \cap \bigcap_{i=1}^{b} Y_i\right| &\geq \left|\mathcal{M}_1 \cap \bigcap_{i=1}^{a} Y_i\right| - a \cdot t_a \\ &\geq |\mathcal{M}_1 \cup \mathcal{M}_2| - b \cdot k_\cup - t_s - a \cdot t_a \geq 1. \end{aligned}$$

□

5.2 Analysis of $\Pi_{\text{AA-it}}$

In this subsection, we prove a few properties for $\Pi_{\text{AA-it}}$. We use $I_{\text{it}-1}$ to denote the multiset of values the honest parties join $\Pi_{\text{AA-it}}$ with. Then, the multiset I_{it} denotes the multiset of outputs obtained by the honest parties.

In the lemmas below, we assume that either the network is asynchronous, or the network is either synchronous and all honest parties start executing $\Pi_{\text{AA-it}}$ at the same time (meaning that Π_{oBC}'s properties hold).

We first ensure that the values obtained by the honest parties are indeed in the convex hull of $I_{\text{it}-1}$. The result below follows from Lemma 5.7, which shows that the safe area obtained by each honest party is included in $\text{convex}(I_{\text{it}-1})$.

Lemma 5.13. *If an honest party P obtains an output v, then $v \in \textit{convex}(I_{it-1})$.*

We now focus on showing that the honest parties' new values indeed get closer. We make use of the technical result below, which comes directly from [18].

Lemma 5.14. *Let $a, b, a', b' \in \mathbb{R}^D$ and some $\gamma \geq 0$ such that $\delta_{\max}(\{a, b, a', b'\}) \leq \gamma \leq \delta(a, b) + \delta(a', b')$. Then, setting $v = (a + b)/2$ and $v' = (a' + b')/2$, we have $\delta(v, v') \leq \sqrt{\frac{7}{8}}\gamma$.*

Lemma 5.8, which shows that the safe areas obtained by honest parties intersect, will now enable us to prove that honest values indeed get closer.

Lemma 5.15. *If two honest parties P and P' obtain outputs v and v', then $\delta(v, v') \leq \sqrt{\frac{7}{8}} \cdot \delta_{\max}(I_{it-1})$.*

Proof. Let S and S' denote the safe areas obtained by P and P' respectively. Then, let $a, b \in S$ such that $v = (a + b)/2$, and $a', b' \in S'$ such that $v' = (a' + b')/2$. We prove the statement with the help of Lemma 5.14.

Note that Π_{oBC}'s properties guarantee that $|\mathcal{M} \cap \mathcal{M}'| \geq n - t_s$ and $|\mathcal{M} \cap \mathcal{M}'| \leq n$. Then, Lemma 5.7 guarantees that S and S' are included in convex$(I_{\text{it}-1})$. This enables us to obtain an upper bound on the diameter of $\{a, b, a', b'\}$: $\gamma := \delta_{\max}(\{a, b, a', b'\}) \leq \delta_{\max}(I_{\text{it}-1})$. In order to apply Lemma 5.14, it remains to show that the condition $\gamma \leq \delta(a, b) + \delta(a', b')$ holds.

We need to provide an upper bound on $\delta(a, a')$ and $\delta(b, b')$. The properties of Π_{oBC} also enable us to use Lemma 5.8, which implies that S and S' have a non-empty intersection. Then, let c denote an arbitrary point in $S \cap S'$. Since a, b have the maximum distance in S, $\delta(a, c), \delta(b, c) \leq \delta(a, b)$. Analogously, $\delta(a', c), \delta(b', c) \leq \delta(a', b')$. Using the triangular inequality, we obtain that $\delta(a, a') \leq \delta(a, c) + \delta(c, a') \leq \delta(a, b) + \delta(a', b')$. Analogously, we obtain that $\delta(b, b') \leq \delta(a, b) + \delta(a', b')$.

Hence, we have shown that $\delta_{\max}(\{a, b, a', b'\}) = \gamma \leq \delta(a, b) + \delta(a', b')$. We can now conclude from Lemma 5.14 that $\delta(v, v') \leq \sqrt{\frac{7}{8}}\gamma \leq \sqrt{\frac{7}{8}} \cdot \delta_{\max}(I_{\text{it}-1})$. □

Finally, we focus on the liveness properties of $\Pi_{\text{AA-it}}$. The results below follow from Π_{oBC}'s (t_s, c_{oBC})-Synchronized Liveness and t_a-Liveness, along with the fact that the values computed by honest values are well defined, as ensured by Lemma 5.6.

Lemma 5.16. *Assume that the network is synchronous, and the honest parties start executing $\Pi_{AA\text{-}it}$ at the same time τ. Then, every honest party obtains an output by time $\tau + c_{AA\text{-}it} \cdot \Delta$, where $c_{AA\text{-}it} = c_{oBC}$.*

Lemma 5.17. *If the network is asynchronous and every honest party participates in $\Pi_{AA\text{-}it}$ until it obtains an output, then every honest party obtains an output.*

5.3 Analysis of Π_{init}

Before focusing on Π_{AA}, we will need the result included below. We defer its proof to the appendix.

Theorem 5.18. *Every honest party outputs (T, v_0) in Π_{init} such that v_0 is within the convex hull of honest inputs and, if I_0 denotes the multiset of outputs v_0, $T \geq \log_{\sqrt{7/8}}(\varepsilon/\delta_{\max}(I_0))$. In addition, if the network is synchronous, the honest parties obtain outputs at time $c_{init} \cdot \Delta$, where $c_{init} = 2c_{rBC} + c'_{rBC}$.*

5.4 Analysis of Π_{AA}

We now show that our protocol Π_{AA} achieves (t_s, t_a)-secure D-AA.

Theorem 5.19 follows directly from Lemma 5.23, which focuses on the synchronous setting, and Lemma 5.26, which focuses on the asynchronous setting.

Theorem 5.19. *Let $D > 1$ and $0 \leq t_a \leq t_s$ such that $n > (D + 1) \cdot t_s + t_a$. Then, Π_{AA} achieves:*

- *t_s-secure D-AA when it runs in a synchronous network;*
- *t_a-secure D-AA when it runs in an asynchronous network.*

Synchronous Network. We first analyze Π_{AA} in a synchronous network, where t_s of the parties involved are corrupted.

Firstly, we ensure that, until the first honest party outputs, honest parties stay synchronized and complete $\Pi_{\text{AA-it}}$ successfully in each iteration.

Lemma 5.20. *If no honest party outputs until time $\tau = (c_{init} + it \cdot c_{AA\text{-}it}) \cdot \Delta$, then the honest parties complete iteration it at time τ, and start every iteration $it' \leq it$ synchronously.*

Proof. We prove the statement using induction on it ≥ 0.

The base case is for it $= 0$: Every honest party completes the execution of Π_{init} at time $c_{\text{init}} \cdot \Delta$ according to Lemma 5.18, and starts the first iteration immediately.

For the induction step, we assume that the statement holds for iteration it, and we show that it also holds for iteration it + 1. Every honest party completes iteration it at time $(c_{\text{init}} + \text{it} \cdot c_{\text{AA-it}}) \cdot \Delta$, and the honest parties start iteration it + 1 immediately. As no honest party outputs before $c_{\text{AA-it}} \cdot \Delta$ time passes, Lemma 5.16 ensures that every honest party obtains an output by time $(c_{\text{init}} + (\text{it} + 1) \cdot c_{\text{AA-it}}) \cdot \Delta$. □

Then, when the first honest party P outputs, if it does so in some iteration it, the remaining honest parties may try to execute $\Pi_{\text{AA-it}}$ one more time, in iteration it + 1. Whether this additional execution of $\Pi_{\text{AA-it}}$ is successful is irrelevant – all honest parties output by iteration it + 1, and their outputs are values obtained when $\Pi_{\text{AA-it}}$ still offered guarantees.

Lemma 5.21. *If the first honest party P that outputs does so in iteration it, then every honest party outputs by iteration it + 1 a value computed by the end of iteration it.*

Proof. According to Lemma 5.20, every iteration it$' \leq$ it is completed by every honest party, and every honest party reaches the end of iteration it at the same time as P. As P has obtained $t_s + 1$ halting messages for previous iterations via Π_{rBC} by the end of iteration it, Π_{rBC}'s (t_s, c'_{rBC})-Conditional Liveness guarantees that every honest party receives $t_s + 1$ halting messages for iterations it$'$ < it + 1 within $c'_{\text{rBC}} < c_{\text{AA-it}}$ communications rounds, hence in iteration it + 1 the latest, and outputs a value computed in iteration it$'$ < it + 1. □

We now show that honest parties indeed obtain outputs.

Lemma 5.22. *Every honest party outputs a value obtained in an iteration completed by all honest parties.*

Proof. Let $T_{\min}$ denote the (t_s+1)-th smallest honest estimation T obtained in Π_{init} and assume that no honest party obtains output by time $\tau = (c_{\text{init}} + (T_{\min} + 1) \cdot c_{\text{AA-it}}) \cdot \Delta$. Otherwise, Lemma 5.21 proves the statement.

Then, according to Lemma 5.20, the honest parties complete the execution of $\Pi_{\text{AA-it}}$ in each iteration it $\leq T_{\min} + 1$ synchronously. Hence, at least $t_s + 1$ honest parties send (halt, it) messages for it $\leq T_{\min}$ by time $(c_{\text{init}} + T_{\min} \cdot c_{\text{AA-it}}) \cdot \Delta$ via Π_{rBC}. As Π_{rBC} achieves (c_{rBC}, t_s)-Honest Liveness, these messages are received within $c_{\text{rBC}} < c_{\text{AA-it}}$ communication rounds. At this point, every honest party is in iteration $T_{\min} + 1$, and then checks whether it has received $t_s + 1$ halting messages for iterations it $< T_{\min} + 1$ at time $(c_{\text{init}} + (T_{\min} + 1) \cdot c_{\text{AA-it}}) \cdot \Delta > (c_{\text{init}} + T_{\min} \cdot c_{\text{AA-it}} + c_{\text{rBC}}) \cdot \Delta$. Therefore, all honest parties output. □

So far, we have shown that all honest parties obtain outputs, and their outputs are obtained in iterations where $\Pi_{\text{AA-it}}$'s guarantees hold. We can now prove that Π_{AA} indeed achieves D-AA when it runs in a synchronous network.

Lemma 5.23. *When it runs in a synchronous network, Π_{AA} is a t_s-secure D-AA protocol.*

Proof. Lemma 5.22 ensures that every honest party outputs a value computed in an iteration where every honest party has completed $\Pi_{\text{AA-it}}$. Then, t_s-Liveness is immediately implied.

The t_s-Validity property follows from Lemma 5.13: the honest parties' outputs in Π_{AA} are obtained via $\Pi_{\text{AA-it}}$, hence within the convex hull of the values v_0 that honest parties start the first iteration with. Then, Lemma 5.18 ensures that these values v_0 are in the convex hull of the honest inputs.

It remains to show that (t_s, ε)-Agreement also holds. The values the honest parties output may be computed in different iterations. Let it_h denote the smallest such iteration. Additionally, let I_0 denote the set of values v_0 the honest parties obtain via Π_{init}, and let I_{it_h} denote the set of values obtained by the honest parties in iteration it_h. With an inductive argument, and with the help of Lemma 5.15, we obtain that $\delta_{\max}(I_{\text{it}_h}) \leq \left(\sqrt{7/8}\right)^{\text{it}_h} \cdot \delta_{\max}(I_0)$.

Note that it_h is the $(t_s + 1)$-th smallest iteration considered by the honest party for its output, and there are only t_s corrupted parties involved. This implies that it_h is greater than or equal to the smallest estimation T obtained by the honest parties via Π_{init}. Lemma 5.18 then ensures that $\text{it}_h \geq \log_{\sqrt{7/8}}(\varepsilon/\delta_{\max}(I_0))$. Then, we obtain that $\delta_{\max}(I_{\text{it}_h}) \leq \varepsilon$.

Therefore, the values computed by the honest parties in iteration it_h are already ε-close. If honest parties output values obtained in later iterations, Lemma 5.13 ensures that these values are included in convex(I_{it_h}), which means that (t_s, ε)-Agreement is achieved.

□

Asynchronous Network. We now analyze Π_{AA} when it runs in an asynchronous network and at most t_a parties are corrupted.

The following lemma shows that once the *first* honest party outputs, all parties can output. As opposed to the proof of Lemma 5.20, the first honest party is not necessarily the fastest to obtain an output. Instead, it is the honest party who obtains output in the earliest iteration.

Lemma 5.24. *If the earliest iteration when an honest party obtains output is iteration it, then all honest parties advance to iteration it and eventually obtain outputs.*

Proof. Since no honest party has obtained output before iteration it, all honest parties have joined the previous iterations' executions of $\Pi_{\text{AA-it}}$. Then, Lemma 5.17 guarantees that each of these executions is eventually completed, and therefore all honest parties eventually advance to iterations it.

Since an honest party obtains output in iteration it, it has received $t_s + 1$ halting messages for previous iterations. These messages are sent via Π_{rBC}, hence all other honest parties receive these messages as well. Since all other honest parties advance to iteration it, they eventually fulfil the condition for obtaining an output. □

Similarly to the synchronous case, we have to ensure that there is such a *first* honest party.

Lemma 5.25. *Every honest party eventually obtains an output.*

Proof. Let $T_{\min}$ denote the (t_s+1)-th smallest honest estimation obtained in Π_{init}. If an honest party outputs within these first $T_{\min}$ iterations, then Lemma 5.24 ensures that all honest parties obtain outputs eventually.

Otherwise, assume that no honest party outputs within the first $T_{\min}$ iterations. Lemma 5.17 ensures that the parties successfully complete every iteration up to and including $T_{\min}$, and advance to iteration $T_{\min} + 1$.

Then, at least $t_s + 1$ honest parties send halt messages during these $T_{\min}$ iterations, and these messages are eventually received. Since honest parties advance to iteration $T_{\min} + 1$, they fulfil the conditions to output.

□

We can now show that Π_{AA} indeed achieves t_a-secure D-AA.

Lemma 5.26. *When it runs in an asynchronous network, Π_{AA} is a t_a-secure D-AA protocol.*

Proof. Lemma 5.25 shows that each honest party eventually outputs a value in Π_{AA}, hence the t_a-Liveness property is satisfied. Afterwards, the arguments proving that t_a-Validity and (t_s, ε)-Agreement hold are analogous to the proof of Lemma 5.23.

□

REFERENCES

[1] Ittai Abraham, Yonatan Amit, and Danny Dolev. 2005. Optimal Resilience Asynchronous Approximate Agreement. In *Principles of Distributed Systems*, Teruo Higashino (Ed.). Springer Berlin Heidelberg, Berlin, Heidelberg, 229–239.

[2] Dan Alistarh, Faith Ellen, and Joel Rybicki. 2021. Wait-Free Approximate Agreement on Graphs. In *Structural Information and Communication Complexity*, Tomasz Jurdziński and Stefan Schmid (Eds.). Springer International Publishing, Cham, 87–105. https://doi.org/10.1007/978-3-030-79527-6_6

[3] Ananya Appan, Anirudh Chandramouli, and Ashish Choudhury. 2022. Perfectly-Secure Synchronous MPC with Asynchronous Fallback Guarantees. In *Proceedings of the 2022 ACM Symposium on Principles of Distributed Computing* (Salerno, Italy) *(PODC'22)*. Association for Computing Machinery, New York, NY, USA, 92–102. https://doi.org/10.1145/3519270.3538417

[4] Michael Ben-Or, Danny Dolev, and Ezra N. Hoch. 2010. Brief Announcement: Simple Gradecast Based Algorithms. In *Distributed Computing*, Nancy A. Lynch and Alexander A. Shvartsman (Eds.). Springer Berlin Heidelberg, Berlin, Heidelberg, 194–197.

[5] Erica Blum, Jonathan Katz, and Julian Loss. 2019. Synchronous consensus with optimal asynchronous fallback guarantees. In *Theory of Cryptography Conference*. Springer, 131–150.

[6] Erica Blum, Jonathan Katz, and Julian Loss. 2021. Tardigrade: An Atomic Broadcast Protocol for Arbitrary Network Conditions. In *ASIACRYPT 2021, Part II (LNCS, Vol. 13091)*, Mehdi Tibouchi and Huaxiong Wang (Eds.). Springer, Heidelberg, 547–572. https://doi.org/10.1007/978-3-030-92075-3_19

[7] Erica Blum, Chen-Da Liu-Zhang, and Julian Loss. 2020. Always Have a Backup Plan: Fully Secure Synchronous MPC with Asynchronous Fallback. In *Advances in Cryptology – CRYPTO 2020*, Daniele Micciancio and Thomas Ristenpart (Eds.). Springer International Publishing, Cham, 707–731.

[8] Zohir Bouzid, Maria Gradinariu Potop-Butucaru, and Sébastien Tixeuil. 2010. Optimal Byzantine-resilient convergence in uni-dimensional robot networks. *Theoretical Computer Science* 411, 34 (2010), 3154–3168. https://doi.org/10.1016/j.tcs.2010.05.006

[9] Gabriel Bracha. 1987. Asynchronous Byzantine agreement protocols. *Information and Computation* 75, 2 (1987), 130–143.

[10] Ran Canetti and Tal Rabin. 1993. Fast asynchronous Byzantine agreement with optimal resilience. In *25th ACM STOC*. ACM Press, 42–51. https://doi.org/10.1145/167088.167105

[11] Ludwig Danzer. 1963. " Helly's theorem and its relatives," in Convexity. In *Proc. Symp. Pure Math.*, Vol. 7. Amer. Math. Soc., 101–180.

[12] Giovanni Deligios, Martin Hirt, and Chen-Da Liu-Zhang. 2021. Round-efficient byzantine agreement and multi-party computation with asynchronous fallback. In *Theory of Cryptography Conference*. Springer, 623–653.

[13] Danny Dolev, Nancy A. Lynch, Shlomit S. Pinter, Eugene W. Stark, and William E. Weihl. 1986. Reaching Approximate Agreement in the Presence of Faults. *J. ACM* 33, 3 (May 1986), 499–516. https://doi.org/10.1145/5925.5931

[14] El-Mahdi El-Mhamdi, Rachid Guerraoui, Arsany Guirguis, Lê Nguyên Hoang, and Sébastien Rouault. 2020. Genuinely distributed byzantine machine learning. In *Proceedings of the 39th Symposium on Principles of Distributed Computing*. 355–364.

[15] Alan David Fekete. 1986. Asymptotically Optimal Algorithms For Approximale Agreement. In *5th ACM PODC*, Joseph Y. Halpern (Ed.). ACM, 73–87. https://doi.org/10.1145/10590.10597

[16] Alan David Fekete. 1987. Asynchronous Approximate Agreement. In *6th ACM PODC*, Fred B. Schneider (Ed.). ACM, 64–76. https://doi.org/10.1145/41840.41846

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

[18] Matthias Függer and Thomas Nowak. 2018. Fast Multidimensional Asymptotic and Approximate Consensus. In *International Symposium on DIStributed Computing (DISC) 2018*.

[19] Matthias Függer, Thomas Nowak, and Manfred Schwarz. 2021. Tight bounds for asymptotic and approximate consensus. *Journal of the ACM (JACM)* 68, 6 (2021), 1–35.

[20] Diana Ghinea, Chen-Da Liu-Zhang, and Roger Wattenhofer. 2022. Optimal Synchronous Approximate Agreement with Asynchronous Fallback. In *Proceedings of the 2022 ACM Symposium on Principles of Distributed Computing* (Salerno, Italy) *(PODC'22)*. Association for Computing Machinery, New York, NY, USA, 70–80. https://doi.org/10.1145/3519270.3538442

[21] Joseph Y. Halpern, Barbara Simons, H. Raymond Strong, and Danny Dolev. 1984. Fault-Tolerant Clock Synchronization. In *3rd ACM PODC*, Robert L. Probert, Nancy A. Lynch, and Nicola Santoro (Eds.). ACM, 89–102. https://doi.org/10.1145/800222.806739

[22] Martin Hirt, Ard Kastrati, and Chen-Da Liu-Zhang. 2021. Multi-Threshold Asynchronous Reliable Broadcast and Consensus. In *24th International Conference on Principles of Distributed Systems*. 1.

[23] Leslie Lamport and P. M. Melliar-Smith. 1985. Synchronizing Clocks in the Presence of Faults. *J. ACM* 32, 1 (Jan. 1985), 52–78. https://doi.org/10.1145/2455.2457

[24] Jérémy Ledent. 2021. Brief Announcement: Variants of Approximate Agreement on Graphs and Simplicial Complexes. In *Proceedings of the 2021 ACM Symposium on Principles of Distributed Computing* (Virtual Event, Italy) *(PODC'21)*. Association for Computing Machinery, New York, NY, USA, 427–430. https://doi.org/10.1145/3465084.3467946

[25] Chen-Da Liu-Zhang, Julian Loss, Ueli Maurer, Tal Moran, and Daniel Tschudi. 2020. MPC with Synchronous Security and Asynchronous Responsiveness. In *ASIACRYPT 2020, Part III (LNCS, Vol. 12493)*, Shiho Moriai and Huaxiong Wang (Eds.). Springer, Heidelberg, 92–119. https://doi.org/10.1007/978-3-030-64840-4_4

[26] Hammurabi Mendes and Maurice Herlihy. 2013. Multidimensional approximate agreement in Byzantine asynchronous systems. In *45th ACM STOC*, Dan Boneh, Tim Roughgarden, and Joan Feigenbaum (Eds.). ACM Press, 391–400. https://doi.org/10.1145/2488608.2488657

[27] Hammurabi Mendes, Maurice Herlihy, Nitin Vaidya, and Vijay K Garg. 2015. Multidimensional agreement in Byzantine systems. *Distributed Computing* 28, 6 (2015), 423–441.

[28] Atsuki Momose and Ling Ren. 2021. Multi-Threshold Byzantine Fault Tolerance. In *Proceedings of the 2021 ACM SIGSAC Conference on Computer and Communications Security* (Virtual Event, Republic of Korea) *(CCS '21)*. Association for Computing Machinery, New York, NY, USA, 1686–1699. https://doi.org/10.1145/3460120.3484554

[29] Thomas Nowak and Joel Rybicki. 2019. Byzantine Approximate Agreement on Graphs. In *33rd International Symposium on Distributed Computing (DISC 2019) (Leibniz International Proceedings in Informatics (LIPIcs), Vol. 146)*, Jukka Suomela (Ed.). Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, Dagstuhl, Germany, 29:1–29:17. https://doi.org/10.4230/LIPIcs.DISC.2019.29

[30] Maria Potop-Butucaru, Michel Raynal, and Sebastien Tixeuil. 2011. Distributed Computing with Mobile Robots: An Introductory Survey. *Proceedings - 2011 International Conference on Network-Based Information Systems, NBiS 2011*, 318 – 324. https://doi.org/10.1109/NBiS.2011.55

[31] Lili Su and Nitin H Vaidya. 2016. Fault-tolerant multi-agent optimization: optimal iterative distributed algorithms. In *Proceedings of the 2016 ACM symposium on principles of distributed computing*. 425–434.

[32] Nitin H. Vaidya and Vijay K. Garg. 2013. Byzantine vector consensus in complete graphs. In *32nd ACM PODC*, Panagiota Fatourou and Gadi Taubenfeld (Eds.). ACM, 65–73. https://doi.org/10.1145/2484239.2484256

[33] Jennifer Lundelius Welch and Nancy Lynch. 1988. A new fault-tolerant algorithm for clock synchronization. *Information and Computation* 77, 1 (1988), 1–36. https://doi.org/10.1016/0890-5401(88)90043-0

6 APPENDIX

6.1 Protocol Π_{rBC}

We prove the theorem below.

Theorem 4.2. *There is a (t, c_{rBC}, c'_{rBC})-secure Reliable Broadcast protocol Π_{rBC} for $n > 3t$, $c_{rBC} = 3$, and $c'_{rBC} = 2$.*

Consider Bracha's Reliable Broadcast protocol below [9], as presented in [10].

Protocol Π_{rBC}

Code for sender S on input v

```
1: Send $v$ to all the parties
```

Code for party P

```
1: Upon receiving $v$ from $S$:
2:     Send (echo, $v$) to all the parties.
3: Upon receiving (echo, $v$) from $n - t$ parties or
4:                (ready, $v$) from $t + 1$ parties:
5:     Send (ready, $v$) to all the parties.
6: Upon receiving (ready, $v$) from $n - t$ parties:
7:     Output $v$.
```

The results below show that Π_{rBC} is a $(t, c_{\mathrm{rBC}}, c'_{\mathrm{rBC}})$-secure Reliable Broadcast protocol, where $c_{\mathrm{rBC}} = 3$ and $c'_{\mathrm{rBC}} = 2$. In Lemmas 6.1 and 6.4, we show that Π_{rBC} satisfies t-Validity and t-Consistency. If every honest party joins the execution of the protocol, Lemma 6.5 shows that $(t, c_{\mathrm{rBC}'})$-Honest Liveness is achieved. If, in addition, the network is synchronous $(t, c_{\mathrm{rBC}}, c'_{\mathrm{rBC}})$-Conditional Liveness follows from 6.5.

We first show that Π_{rBC} achieves t-Validity.

Lemma 6.1 (t-Validity). *Assume that the sender S is honest and has input v. Then, even if not all honest parties participate in the protocol's execution, for any $v' \neq v$, no honest party sends (echo, v'), (ready, v'), or outputs v'.*

Proof. Since S is honest, no party receives $v' \neq v$ from S. Hence, no honest party sends (echo, v'). As no honest party receives $n-t > t$ (echo, v') messages, no honest party sends (ready, v'). It follows that no honest party outputs v'. □

Lemma 6.2. *Assume that the sender S is honest and has input v. Then, if all honest parties join the protocol's execution, every honest party outputs v eventually. In addition, if the network is synchronous and the honest parties started executing the protocol at the same time τ, they output v by time $\tau + 3 \cdot \Delta$.*

Proof. Lemma 6.1 guarantees that no honest party outputs $v' \neq v$. Then, every party eventually receives v from the honest sender. If the network is synchronous, these messages are received by time $\tau + \Delta$. Then, every honest party sends (echo, v).

Every party eventually (or by time $\tau + 2 \cdot \Delta$, if the network is synchronous) receives $n - t$ messages (echo, v) and hence sends (ready, v).

Finally, every honest party eventually (or by time $\tau + 3 \cdot \Delta$, if the network is synchronous) receives $n - t$ messages (ready, v) and terminates with output v. □

Lemma 6.3. *If an honest party P sends (ready, v), then no honest party sends (ready, v') for $v' \neq v$, even if not all honest parties join the protocol's execution.*

Proof. Without loss of generality, assume that P and P' are the first honest parties that send (ready, v) and (ready, v') respectively. Then, P has received (echo, v) from $n - t$ parties, while P' has received (echo, v') from $n-t$ parties. Then, there is a set of $n-2t > t$ parties, hence at least one honest party, that sent multiple echo messages for different values, which contradicts the protocol. □

Lemma 6.4 (t-Consistency). *If two honest parties P, P' obtain outputs v and v' respectively, then $v = v'$, even if not all honest parties join the protocol's execution.*

Proof. As P has output v, at least one honest party has sent (ready, v). From Lemma 6.3 it follows that no honest party has sent (ready, v'') for $v'' \neq v$. Therefore, if P' obtains an output v', then $v = v'$. □

Lemma 6.5. *If all honest parties join the protocol's execution, and an honest party P outputs v, then every honest party outputs v. In addition, if the network is synchronous and the parties have started executing the protocol at the same time, then every honest party outputs v within $2 \cdot \Delta$ time.*

Proof. By Lemma 6.4, no honest party outputs $v' \neq v$.

P has received $n - t$ messages (ready, v). Lemma 6.3 guarantees that no honest party sends (ready, v') for $v' \neq v$. Hence, every honest party receives at least $n - 2t > t + 1$ messages (ready, v) eventually, or within Δ time from the moment P has obtained an output, if the network is synchronous. Then, every honest party sends (ready, v). If the network is asynchronous, the messages are delivered eventually, and each honest party terminates. If the network is synchronous, these messages are delivered within additional Δ time, hence every honest party outputs v within $2 \cdot \Delta$ time after P has obtained output. □

6.2 Analysis of Π_{oBC}

We prove the theorem below.

Theorem 4.4. *Π_{oBC} is a (t_s, t_a, c_{oBC})-secure Overlap All-to-All Broadcast protocol for $c_{oBC} = c_{rBC} + c'_{rBC}$.*

The results below trivially follow from Π_{rBC}'s t_s-Validity and t_s-Consistency, regardless of whether the network is synchronous or asynchronous.

Lemma 6.6 (t_s-Validity). *Let P and P' denote two honest parties, and assume that P outputs $\mathcal{M}$. If $(v, P') \in \mathcal{M}$, then $v = v_{P'}$, even if not all honest parties join the protocol's execution.*

Lemma 6.7 (t_s-Consistency). *Let P and P' denote two honest parties, and assume they output $\mathcal{M}$ and $\mathcal{M}'$ respectively. If $(v, P'') \in \mathcal{M}$ and $(v', P'') \in \mathcal{M}'$, then $v = v'$, even if not all honest parties join the protocol's execution.*

Lemma 6.8 ((t_s, t_a)-Overlap). *Let P and P' denote two honest parties. If P and P' output $\mathcal{M}$ and $\mathcal{M}'$ respectively, then $|\mathcal{M} \cap \mathcal{M}'| \geq n - t_s$, even if not all parties join the protocol's execution.*

Proof. We show that P and P' have an honest witness in common. Since they obtain outputs, $|W|, |W'| \geq n - t_s$, which implies that $|W \cap W'| \geq n - 2t_s \geq t_s + t_a + 1$. Hence, as at most t_s of the parties involved are corrupted, $W \cap W'$ contains an honest witness P_W.

Then, P and P' have received the same set $\mathcal{M}_{P_W}$ of size at least $n - t_s$ from P_W such that $\mathcal{M}_{P_W} \subseteq \mathcal{M}$ and $\mathcal{M}_{P_W} \subseteq \mathcal{M}'$, which proves the statement. □

Lemma 6.9 ((t_s, t_a)-Synchronized Overlap and Synchronized (t_s, c_{oBC})-Synchronized Liveness). *Assume that the honest parties start executing Π_{oBC} at the same time τ. Let $c_{oBC} = c_{rBC} + c'_{rBC}$.*

Then, every honest party P outputs $\mathcal{M}$ by time $\tau + c_{oBC} \cdot \Delta$, and $\mathcal{M}$ contains $(v_{P'}, P')$ for every honest party P'.

Proof. Since every honest party starts the protocol at time τ, Π_{rBC} achieves (t_s, c_{rBC})-Honest Liveness. That is, every honest value is received by time $\tau + c_{\mathrm{rBC}} \cdot \Delta$. Hence, at time $\tau + c_{\mathrm{rBC}} \cdot \Delta$, the set $\mathcal{M}$ of every honest party contains the $n - t_s$ pairs corresponding to honest values, and possibly some pairs corresponding to corrupted values. Hence, (t_s, t_a)-Synchronized Overlap holds.

Then, we need to show that $|W| \geq n - t_s$ holds by time $\tau + (c_{\mathrm{rBC}} + c'_{\mathrm{rBC}}) \cdot \Delta$ for every honest party P. Since every honest party P' has sent its set $\mathcal{M}_{P'}$ at time $\tau + c_{\mathrm{rBC}} \cdot \Delta$, every set $\mathcal{M}_{P'}$ is received by time $\tau + (c_{\mathrm{rBC}} + 1) \cdot \Delta$. In addition, Π_{rBC} achieves (t_s, c'_{rBC})-Conditional Liveness, which ensures that every value received by P' by time $\tau + c_{\mathrm{rBC}} \cdot \Delta$ and included in $\mathcal{M}_{P'}$ is received by every party by time $\tau + (c_{\mathrm{rBC}} + c'_{\mathrm{rBC}}) \cdot \Delta$. As $c'_{\mathrm{rBC}} \geq 1$, by time $\tau + (c_{\mathrm{rBC}} + c'_{\mathrm{rBC}}) \cdot \Delta$, P adds P' to its set W for each honest party P', and hence $|W| \geq n - t_s$ holds. Therefore, (t_s, c_{oBC})-Synchronized Liveness holds as well. □

Lemma 6.10 (t_a-Liveness). *If all honest parties join the protocol's execution, all honest parties obtain an output.*

Lemma 6.11. *Let P and P' denote two honest parties, and assume that P' has not yet obtained an output. Since every honest party actively participates in the protocol, P eventually receives at least $n - t_s$ values via Π_{rBC}, and hence eventually sends a set $\mathcal{M}_P$ to all the parties. As Π_{rBC} achieves (t_a, c_{rBC})-Conditional Liveness, P' can eventually receive each of these values, along with $\mathcal{M}_P$, and therefore mark P as a witness. As this result holds for every honest P, P' eventually obtains $n - t_s$ witnesses and outputs.*

6.3 Proofs for Safe Area Properties

Lemma 5.4. *Assume that X is a finite set, $m \leq D + 1$, and let $X_1, \ldots, X_m \in \mathrm{restrict}_t(X)$. Then, $|\bigcap_{i=1}^{m} X_i| \geq |X| - mt$.*

In addition, given a finite set Y, $|X \cap Y \cap \bigcap_{i=1}^{m} X_i| \geq |X \cap Y| - mt$.

Proof. We prove the second part of the statement by induction on m.

The base case is when $m = 1$: $|X \cap Y \cap \bigcap_{1 \leq i \leq 1} X_i| = |(X \cap Y) \cap X_1| = |(X \cap Y) \setminus ((X \cap Y) \setminus X_1)|$. Since $X \cap Y \subseteq X$ and $|X \setminus X_1| = t$, then $|(X \cap Y) \setminus X_1| \leq t$. Therefore, $|X \cap Y \cap X_i| \geq |X \cap Y| - t$.

For the induction step, assume that $|X \cap Y \cap \bigcap_{1 \leq i \leq m-1} X_i| \geq |X \cap Y| - (m-1)t$. Similarly to the argument for the base case, it holds that $|(X \cap Y \cap \bigcap_{1 \leq i \leq m-1} X_i) \setminus X| \leq t$ since $X \cap Y \cap \bigcap_{1 \leq i \leq m-1} X_i \subseteq X$ and $|X \setminus X_i| = t$. Therefore, $|X \cap Y \cap \bigcap_{1 \leq i \leq m} X_i| \geq |X \cap Y| - mt$.

Then, by setting $Y = X$, we obtain that $|X \cap Y \cap \bigcap_{1 \leq i \leq m} X_i| \geq |X \cap Y| - mt = |X| - mt$. □

Before proving Lemma 5.10, we need to include the following technical result.

Lemma 6.12. *$\mathit{safe}_t(\mathcal{M}) \subseteq \mathit{safe}_{t-1}(\mathcal{M})$.*

Proof.

$$\begin{aligned}
\mathsf{safe}_t(\mathcal{M}) &= \bigcap_{M \in \mathsf{restrict}_t \mathcal{M}} \mathsf{convex}(\mathsf{val}(M)) \\
&\subseteq \bigcap_{M \in \mathsf{restrict}_t(\mathcal{M})} \left(\bigcap_{m \in \mathcal{M} \setminus M} \mathsf{convex}(\mathsf{val}(M \cup \{m\})) \right) \\
&= \bigcap_{M \in \mathsf{restrict}_{t-1}(\mathcal{M})} \mathsf{convex}(\mathsf{val}(M)) = \mathsf{safe}_{t-1}(\mathcal{M}).
\end{aligned}$$

□

Lemma 5.10. *Let m denote a value-party pair, and $\mathcal{M}$ a set of value-party pairs. Then, $\mathit{safe}_t(\mathcal{M}) \subseteq \mathit{safe}_t(\mathcal{M} \cup \{m\})$.*

Proof. Firstly, note that every set $M \in \mathsf{restrict}_t(\mathcal{M} \cup \{m\})$ contains $k = |\mathcal{M}| + 1 - t$ values. Out of these values, either k are in $\mathsf{safe}_t(\mathcal{M})$, or $k - 1$ are in $\mathsf{safe}_t(\mathcal{M})$ and the k-th value is m. Then, since the sets of $\mathsf{restrict}_t(\mathcal{M})$ only contain $|\mathcal{M}| - t$ values, we obtain that $\mathsf{restrict}_t(\mathcal{M} \cup \{m\}) = \mathsf{restrict}_{t-1}(\mathcal{M}) \cup \{M \cup \{m\} : M \in \mathsf{restrict}_t(\mathcal{M})\}$.

Hence, we can write $\mathsf{safe}_t(\mathcal{M} \cup \{m\})$ as follows:

$$\begin{aligned}
&\bigcap_{M \in \mathsf{restrict}_{t-1}(\mathcal{M})} \mathsf{convex}(\mathsf{val}(M)) \cap \bigcap_{M \in \mathsf{restrict}_t(\mathcal{M})} \mathsf{convex}(\mathsf{val}(M)) \\
&\quad = \mathsf{safe}_{t-1}(\mathcal{M}) \cap \bigcap_{M \in \mathsf{restrict}_t(\mathcal{M})} \mathsf{convex}(\mathsf{val}(M \cup \{m\})).
\end{aligned}$$

From Lemma 6.12, we obtain that $\mathsf{safe}_t(\mathcal{M}) \subseteq \mathsf{safe}_{t-1}(\mathcal{M})$. In addition, for any $V \subseteq \mathbb{R}^D$ and $v \in \mathbb{R}^D$, $\mathsf{convex}(V) \subseteq \mathsf{convex}(V \cup \{v\}$. This is because v is either inside $\mathsf{convex}(V)$ or it extends the convex hull. Then, for any $M \in \mathsf{restrict}_t(\mathcal{M})$, $\mathsf{convex}(\mathsf{val}(M)) \subseteq \mathsf{convex}(\mathsf{val}(M \cup \{m\}))$. Therefore, $\mathsf{safe}_t(\mathcal{M}) \subseteq \mathsf{safe}_t(\mathcal{M} \cup \{m\})$. □

Lemma 5.3. *Let $\mathcal{M}$ denote a set of $n - t_s + k$ value-party pairs, where $k \leq t_s$. Then, $|\mathit{restrict}_{\max(k,t_a)}(\mathcal{M})| \geq D + 1$.*

Proof. We obtain a lower bound for $m = |\mathsf{restrict}_{\max(k,t_a)}(\mathcal{M})|$ as follows:

$$m = \binom{n - t_s + k}{\max(k, t_a)} = \prod_{i=1}^{\max(k,t_a)} \frac{n - t_s + k - \max(k, t_a) + i}{i}.$$

Note that each term of the product is greater than 1 since $n - t_s + k - \max(k, t_a) + i \geq D \cdot t_s + k + 1 + i \geq 1 + i$. Therefore, we obtain that: $m \geq n - t_s + k - \max(k, t_a) + 1 \geq D \cdot t_s + k + 1 \geq D + 1$. □

6.4 Analysis of Π_{init}

Lemma 6.13. *For every honest party P, it eventually holds that $|W| \geq n - t_s$. In addition, if the network is synchronous, every honest party obtains a set W that contains every honest party by time $\tau_{\mathtt{start}} + 2c_{rBC} \cdot \Delta$.*

Proof. Π_{rBC} guarantees t_s-Validity, (t_s, c_{rBC})-Honest Liveness, and (t_s, c'_{rBC})-Conditional Liveness.

Then, if the network is asynchronous, every honest party eventually receives $n - t_s$ values and reliably broadcasts its set $\mathcal{M}$. These sets are delivered eventually.

If the network is synchronous, each honest party receives at least the $n - t_s$ honest values by time $\tau_{\mathtt{start}} + c_{\mathrm{rBC}} \cdot \Delta$ and reliably broadcasts its set $\mathcal{M}$ immediately. These sets are delivered by time $\tau_{\mathtt{start}} + 2c_{\mathrm{rBC}} \cdot \Delta$.

Due to Π_{rBC}'s t_s-Consistency and (t_s, c'_{rBC})-Conditional Liveness, every value in a set $\mathcal{M}$ sent by an honest party can be received by all other parties, hence at least $n - t_s$ parties are marked as witnesses eventually.

If the network is synchronous, any value in a set $\mathcal{M}$ sent by an honest party P is received by time $\tau_{\mathtt{start}} + (c_{\mathrm{rBC}} + c'_{\mathrm{rBC}}) \cdot \Delta < \tau_{\mathtt{start}} + 2c_{\mathrm{rBC}} \cdot \Delta$, and hence P can be marked immediately as a witness. As this result holds for any honest party P, at time $\tau_{\mathtt{start}} + (c_{\mathrm{rBC}} + c'_{\mathrm{rBC}}) \cdot \Delta$, every honest party has obtained a set W containing every honest party. □

Lemma 6.14. *Eventually, $|W_2| \geq n - t_s$ for every honest party. In addition, if the network is synchronous, W_2 contains all the honest parties at time $\tau_{\mathtt{start}} + (2c_{rBC} + c'_{rBC}) \cdot \Delta$ for every honest party.*

Proof. Lemma 6.13 shows that every honest party sends its set W eventually. In addition, if the network is synchronous, this set contains every honest party and is sent by time $\tau_{\mathtt{start}} + 2c_{\mathrm{rBC}} \cdot \Delta$.

If the network is asynchronous and $P' \in W$ for some honest party P, then eventually every honest party receives the same set $\mathcal{M}_{P'}$ as P due to (t_s, c'_{rBC})-Conditional Liveness. Then, every honest party can add P' to its own set W and mark P as a double-witness. Therefore, $|W_2| \geq n - t_s$ eventually holds for every honest party.

If the network is synchronous, then the set W sent by an honest party P is received by time $\tau_{\mathtt{start}} + (2c_{\mathrm{rBC}} + 1) \cdot \Delta$. Π_{rBC} guarantees (t_s, c'_{rBC})-Conditional Liveness, which implies that every party $P' \in W$ is marked as a witness by every honest party by time $\tau_{\mathtt{start}} + (2c_{\mathrm{rBC}} + c'_{\mathrm{rBC}}) \cdot \Delta$, and hence P is marked immediately as a double-witness. Finally, since $c'_{\mathrm{rBC}} \geq 1$, every honest party marks every honest party as a double-witness by time $\tau_{\mathtt{start}} + (2c_{\mathrm{rBC}} + c'_{\mathrm{rBC}}) \cdot \Delta$. □

The result below follows from Lemma 5.6 and Lemma 5.7.

Lemma 6.15. *If an honest party P adds P' to W, then $v_{P'}$ is well-defined. In addition, if P' is honest, $v_{P'}$ is in the honest inputs' convex hull.*

Lemmas 6.14 and 6.15 show that every party obtains a set I_e of $n - t_s + k$ value-party pairs, containing at least $n - t_s + k - \max(k, t_a)$ values in the honest inputs' convex hull. Then, we apply Lemma 5.7 once again and obtain the following.

Lemma 6.16. *Every honest party P eventually obtains a well-defined set I_e, a well-defined value $v_0 \subseteq$ convex(I), and outputs. In addition, if the network is synchronous, every honest party outputs at time $\tau_{\mathtt{start}} + (2c_{rBC} + c'_{rBC}) \cdot \Delta$.*

The property below follows from the t_s-Consistency property of Π_{rBC}.

Lemma 6.17. *Let P, P' denote two honest parties, and let I_e and I'_e denote their sets of estimations. If $(v_{P''}, P'') \in I_e$ and $(v'_{P''}, P'') \in I'_e$, then $v_{P''} = v'_{P''}$.*

Lemma 6.18. *Let P and P' denote two honest parties and let I_e and I'_e denote their estimated values, and v_0 and v'_0 their outputs. Then, $v_0 \in$ convex(I'_e).*

Proof. If the network is synchronous, Lemma 6.13 guarantees that P and P' have $n - t_s$ witnesses in common. If the network is asynchronous, P and P' obtain $n - 2t_s > t_a + 1$ common double witnesses, hence at least one honest double witness in common, from which they received the same set W of $n - t_s$ witnesses. Hence, P and P' have $n - t_s$ witnesses in common. In both cases, I_e and I'_e have a subset of $n - t_s$ common pairs according to Lemma 6.17.

Let $k = |I_e| - (n - t_s)$. Then, $\mathrm{restrict}_{\max(k, t_a)}(I_e)$ includes a subset $I_{e,\cap}$ of the common $n - t_s$ pairs, which implies that $v_0 \in \mathrm{safe}_{\max(k, t_a)}(I_e) \subseteq \mathrm{convex}(I_{e,\cap}) \subseteq I'_e$. □

The result below follows immediately from Lemmas 6.16 and 6.18

Lemma 5.18. *Every honest party outputs (T, v_0) in Π_{init} such that v_0 is within the convex hull of honest inputs and, if I_0 denotes the multiset of outputs v_0, $T \geq \log_{\sqrt{7/8}}(\varepsilon / \delta_{\max}(I_0))$. In addition, if the network is synchronous, the honest parties obtain outputs at time $c_{init} \cdot \Delta$, where $c_{init} = 2c_{rBC} + c'_{rBC}$.*

Proof. Lemma 6.16 shows that each honest party eventually obtains a set of estimations I_e, a value $v_0 \subseteq$ convex(I), and hence an estimation T. If the network is synchronous, these outputs are obtained at time $c_{\mathrm{init}} \cdot \Delta$. Then, Lemma 6.18 guarantees that $I_0 \subseteq$ convex(I_e) for every honest party. Then, $\delta_{\max}(I_e) \geq \delta_{\max}(I_0)$ and hence $T = \log_{\sqrt{7/8}}(\varepsilon / \delta_{\max}(I_e)) \geq \log_{\sqrt{7/8}}(\varepsilon / \delta_{\max}(I_0))$. □