Multidimensional Approximate Agreement in Byzantine Asynchronous Systems

Hammurabi Mendes, Maurice Herlihy
Brown University
Computer Science Department
Providence RI 02916
{hmendes,mph}@cs.brown.edu

ABSTRACT

The problem of ϵ-approximate agreement in Byzantine asynchronous systems is well-understood when all values lie on the real line. In this paper, we generalize the problem to consider values that lie in $\mathbb{R}^m$, for $m \geq 1$, and present an optimal protocol in regard to fault tolerance.

Our scenario is the following. Processes start with values in $\mathbb{R}^m$, for $m \geq 1$, and communicate via message-passing. The system is *asynchronous*: there is no upper bound on processes' relative speeds or on message delay. Some *faulty* processes can display arbitrarily malicious (i.e. Byzantine) behavior. Non-faulty processes must decide on values that are: (1) in $\mathbb{R}^m$; (2) within distance ϵ of each other; and (3) in the convex hull of the non-faulty processes' inputs. We give an algorithm with a matching lower bound on fault tolerance: we require $n > t(m+2)$, where n is the number of processes, t is the number of Byzantine processes, and input and output values reside in $\mathbb{R}^m$. Non-faulty processes send $O(n^2 d \log(m/\epsilon \max\{\delta(d) : 1 \leq d \leq m\}))$ messages in total, where $\delta(d)$ is the range of non-faulty inputs projected at coordinate d. The Byzantine processes do not affect the algorithm's running time.

Categories and Subject Descriptors

D.1.3 [**Programming Techniques**]: Concurrent Programming; F.1.2 [**Computation by Abstract Devices**]: Modes of Computation – parallelism and concurrency

Keywords

approximate agreement; higher dimension; Byzantine protocols; asynchronous systems

1. INTRODUCTION

Consider a system comprised of n processes that communicate by message-passing. Scheduling and communication are *asynchronous*: there is no bound on processes' relative speeds or on message delivery time. At most t processes may be faulty, displaying *Byzantine* (arbitrary, even malicious) behavior. The combination of asynchrony and failures means that it is impossible to distinguish between a faulty process that has halted and a non-faulty process that is simply slow to respond.

In our *multidimensional ϵ-approximate agreement task*, for arbitrary $\epsilon > 0$ and $m \geq 1$, each process starts with an input value in $\mathbb{R}^m$, and all non-faulty processes must choose output values, also in $\mathbb{R}^m$, such that (1) all outputs lie within ϵ of one another, and (2) all outputs lie in the convex hull of the inputs of the non-faulty processes. A *t-resilient protocol* for this task is an algorithm guaranteeing that each non-faulty process decides on a correct output, despite the presence of up to t Byzantine processes.

Our problem has a long background. In the *consensus task* [9], processes start with values from an arbitrary domain, and must choose values such that: (1) all processes decide on the same value and (2) that value is some process' input. It is well-known that asynchronous consensus is impossible in the presence of even a single crash failure [9]. However, there are many ways to circumvent this impossibility[1], and one is to settle for (unidimensional) *ϵ-approximate agreement* [6]. In this case, non-faulty processes start with input values in $\mathbb{R}$, and must also finish with output values in $\mathbb{R}$, satisfying:

Agreement: all non-faulty processes decide on values that are within an arbitrary distance $\epsilon \geq 0$ of each other;

Validity: all non-faulty processes decide on values in the range of the non-faulty process inputs.

In contrast to consensus, the unidimensional asynchronous ϵ-approximate agreement is possible, even with (a limited number of) Byzantine failures. We can think of the protocol geometrically: non-faulty processes start on points in $\mathbb{R}$, and then converge to arbitrarily close values in the range of their inputs, despite difficulties presented by asynchrony and malicious behavior of participants.

Specifically, let n be the number of participating processes, and t an upper bound on the number of Byzantine processes. For asynchronous message-passing systems, early protocols required $n > 5t$ [6, 7], improved later to $n > 3t$ [1], which is optimal in terms of resilience [8]. The optimal protocol uses *reliable broadcast* [2, 13, 4] to force Byzantine processes to communicate consistently (avoiding the situation where they tell different things to different processes), as well as the *witness technique* [1] to improve data collection under failures.

[1]Synchrony, randomization, weaker agreement, and others.

STOC'13, June 1-4, 2013, Palo Alto, California, USA.

1.1 Our Contributions

As mentioned before, we present a non-trivial generalization for the ϵ-approximate agreement, allowing inputs and outputs to lie within $\mathbb{R}^m$, for $m \geq 1$. We provide an optimal protocol for asynchronous Byzantine systems. The protocol centers around the *safe area* concept, permitting processes to converge in Euclidean space of arbitrary dimension.

As we have inputs and outputs in $\mathbb{R}^m$, with $m \geq 1$, we now require that all non-faulty processes halt with close values in the *convex hull* of the non-faulty process inputs. We call this requirement *convexity*.

The convexity requirement is the natural generalization of the 1-dimensional validity condition, and it is essential to some applications, as discussed later. The convex hull of the non-faulty process inputs is independent of the choice of coordinate system. Convexity guarantees that if such inputs lie in a linear subspace of $\mathbb{R}^m$ (say, along a line), so do the non-faulty process outputs.

We also require that the protocol performance, measured in the number of messages required to be sent, depends only on the inputs of the non-faulty processes, and is unaffected by any malicious behavior of the Byzantine processes.

While we converge values dimension-by-dimension in our protocol, we *cannot* simply reuse the 1-dimensional protocol for that goal. Fig. 1 shows the problem, taking $m = 2$: with the 1-dimensional algorithm used in consecutive dimensions, we may output any point in the highlighted rectangle, but the actual allowed convex hull is strictly smaller. With the safe area concept, however, we can make sure that all the individual convergence steps maintain convexity (see Sec. 4). A matching lower bound for fault tolerance confirms that the safe area concept, which is fundamental for correctness, permits an optimally-resilient protocol (see Sec. 5).

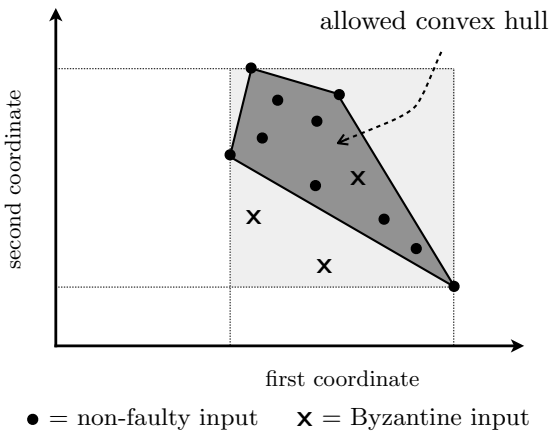

Figure 1: Performing traditional approximate agreement in separate dimensions breaks convexity.

Our contributions are the following. We **generalize** the ϵ-approximate agreement to higher dimensions (Sec. 1.1.1), and discuss its applicability (Sec. 1.1.2). Using the safe area concept (Sec. 3), we **develop a protocol** for Byzantine asynchronous systems (Sec. 4), that we **prove as optimal** (Sec. 5) in terms of resilience: we require $n > t(m + 2)$, for n processes, up to t Byzantine ones, with values taken from $\mathbb{R}^m$. We use some geometric arguments, as Helly's Theorem, to **prove the correctness** of the protocol. We learned recently that similar results were discovered independently and concurrently by Vaidya and Garg [14].

1.1.1 Formal Definition

Consider a set of n participating processes P, including no more than t Byzantine processes. The set of non-faulty (good) processes is called G, and the set of Byzantine (bad) processes is called B.

Every process $p_i \in G$ has an input $I_i \in \mathbb{R}^m$ and an output $O_i \in \mathbb{R}^m$. The set $I_G = \{I_i : p_i \in G\}$ is henceforth called *non-faulty inputs*. After we run the multidimensional ϵ-approximate agreement protocol, we require the following:

Agreement: for any non-faulty processes p_i and p_j, the Euclidean distance between their outputs O_i and O_j is $\leq \epsilon$, an error tolerance fixed *a priori*.

Convexity: for any non-faulty process p_i, its output O_i is in the convex hull of the non-faulty inputs I_G.

Processes communicate asynchronously via message passing. Communication channels are point-to-point reliable (all messages are eventually delivered), complete (any pairwise communication is possible), and FIFO. The processes can reliably identify the sender of any message[2].

1.1.2 Applications

Applications of our protocol include:

Robot convergence: Consider autonomous mobile entities, such as robots, that must converge to nearby locations in the 2 or 3-dimensional space. They must do so despite arbitrary behavior of a subset of robots (Byzantine failures), and unbounded communication delays (asynchronous communication). In other words, they cannot discern between benign and malicious robots, likewise between failed or slowly responsive ones. Finally, non-faulty robots must *respect convexity*, and only move within their original convex region – or they would wander through unsafe territory, say. Byzantine robots must not have the power to influence benign robots to move outside their original convex region.

Previous results in the robot network literature relate approximate agreement with robot convergence in the real line [3, 11]. Our protocol is applicable as long as entities agree on a coordinate system, and the number of faulty robots, t, compared to the number of existent robots, n, is $t < n/4$ for the 2-dimensional case, and $t < n/5$ for the 3-dimensional case. In the terminology of Potop-Butucaru et al. [11], our model is a fully asynchronous, cautious variant of the CORDA model.

Distributed voting: Say distributed voters must choose from a number of options. Each voter gives its relative preferences by assigning *weights* to options, where the weights sum to one. For example, for three options, a voter may give 0.3 option a, 0.6 to option b, and 0.1 to option c. A preference can be viewed as the barycentric coordinates of a point in the triangle of Fig. 2.

As the result of the voting, each process receives a new assignment of weights to options, and all assignments agree to within ϵ, and all lie within the convex hull of

[2]If unavailable, these assumptions could be implemented on top of regular channels – see Sec. 4.

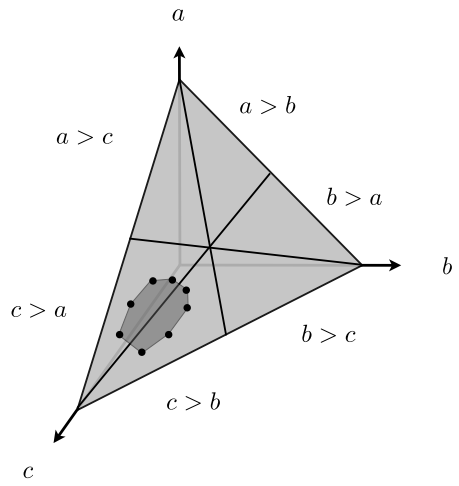

Figure 2: Respecting convexity means respecting unanimity in voting systems.

the original votes. The convexity requirement implicates that the voting respects unanimity: if all voters prefer c over a and b (as in Fig. 2), then every voter's final assignment also reflects that. Here, our protocol is applicable, and respects convexity and unanimity in Byzantine asynchronous systems. Other related interpretations for convexity exist – see Saari [12].

In the introduction we presented motivation, formalization, and application examples for our problem. In Sec. 2, we discuss background material. In Sec. 3, we present the key concept in our protocol: the *safe area*. We formalize the protocol for multidimensional approximate agreement in Sec. 4, which is proved as optimal in terms of fault tolerance in Sec. 5. We analyze message complexity in Sec. 6, and other computational costs in Sec. 7. Our concluding remarks, in Sec. 8, include discussion on possible future work.

2. BACKGROUND

In this section, we review two existing communication primitives, namely the reliable broadcast and the witness technique. Those primitives support our main algorithmic procedure, which is basically as follows. In multiple discrete rounds, any non-faulty processes will:

1. Broadcast its current value;
2. Receive multiple process values (including its own), never waiting for more than $n-t$ process values, since t processes might have crashed;
3. Update its current value to a particular point inside a "safe area" in $\mathbb{R}^m$, guaranteed to be in the convex hull of the non-faulty inputs.

The insight of our protocol lies in step (3), which, despite seemingly simplicity, curtains elaborate combinatorial and geometric arguments for correctness and optimality. Here, we discuss the primitives corresponding to (1) and (2).

2.1 Reliable Broadcast

The reliable broadcast technique avoids the situation when Byzantine processes convey different contents to different processes in a single round of communication. This technique is thoroughly discussed in [2], with original ideas due to Srikanth and Toueg [13] and Bracha [4].

In each round, the messages are decorated with the sender identification, say p, and the current round, say r. So, a message with contents c will look like $M = (p, r, c)$. The reliable broadcast technique has the following properties:

Non-faulty integrity: If a non-faulty process p never reliably broadcasts (p, r, c), no other non-faulty process will ever receive (p, r, c).

Non-faulty liveness: If a non-faulty process p does reliably broadcast (p, r, c), all other non-faulty processes eventually receive (p, r, c).

Global uniqueness: If two non-faulty processes reliably receive (p, r, c) and (p, r, c'), the messages are equal $(c = c')$, even when the sender, p, is Byzantine.

Global liveness: For two non-faulty processes p_1 and p_2, if p_1 reliably receives (p, r, c), p_2 also reliably receives (p, r, c), even when the sender, p, is Byzantine.

Algorithms 1 to 3 illustrate the technique for sender p, round r, and contents c. In summary, (1) p broadcasts a decorated message $M = (p, r, c)$; (2) when other processes receive M, they echo it; (3) when processes receive $n-t$ echo messages for M, they send ready messages for M; (4) when processes see $t+1$ ready messages for M, meaning that a non-faulty process necessarily advocates the existence of M, they also send ready messages; (5) finally, when a process receives at least $n-t$ ready messages, the original message is accepted. For formal proofs of the properties above, please refer to [2].

Algorithm 1 p.`RBSend`$((p, r, c))$

send(p, r, c) to all processes

Algorithm 2 p.`RBEcho`$()$

upon recv(q, r, c) from q **do**
  if never sent $(p, r\{\texttt{echo}\}, \cdot)$ **then**
    send$(p, r\{\texttt{echo}\}, c)$ to all processes
upon recv$(\cdot, r\{\texttt{echo}\}, c)$ from $\geq n - t$ processes **do**
  if never sent $(p, r\{\texttt{ready}\}, \cdot)$ **then**
    send$(p, r\{\texttt{ready}\}, c)$ to all processes
upon recv$(\cdot, r\{\texttt{ready}\}, c)$ from $\geq t + 1$ processes **do**
  if never sent $(p, r\{\texttt{ready}\}, \cdot)$ **then**
    send$(p, r\{\texttt{ready}\}, c)$ to all processes

Algorithm 3 p.`RBRecv`((p,r,c))

recv$(\cdot, r\{\texttt{ready}\}, c)$ from $n - t$ processes
return (p, r, c)

2.2 Witness Technique

To promote agreement, we want, in every round, that the collected values of any two non-faulty processes suitably overlap. However, non-faulty processes cannot wait indefinitely for more than $n - t$ messages, as t processes might

be crashed. If we use reliable broadcast, which indeed never waits for more than $n-t$ processes, any two non-faulty processes will have $n-2t$ values in common after one round of communication. With the witness technique, originally presented by Abraham et al. [1], we can make non-faulty processes have $n-t$ common values, which is essential for our correctness and optimality arguments (Secs. 4.1 and 5, respectively). The witness technique will only wait for messages certain to be delivered.

Algorithm 4 overviews the technique for process p and round r. First, p reliably receives $n-t$ messages from other processes, storing them into *Val*. Then, p reliably transmits its *report*, which contains the $n-t$ messages first collected in *Val*, and reliably receives reports from other processes, storing them into *Rep*.

A *witness* to p is a process whose report consists of messages received by p, either in Line 4 or 8. We note that p collects reports in *Rep* until $n-t$ witnesses are identified in *Wit*. This eventually happens since we are certain to receive $n-t$ non-faulty process reports, and, by the global liveness of reliable broadcast, we are also certain to receive the values received by each of them.

As witnesses are obtained via reliable broadcast, any two non-faulty processes obtain $\geq n-2t \geq t+1$ witnesses in common, of which at least one is non-faulty. So, they receive at least $n-t$ values in common. As formally shown in [1]:

FACT 2.1. *If $n > 3t$, any two non-faulty processes that obtain messages through the witness technique in a single round r obtain $n-t$ messages in common.*

Algorithm 4 p.RBReceiveWitness(r)

```
    Val, Rep, Wit ← ∅
    while |Val| < n − t do
        upon RBRecv((p_x, r, c_x)) do
4:          Val ← Val ∪ {(p_x, r, c_x)}
    RBSend((p, r, Val))
    while |Wit| < n − t do
        upon RBRecv((p_x, r, c_x)) do
8:          Val ← Val ∪ {(p_x, r, c_x)}
        upon RBRecv((p_x, r, Val_x)) do
            Rep ← Rep ∪ {(p_x, r, Val_x)}
        Wit ← {(p_x, r, Val_x) ∈ Rep : Val_x ⊆ Val}
12: return Val
```

3. THE SAFE AREA

As we discussed, non-faulty processes collect messages in multiple discrete rounds. They collect these messages via reliable broadcast and witness technique. We now formalize this scenario, over which we define the safe area concept.

Formally, in every round r, non-faulty processes obtain a *message set*, which contains messages (p_i, r, c_i). For any message set, each message contains the sending process p_i, the current round r, and the message contents c_i (normally values in $\mathbb{R}^m$). No process appears twice, because we use reliable broadcast. So, for any (p_i, r, c_i) and (p_j, r, c_j) in an arbitrary message set, we have that $p_i \neq p_j$.

Consider any message set X. We note that any $X' \subseteq X$ is similarly a message set. We define the non-faulty messages of X as $X_G = \{(p_i, r, c_i) \in X : p_i \in G\}$, and the faulty messages of X as $X_B = \{(p_i, r, c_i) \in X : p_i \in B\}$. We always assume that $|X| > t$ and that $|X_B| \leq t$, as we have no more than t Byzantine processes.

A *restriction of* X is a subset $X' \subset X$ containing exactly $|X|-t$ elements. The set of all possible restrictions is written $\text{Restrict}_t(X)$. The *contents of* X is the multiset $\text{Cont}(X)$ such that $c \in \text{Cont}(X)$ only when $(p, r, c) \in X$, for any process p and round r. If $c \in \text{Cont}(X)$, we can also say that X *contains* c.

DEFINITION 3.1. *For any message set X, if $\text{Cont}(X)$ is comprised of values $\in \mathbb{R}^m$, X is a* valued message set*; if not, X is an* unrestricted message set.

Valued message sets contain only values in $\mathbb{R}^m$. The convex hull of any multiset of values C is denoted by $\text{Poly}(C)$. If X is a valued message set, the *polytope of* X, written $\text{Poly}(X)$, is defined as $\text{Poly}(\text{Cont}(X))$. The *safe area* of X, written $\text{Safe}_t(X)$, is the intersection of the polytopes of all possible restrictions of X (see Fig. 3, and the following definition).

DEFINITION 3.2. *For any valued message set X, the safe area of X is*

$$\text{Safe}_t(X) = \bigcap_{X' \in \text{Restrict}_t(X)} \text{Poly}(X').$$

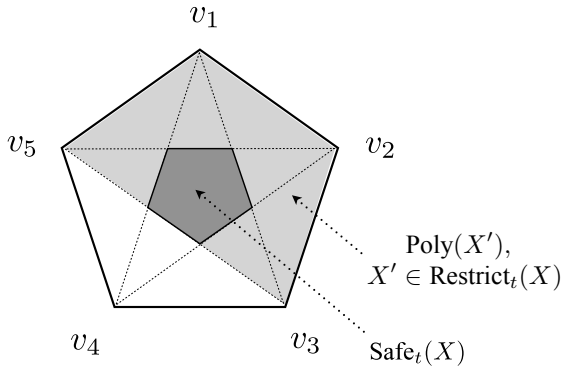

Figure 3: For $\text{Cont}(X) = \{v_1, \ldots, v_5\}$ and $t = 1$, we highlight $\text{Poly}(X')$, for one $X' \in \text{Restrict}_t(X)$, and also show $\text{Safe}_t(X)$. Values are in $\mathbb{R}^2$.

The safe area is a convex polytope, as it is an intersection of other convex polytopes. Also, $\text{Safe}_t(X) \subseteq \text{Poly}(X_G)$, which follows from the lemma below, taking $C = \text{Cont}(X_G)$.

LEMMA 3.3. *If X is a valued message set, $C \subseteq \text{Cont}(X)$, and $|C| \geq |X| - t$, then $\text{Safe}_t(X) \subseteq \text{Poly}(C)$.*

PROOF. As defined before, $\text{Safe}_t(X) \subseteq \text{Poly}(X_C)$, taking a particular restriction X_C of X such that $\text{Cont}(X_C) \subseteq C$. In this case,

$$\text{Safe}_t(X) \subseteq \text{Poly}(X_C) = \text{Poly}(\text{Cont}(X_C)) \subseteq \text{Poly}(C),$$

proving the lemma. □

In the following lemmas, we relate t and $|X|$ in order to ensure the existence of the safe area. We use the following theorem from discrete geometry [5]:

THEOREM 3.4 (HELLY'S THEOREM). *Consider a finite collection of closed convex sets $P = \{P_1, \ldots, P_x\}$ on $\mathbb{R}^m$, with $x \geq m+1$. If every subset of $m+1$ members of P intersect, then*

$$\bigcap_{P_i \in P} P_i \neq \emptyset.$$

The relationship between t and $|X|$, guaranteeing the existence of the safe area, follows in Lemmas 3.6 and 3.10. The next lemma applies to unrestricted message sets; the following lemmas in the current section apply to valued message sets only. In this paper, unless otherwise noted, we presume valued message sets.

LEMMA 3.5. *For any $X_1, \ldots, X_j \in \text{Restrict}_t(X)$, where X is an unrestricted message set,*

$$\left| \bigcap_{1 \leq i \leq j} X_i \right| \geq |X| - jt,$$

for any $j \leq m+1$.

PROOF. We prove the lemma by induction on j.

Base. For $j = 1$, we have that

$$\left| \bigcap_{1 \leq i \leq 1} X_i \right| = |X_1| = |X| - t = |X| - 1t,$$

since $X_1 \in \text{Restrict}_t(X)$.

Induction Hypothesis. Assume valid

$$\left| \bigcap_{1 \leq i \leq k} X_i \right| \geq |X| - kt,$$

for $k < m+1$.

$$\left| \bigcap_{1 \leq i \leq k+1} X_i \right| = \left| \left(\bigcap_{1 \leq i \leq k} X_i \right) \cap X_{k+1} \right| \qquad (\star)$$

$$\geq |X| - kt - t = |X| - (k+1)t \qquad (\star\star).$$

$(\star)$ happens by associativity of $\cap$; $(\star\star)$ happens since $\leq t$ values in $\cap_{1 \leq i \leq k} \text{Cont}(X_i)$ are not in $\text{Cont}(X_{k+1})$. □

LEMMA 3.6. *In $\mathbb{R}^m$, if $|X| > t(m+1)$, then $\text{Safe}_t(X) \neq \emptyset$.*

PROOF. As $|X| > t(m+1)$, by definition of $\text{Restrict}_t(X)$, we know that $|\text{Restrict}_t(X)| \geq m+1$. Therefore, consider any $X_1, \ldots, X_{m+1} \in \text{Restrict}_t(X)$. By Lemma 3.5,

$$\left| \bigcap_{1 \leq i \leq m+1} X_i \right| \geq |X| - (m+1)t \geq (m+1)t + 1 - (m+1)t = 1.$$

So, any $X_1, \ldots, X_{m+1}$ from $\text{Restrict}_t(X)$ will have a non-empty intersection, therefore any $\text{Poly}(X_1), \ldots, \text{Poly}(X_{m+1})$ from $P = \{\text{Poly}(X') : X' \in \text{Restrict}_t(X)\}$ will also have a non-empty intersection. By our first observation, we know that $|P| \geq m+1$.

Finally, as all message contents are in $\mathbb{R}^m$, *all* subsets of $P = \{\text{Poly}(X') : X' \in \text{Restrict}_t(X)\}$ will also have a non-empty intersection, by Helly's Theorem, and therefore $\text{Safe}_t(X) \neq \emptyset$. □

For $v \in R^m$, its projection in coordinate $1 \leq d \leq m$ is denoted by $v(d)$. Now, we characterize special arrangements of values in $\mathbb{R}^m$, used in further discussions and proofs.

DEFINITION 3.7. *A standard basic value $e_d \in \mathbb{R}^m$ is such that $e_d(d) = 2\epsilon$ and $e_d(d') = 0$, considering $1 \leq d \leq m$ and $d' \neq d$. Additionally, $e_0 = 0^m$.*

Note that the distance between any different standard basic values exceeds our threshold ϵ. These special values might be arranged as follows:

DEFINITION 3.8. *In $\mathbb{R}^m$, X configures a (k,t)-simplicial state, for $k \leq m+1$, if X solely contains k different standard basic values, but, for any chosen e_d, $0 \leq d \leq m$, e_d appears $\leq t$ times in $\text{Cont}(X)$.*

We illustrate the arrangement on Fig. 4. There, $4t$ values are located in 4 different positions corresponding to standard basic values, $e_0, \ldots, e_3$, with $\leq t$ values in a single position.

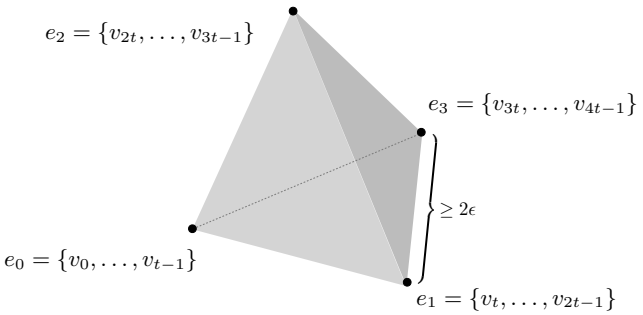

Figure 4: $v_0, \ldots, v_{4t-1} \in \text{Cont}(X)$, which configures a $(4,t)$-simplicial state.

LEMMA 3.9. *In $\mathbb{R}^m$, if X configures some (k,t)-simplicial state, for $k \leq m+1$, then $\text{Safe}_t(X) = \emptyset$.*

PROOF. The arrangement is a geometric $(k-1)$-simplex $\sigma = (x_0, \ldots, x_{k-1})$ [3], with $x_0 \ldots x_{k-1}$ as different standard basic values in $\mathbb{R}^m$. As $\leq t$ members of $\text{Cont}(X)$ are equal to a particular standard basic value x_i, for $0 \leq i \leq k-1$, the safe area is the intersection of proper faces of σ, which is empty. □

LEMMA 3.10. *In $\mathbb{R}^m$, there exists a message set X where $|X| \leq t(m+1)$ and $\text{Safe}_t(X) = \emptyset$.*

PROOF. Say $\text{Cont}(X) = \{x_0, \ldots, x_{t(m+1)-1}\}$, where $x_i = e_{(i \mod m+1)}$. Note that X configures a $(m+1,t)$-simplicial state, so, by the previous lemma, $\text{Safe}_t(X) = \emptyset$. □

In this section, we formalized the concept of the safe area, and now we know that, for any valued message set X, if $|X| > t(m+1)$ its safe area is necessarily nonempty, and if $|X| \leq t(m+1)$ its safe area might be empty. In the next section we present our protocol and proofs of correctness.

4. PROTOCOL

We now present our protocol, in Alg. 5, which tolerates t Byzantine failures in asynchronous systems. In our presentation, we assume FIFO point-to-point channels with reliable message delivery and sender identification. We can simulate FIFO on top of regular channels, if senders put sequential numbers in their messages, and receivers queue messages,

[3]For basic definitions on simplicial topology, see [10].

considering only appropriately numbered ones. Sender identification is also viable using message authentication codes, for instance.

The process input $I \in \mathbb{R}^m$ is passed as argument. On Line 1, `CalculateRounds` takes the input and returns R, the number of rounds needed to converge along each dimension d. The procedure also provides the process a starting point $v \in \mathbb{R}^m$.

Note that although each process' output value is computed dimension-by-dimension, these computations are not really independent. First, both the number of rounds and the starting value are computed by `CalculateRounds` under a holistic approach (see Sec. 4.1.3). Most importantly, the safe area concept allows processes to choose values always within $\text{Poly}(I_G)$, as we formally demonstrate later.

For each dimension, we execute a number of *convergence rounds*, indexed by r, until we accept $> t$ halt messages, accumulated in H. A non-faulty process sends a halt message for the current dimension after R convergence rounds.

Algorithm 5 p.agree(I)

```
(R, v) ← CalculateRounds(I)
for d → 1, ..., m do
    H ← ∅
    r ← 1
5:  while |H| ≤ t do
        RBSend((p, d.r, v))
        upon V ← RBReceiveWitness(d.r) do
            S ← Safe_t(V)
            v ← v ∈ S such that v(d) = Midpoint(S(d))
10:         if r = R then
                RBSend((p, d.r, {halt}))
            r ← r + 1
        upon RBRecv((p', d.r', {halt})), with r' ≥ r do
            H ← H ∪ {(p', d.r', {halt})}
15: return v
```

On Lines 6 and 7, the process transmits its current value to other processes via reliable broadcast, and receives the current values of other processes via the witness technique, which updates V. In Line 8, the safe area is calculated. In Line 9, the process computes the interval $S(d)$, the projection of the safe area on coordinate d, then chooses a point in S such that its d-th coordinate is in the midpoint of $S(d)$, which updates v (see Lemma 4.8). We discuss the relevant properties of the safe area in Sec. 4.1.1.

Note that a non-faulty process accepts only halt messages with an indexed round bigger or equal than the current round. This is essential for convergence, as seen in Sec. 4.1.2 and Sec. 4.1.3. We show that v is well-defined at every round, and correct after all rounds, in Sec. 4.1.4.

Assume we are executing our procedure for dimension d. On process p_i and round r, V_i^r denotes the updated message set in Line 7, S_i^r the updated safe area in Line 8, and v_i^r the updated current value in Line 9. The projections of v_i^r and S_i^r over coordinate d are denoted respectively by $v(d)_i^r$ and $S(d)_i^r$. We omit subscripts or superscripts when they are irrelevant or obvious.

4.1 Proof of Correctness

In this section, we show that the values of the processes converge to a ball in $\mathbb{R}^m$ with radius ϵ.

4.1.1 Intersecting Safe Areas

Say we are executing our procedure for dimension d. For any two non-faulty processes $p_i, p_j \in G$ and convergence round r, define the intersection of their received message sets as $\mathcal{V}_{i,j}^r = V_i^r \cap V_j^r$, written simply as $\mathcal{V}_{i,j} = V_i \cap V_j$ if r is irrelevant or obvious. From fact 2.1:

COROLLARY 4.1. *For any two non-faulty processes, say $p_i, p_j \in G$, we have that $|\mathcal{V}_{i,j}| \geq n - t$.*

As we show next, this implicates in intersecting safe areas between any two non-faulty processes in every round. Formally, we show that, for any non-faulty processes $p_i, p_j \in G$, $S_i \cap S_j \neq \emptyset$.

The reasoning is the following. Since $\mathcal{V}_{i,j} \geq n-t$, the safe area, considering only values in $\mathcal{V}_{i,j} = V_i \cap V_j$, is non-empty. We are interested, however, in the intersection of the safe areas of $V_i \supseteq \mathcal{V}_{i,j}$ and $V_j \supseteq \mathcal{V}_{i,j}$. We show that adding the extra values to the safe area computation at p_i and p_j will maintain a non-empty intersection between $S_i = \text{Safe}_t(V_i)$ and $S_j = \text{Safe}_t(V_j)$.

For that goal, we use the supporting Lemmas 4.2 and 4.3. Consider a valued message set X, with $|X| > t$, and define a valued message as a message containing a value $\in \mathbb{R}^m$.

LEMMA 4.2. $\text{Safe}_t(X) \subseteq \text{Safe}_{t-1}(X)$.

PROOF.

$$\begin{aligned}
\text{Safe}_t(X) &= \bigcap_{X' \in \text{Restrict}_t(X)} \text{Poly}(X') \\
&\subseteq \bigcap_{X' \in \text{Restrict}_t(X)} \left(\bigcap_{M \in X \setminus X'} \text{Poly}(X' \cup \{M\}) \right) \\
&= \bigcap_{X' \in \text{Restrict}_{t-1}(X)} \text{Poly}(X') \quad (\star) \\
&= \text{Safe}_{t-1}(X),
\end{aligned}$$

while $(\star)$ happens since $\cap$ is associative. □

LEMMA 4.3. *For any valued message $M \notin X$,* $\text{Safe}_t(X) \subseteq \text{Safe}_t(X \cup \{M\})$.

PROOF. First, we note that

$$\begin{aligned}
\text{Safe}_t(X) &= \bigcap_{X' \in \text{Restrict}_t(X)} \text{Poly}(X') \\
&\subseteq \bigcap_{X' \in \text{Restrict}_t(X)} \text{Poly}(X' \cup \{M\}) \\
&= \bigcap_{X' \in \text{Restrict}_t(X \cup \{M\}), M \in X'} \text{Poly}(X') = A,
\end{aligned}$$

calling A the intersection of convex polytopes of $|X \cup \{M\}| - t$ members of $X \cup \{M\}$ that *include* M.

Second, by the previous lemma, we note that

$$\begin{aligned}
\text{Safe}_t(X) &\subseteq \text{Safe}_{t-1}(X) \\
&= \bigcap_{X' \in \text{Restrict}_{t-1}(X)} \text{Poly}(X') \\
&= \bigcap_{X' \in \text{Restrict}_t(X \cup \{M\}), M \notin X'} \text{Poly}(X') = B,
\end{aligned}$$

calling B the intersection of convex polytopes of $|X \cup \{M\}| - t$ members of $X \cup \{M\}$ that *exclude* M. Therefore,

$$\begin{aligned}\text{Safe}_t(X) &\subseteq A \cap B \\ &= \bigcap_{X' \in \text{Restrict}_t(X \cup \{M\})} \text{Poly}(X') \\ &= \text{Safe}_t(X \cup \{M\}),\end{aligned}$$

concluding our proof. □

Now, we are ready to prove that, in every round, any two non-faulty processes will have intersecting safe areas. Recall that $n > t(m+2)$.

LEMMA 4.4. *For any non-faulty processes $p_i, p_j \in G$, and any round r, $S_i^r \cap S_j^r \neq \emptyset$*

PROOF. In every round,

$$\begin{aligned}|\mathcal{V}_{i,j}| &\geq n - t && \text{(corollary 4.1)} \\ &\geq t(m+2) + 1 - t \\ &\geq t(m+1) + 1.\end{aligned}$$

By Lemma 3.6, we conclude that $\text{Safe}_t(\mathcal{V}_{i,j}) \neq \emptyset$. By an iterative application of Lemma 4.3, if we incorporate messages besides those of $V_i \cap V_j$, then the safe area for p_i's messages and the safe area for p_j's messages can possibly increase, but never decrease, which gives that $S_i \cap S_j \neq \emptyset$. See Fig. 5. □

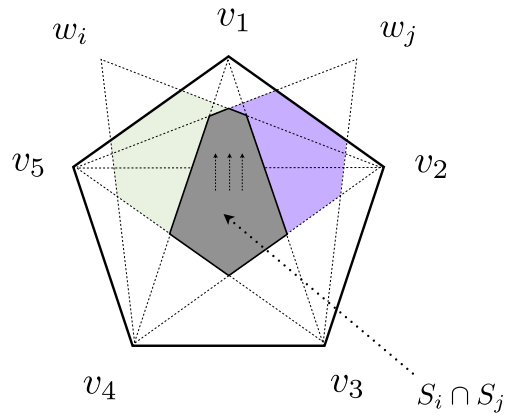

Figure 5: Say $\mathcal{V}_{i,j}$ contains $\{v_1, \ldots, v_5\}$, but V_i also contains $\{w_i\}$ and V_j also contains $\{w_j\}$. The safe area only grows when we also consider w_i at p_i and w_j at p_j, therefore $S_i \cap S_j \neq \emptyset$.

4.1.2 Convergence

If S_i^r and S_j^r intersect, they intersect when projected in any of the m coordinates. Say we are executing our procedure for dimension d.

DEFINITION 4.5. ***Within process*** *p_x, for coordinate d and round r, the lower limit of $S(d)_x^r$ is denoted by $\text{lo}(d)_x^r$, and the upper limit of $S(d)_x^r$ is denoted by $\text{hi}(d)_x^r$.*

DEFINITION 4.6. ***Across non-faulty processes,*** *for coordinate d and round r, $\min(d)^r = \min\{v(d)_x^r : p_x \in G\}$ and $\max(d)^r = \max\{v(d)_x^r : p_x \in G\}$.*

DEFINITION 4.7. ***Across non-faulty processes,*** *for coordinate d and round r, the working range, written $\Delta(d)^r$, is $\max(d)^r - \min(d)^r$.*

We effectively consider `CalculateRounds` as being round 0, so v_i^0 is v as returned by `CalculateRounds` at p_i; the working range $\Delta(d)^0$ is $\max(d)^0 - \min(d)^0$. Between consecutive dimensions, however, consider the values at the end of the previous dimension as being values of round 0.

LEMMA 4.8. *In Line 9, v is well-defined and $v \in S$.*

PROOF. As $|V| \geq n - t$, we know that $S = \text{Safe}_t(V) \neq \emptyset$, by Lemma 3.6. The safe area is convex by definition.

Get two points v' and v'' in S with $v'(d) = \text{lo}(d)$ and $v''(d) = \text{hi}(d)$. Their barycenter is in S, because S is convex, and also satisfies the requirement of the algorithm, since $(v'(d) + v''(d))/2 = \text{Midpoint}(S(d))$. □

LEMMA 4.9. *The working range at the current dimension is halved between consecutive rounds. So, for any $1 \leq d \leq m$ and $r \geq 1$, we have that $\Delta(d)^r \leq \Delta(d)^{r-1}/2$.*

PROOF. Consider any two non-faulty processes p_i and p_j. Without losing generality, say that $v(d)_i^r \geq v(d)_j^r$. However, since $S(d)_i^r \cap S(d)_j^r \neq \emptyset$, there exists some real value $\ell \in S(d)_i^r \cap S(d)_j^r$. Therefore,

$$\begin{aligned}&v(d)_i^{r+1} - v(d)_j^{r+1} \\ &= \frac{\text{lo}(d)_i^r + \text{hi}(d)_i^r}{2} - \frac{\text{lo}(d)_j^r + \text{hi}(d)_j^r}{2} \\ &\leq \frac{\ell + \max(d)^r}{2} - \frac{\min(d)^r + \ell}{2} \\ &= \frac{\max(d)^r - \min(d)^r}{2},\end{aligned}$$

so $\Delta(d)^r$ is halved between rounds. □

If we want processes values to be within ϵ of each other, it is sufficient that they be within $\epsilon/\sqrt{m}$ of each other in every coordinate. We use the argument below.

LEMMA 4.10. *After*

$$\mathcal{R} \geq \log_2\left(\frac{\sqrt{m} \cdot \max\{\Delta(d')^0 : 1 \leq d' \leq m\}}{\epsilon}\right),$$

rounds, the values of the processes are within distance $\epsilon/\sqrt{m}$ of each other at the current dimension d.

PROOF. Since

$$\mathcal{R} \geq \log_2\left(\frac{\sqrt{m} \cdot \max\{\Delta(d')^0 : 1 \leq d' \leq m\}}{\epsilon}\right),$$

we have that

$$\begin{aligned}&2^{\mathcal{R}} \geq \frac{\sqrt{m} \cdot \max\{\Delta(d')^0 : 1 \leq d' \leq m\}}{\epsilon} && \Rightarrow \\ &\frac{\epsilon}{\sqrt{m}} \geq (1/2^{\mathcal{R}}) \cdot \max\{\Delta(d')^0 : 1 \leq d' \leq m\} && \Rightarrow \\ &\frac{\epsilon}{\sqrt{m}} \geq \max\{(1/2^{\mathcal{R}}) \cdot \Delta(d')^0 : 1 \leq d' \leq m\} && \Rightarrow \\ &\frac{\epsilon}{\sqrt{m}} \geq \max\{\Delta(d')^{\mathcal{R}} : 1 \leq d' \leq m\} && \Rightarrow \\ &\frac{\epsilon}{\sqrt{m}} \geq \Delta(d)^{\mathcal{R}},\end{aligned}$$

which satisfies our agreement requirement. □

Note that the number of rounds sufficient for convergence depends only on ϵ and m (naturally), as well on $\Delta(d)^0$, for all $1 \leq d \leq m$. The initial working range $\Delta(d)^0$ is defined to consider non-faulty process values *only*. Next, we see why non-faulty processes run for $\geq \mathcal{R}$ rounds, guaranteeing convergence, while still preventing Byzantine influence.

4.1.3 Initial Estimation of R

In the initial estimation of R, shown in Alg. 6, we use our notion of safe area extensively. We have to make sure that R^i, the estimation of R by process $p_i \in G$, depends *only* on the non-faulty inputs, otherwise Byzantine processes could influence the communication complexity of the protocol.

Algorithm 6 shows the strategy. We obtain $n-t$ witnesses using `RBReceiveWitness`, each surely containing $\leq t$ faulty values. For every witness report, we calculate a safe area, bound to be inside I_G, and obtain its barycenter, defining the multiset U. We calculate a safe area for U, and again its barycenter, defining v. In the algorithm, V is the message set received by `RBReceiveWitness`, and W is a multiset containing witness reports.

The range of values for any multiset C, considering the coordinate d, is defined as

$$\delta_C(d) = \max\{|x(d)^0 - y(d)^0| : x, y \in C\}. \quad (1)$$

Algorithm 6 p.CalculateRounds(I)

```
RBSend((p, 0, I))
(V, W) ← (Val, Cont(Wit)) from RBReceiveWitness(0)
U ← {barycenter of Safe_t(W') : W' ∈ W}
v ← barycenter of Safe_t(U)
R ← ⌈log_2(√m/ε · max{δ_U(d) : 1 ≤ d ≤ m})⌉
return (R, v)
```

In the next two lemmas, we show that the initial values are well-defined and inside the convex hull of non-faulty inputs. We denote W and U within process p_x as W_x and U_x.

LEMMA 4.11. *For any $p_i \in G$, U_i is well-defined and only contains values $\in \text{Poly}(I_G)$.*

PROOF. Consider an arbitrary $W' \in W$. By definition, any report contains exactly $n-t$ values. Therefore,

$$|W'| = n - t \geq t(m+2) + 1 - t = t(m+1) + 1,$$

which gives, by Lemma 3.6, that $\text{Safe}_t(W') \neq \emptyset$. We then conclude that U_i is well-defined.

Note that $W' \subseteq V$ (by definition of witness) and that V contains $\leq t$ values outside I_G. Then, W' also contains $\leq t$ values outside I_G, or, in other words, W' contains at least $|W'| - t$ values inside I_G.

Using Lemma 3.3 , we know that $\text{Safe}_t(W') \in \text{Poly}(I_G)$, which proves that U_i only contains values in $\text{Poly}(I_G)$. □

LEMMA 4.12. *For any non-faulty process $p_i \in G$, we have that v_i^0 is well-defined and $v_i^0 \in \text{Poly}(I_G)$.*

PROOF. Since $|U| \geq n - t$, we use again Lemma 3.6 and see that $\text{Safe}_t(U) \neq \emptyset$. Therefore, v_i^0 is well-defined. Also, since all values in U are in $\text{Poly}(I_G)$, then $v_i^0 \in \text{Poly}(I_G)$. □

In the remaining lemmas, we show that the estimation of R in any non-faulty process guarantees convergence.

DEFINITION 4.13. *The minimum round estimation across non-faulty processes is $\rho = \min\{R_i : i \in G\}$.*

LEMMA 4.14. *For any two non-faulty processes $p_i, p_j \in G$, we have that $U^j \setminus U^i \leq t$.*

PROOF. Non-faulty processes collect $n-t$ witness reports. Moreover, reports are transmitted via reliable broadcast, so $\leq t$ reports in W_j are not in W_i. The result follows since p_i and p_j calculate U identically based on W. □

LEMMA 4.15. *Considering a dimension d, if all non-faulty processes run for $\geq \rho$ rounds,*

$$|v(d)_i^\rho - v(d)_j^\rho| \leq \frac{\epsilon}{\sqrt{m}},$$

for arbitrary non-faulty processes p_i and p_j.

PROOF. Take $p_i \in G$ such that $R_i = \rho$, and consider another arbitrary $p_j \in G$. Defining $D_{j,i} = (U_j \setminus U_i)$, we know that $|D_{j,i}| \leq t$, by Lemma 4.14. Hence, $|U_j \setminus D_{j,i}| \geq |U_j| - t$. Using Lemma 3.3:

$$\begin{aligned} v_j^0 \in \text{Safe}_t(U_j) &\subseteq \text{Poly}(U_j \setminus D_{j,i}) \\ &= \text{Poly}(U_j \setminus (U_j \setminus U_i)) \\ &\subseteq \text{Poly}(U_i). \end{aligned}$$

In conclusion, as p_j was taken arbitrarily, all non-faulty values v_j^0 are inside $\text{Poly}(U_i)$. Noting the calculation of R in Line 5 of `CalculateRounds`,

$$\begin{aligned} R_i &= \lceil \log_2(\sqrt{m}/\epsilon \cdot \max\{\delta_U(d') : 1 \leq d' \leq m\}) \rceil \\ &\geq \log_2(\sqrt{m}/\epsilon \cdot \max\{\delta_U(d') : 1 \leq d' \leq m\}) \\ &\geq \log_2(\sqrt{m}/\epsilon \cdot \max\{\Delta(d')^0 : 1 \leq d' \leq m\}), \qquad (\star) \end{aligned}$$

where $(\star)$ happens since $v_j^0 \in \text{Poly}(U_i)$ for any arbitrary $p_j \in G$. Therefore, if all non-faulty processes run for at least $R_i = \rho$ rounds, we precisely satisfy the condition of Lemma 4.10, and hence the result follows. □

LEMMA 4.16. *Considering a dimension d, all non-faulty processes run for $\geq \rho$ rounds.*

PROOF. Without losing generality, say $p_h \in G$ is a non-faulty process sending $(p_h, d.r_h, \{\texttt{halt}\})$ with earliest round r_h. We know that p_h executed $\geq r_h \geq \rho$ rounds.

Any other non-faulty process $p_x \in G$ either: (1) executes for $R_x \geq \rho$ rounds; or (2) sees $|H| \geq t+1$. If (2), the interesting situation, we know that p_x must have received one halt message from one non-faulty process $p_y \in G$, say $(p_y, d.r_y, \{\texttt{halt}\})$. By definition, $r_y \geq r_h \geq \rho$.

However, $p_x \in G$ only accepts $(p_y, d.r_y, \{\texttt{halt}\})$ if it ran for more than r_y rounds, which we showed to be $\geq \rho$. Since an arbitrary non-faulty process p_x either executes for $R_x \geq \rho$ or for $r_y \geq r_h \geq \rho$ rounds, we are done. □

4.1.4 Putting all Together

In this section, we put previous lemmas together and prove the correctness of our protocol. Define $V_G^r = \{v_i^r : p_i \in G\}$, the set of non-faulty current values at round r.

LEMMA 4.17. *For any $p_i \in G$, it is always the case that v_i is well-defined and $v_i \in \text{Poly}(I_G)$.*

PROOF. We proceed by induction on consecutive rounds. Without losing generality, number all rounds, even the ones

across different dimensions, using consecutive numbers. Fix any non-faulty process $p_i \in G$.

Base. Lemma 4.12 shows that v_i^0 is well-defined and is in $\text{Poly}(I_G)$. As $p_i \in G$ is arbitrary, $V_G^0 \subseteq \text{Poly}(I_G)$.

Induction Hypothesis. Assume that $V_G^x \subseteq \text{Poly}(I_G)$. We know that v_i^{x+1} is well-defined and $v_i^{x+1} \in \text{Safe}_t(V_i^{x+1})$, by Lemma 4.8.

Additionally, V_i^{x+1} contains $\leq t$ values outside V_G^x, since $\leq t$ Byzantine processes are assumed. In light of Lemma 3.3 and our inductive hypothesis, we have that

$$\text{Safe}_t(V_i^{x+1}) \subseteq \text{Poly}(V_G^x) \subseteq \text{Poly}(I_G).$$

As $p_i \in G$ is arbitrary, and $v_i^{x+1} \in \text{Safe}_t(V_i^{x+1})$, as discussed before, we know that $V_G^{x+1} \in \text{Poly}(I_G)$. □

THEOREM 4.18. *After executing the protocol, all values in V_G are within a ball with radius ϵ in R^m and in $\text{Poly}(I_G)$.*

PROOF. For each dimension, non-faulty processes run for $\geq \rho$ rounds (Lemma 4.16), therefore, for any $v_i, v_j \in V_G$, we have that $|v(d)_i^\rho - v(d)_j^\rho| \leq \epsilon/\sqrt{m}$ (Lemma 4.15). Since values are always maintained within $\text{Poly}(I_G)$, by the previous lemma, the result follows. □

5. LOWER BOUND ON RESILIENCE

In this section, we show that our requirement $n > t(m+2)$ is optimal in terms of fault tolerance

THEOREM 5.1. *When $n \leq t(m+2)$, the m-dimensional ϵ-approximate agreement with Byzantine failures is impossible.*

PROOF. We provide a simple counterexample. Consider an execution where all the t Byzantine processes crash before sending messages, and the $n - t$ non-faulty processes, say $p_1, \ldots, p_{n-t} \in G$, start with different standard basic values: say p_i starts with $I_i = e_{i \bmod (m+1)}$.

So, after *any* communication, a non-faulty process p_x can only receive values of non-faulty processes. By hypothesis, all received values are $\in \{e_0, \ldots, e_m\}$, and $\leq t$ received values are $= e_d$, for a chosen $0 \leq d \leq m$.

Therefore, p_x cannot verify that $\text{Poly}(I_G)$ includes anything but I_x, so p_x cannot choose any other value $\in \mathbb{R}^m$ besides its own I_x. For any $0 \leq i \neq j \leq m$, the Euclidean distance between I_i and I_j is $> \epsilon$, since all non-faulty processes start with standard basic values, so any protocol either fails the problem requirements or never terminates. □

This theorem also guarantees that the 1-dimensional approximate agreement, which requires only $n > 3t$, is not enough to solve multidimensional approximate agreement. In higher dimensions, we fundamentally tolerate a smaller fraction of Byzantine processes in order to be solvable. Our protocol, which centers around the safe area concept, solves the problem with optimal fault tolerance.

6. MESSAGE COMPLEXITY

In [2], it is formally shown that a single process spends $O(n^2)$ messages to reliably broadcast a message. In our protocol, non-faulty process reliably broadcast their values in every round, and the witness algorithm (Alg 4) has a single extra reliable broadcast. Therefore, each round of communication requires $O(n^2)$ messages for each non-faulty process.

As the communication channels are FIFO, any non-faulty process executes less than $r_{\max} = \max\{R_i : p_i \in G\}$ rounds for each dimension: before accepting the last halt message from round $r_{\max}$, all others are received, making $|H| \geq t+1$.

For any non-faulty process $p_i \in G$,

$$R_i = \lceil \log_2(\sqrt{m}/\epsilon \cdot \max\{\delta_U(d) : 1 \leq d \leq m\}) \rceil,$$

with $\text{Poly}(U_i) \subseteq \text{Poly}(I_G)$ (Lemma 4.11), which implies that

$$r_{\max} \leq \log_2(\sqrt{m}/\epsilon \cdot \max\{\delta_{I_G}(d) : 1 \leq d \leq m\}) + 1.$$

In conclusion, non-faulty processes run for

$$O(d \log(m/\epsilon \max\{\delta_{I_G}(d) : 1 \leq d \leq m\}))$$

rounds, sending

$$O(n^2 d \log(m/\epsilon \max\{\delta_{I_G}(d) : 1 \leq d \leq m\}))$$

messages in total.

7. SAFE AREA CALCULATION

The distributed computing literature traditionally focuses on communication complexity (number and size of messages exchanged) than on the number of steps of local computation. We believe, however, that the safe area computation is practical for the following reasons.

First, we think of m as constant, and note that $m \leq 3$ in many practical applications. Second, we observe that $\text{Poly}(V)$ and $\text{Safe}_t(V)$ are the intersection of $O(n^m)$ halfspaces, as their facets in $\mathbb{R}^m$ may be defined through $\leq m$ vertices, out of n possible points.

Therefore, when converging on dimension d, we can interpret the halfspaces defining $\text{Safe}_t(V)$ as linear restrictions, and solve two linear programs, one maximizing $v(d)$, and one minimizing $v(d)$. These points are in the safe area, and their barycenter, also in the safe area, could be taken as the updated value v, in agreement with Lemma 4.8.

8. CONCLUSION

In this paper, we give an optimal protocol for multidimensional approximate agreement in Byzantine asynchronous systems. We require $n > t(m+2)$, where n is the number of participating processes, t is the limit on the number of Byzantine processes, and m is the dimension the values lie in, which is optimal, matching a lower bound in terms of resilience. Our lower bound for the number of processes generalizes the previous unidimensional lower bound. In terms of message complexity, non-faulty processes executing our protocol send $O(n^2 d \log(m/\epsilon \max\{\delta(d) : 1 \leq d \leq m\}))$ messages in total, where $\delta(d)$ is the range of non-faulty inputs, when projected at coordinate d.

Previous results in the robot network community relate approximate agreement with robotic convergence in the real line [3, 11]. We formally defined and analyzed our problem in light of distributed computing, but our protocol is suitable to robotic convergence as long as the autonomous agents share a common coordinate system. The safe area concept, however, is independent of common coordinates, which is required in some robot networks. A different strategy for updating v, just after calculating the safe area, could eventually be as well independent of common coordinates, thus being an interesting possibility for future work.

In regard to distributed computing, we particularly indicate that the multidimensional approximate agreement is

a non-trivial generalization of the traditional approximate agreement for Byzantine asynchronous systems, limiting the fraction of Byzantine processes according to the dimension the values lie in. Our protocol solves the problem optimally in terms of fault tolerance, and the safe area concept seems to capture very well the interdependence of dimensions, permitting a systematic convergence of values.

9. REFERENCES

[1] I. Abraham, Y. Amit, and D. Dolev. Optimal resilience asynchronous approximate agreement. In *Principles of Distributed Systems*, volume 3544 of *Lecture Notes in Computer Science*. Springer Berlin / Heidelberg, 2005.

[2] H. Attiya and J. Welch. *Distributed Computing: Fundamentals, Simulations and Advanced Topics (2nd edition)*. John Wiley Interscience, 2004.

[3] Z. Bouzid, M. G. Potop-Butucaru, and S. Tixeuil. Optimal byzantine-resilient convergence in uni-dimensional robot networks. *Theoretical Computer Science*, 411(34-36):3154 – 3168, 2010.

[4] G. Bracha. Asynchronous byzantine agreement protocols. *Information and Computation*, 75(2), 1987.

[5] L. Danzer, B. Grünbaum, and V. Klee. Helly's theorem and its relatives. In *Proceedings of Symposia in Pure Mathematics*. American Mathematical Society, 1963.

[6] D. Dolev, N. Lynch, S. Pinter, E. Stark, and W. Weihl. Reaching approximate agreement in the presence of faults. *Journal of the ACM*, 33:499–516, May 1986.

[7] A. Fekete. Asynchronous approximate agreement. In *Proceedings of the sixth annual ACM symposium on principles of distributed computing (PODC)*, pages 64–76, New York, NY, USA, 1987. ACM.

[8] M. Fischer, N. Lynch, and M. Merritt. Easy impossibility proofs for distributed consensus problems. In *Fault-Tolerant Distributed Computing*, volume 448 of *Lecture Notes in Computer Science*. Springer Berlin / Heidelberg, 1990.

[9] M. Fischer, N. Lynch, and M. Paterson. Impossibility of distributed consensus with one faulty process. *Journal of the ACM*, 32(2):374–382, 1985.

[10] J. Munkres. *Elements of Algebraic Topology*. Prentice Hall, 2 edition, Jan. 1984.

[11] M. Potop-Butucaru, M. Raynal, and S. Tixeuil. Distributed computing with mobile robots: An introductory survey. In *14th International Conference on Network-Based Information Systems (NBiS)*, pages 318 –324, sept. 2011.

[12] D. Saari. *Basic Geometry of Voting*. Springer, 1995.

[13] T. Srikanth and S. Toueg. Simulating authenticated broadcasts to derive simple fault-tolerant algorithms. *Distributed Computing*, 2, 1987.

[14] N. Vaidya and V. Garg. Byzantine vector consensus in complete graphs. *Preprint in arXiv:1302.2543v1*, February 2013.