Validated Byzantine Asynchronous Multidimensional Approximate Agreement

MAYA DOTAN, GILAD STERN, and AVIV ZOHAR, Hebrew University

Consider an asynchronous system where each node begins with some point in $\mathbb{R}^m$. Given some fixed $\epsilon > 0$, we wish to have every nonfaulty node eventually output a point in $\mathbb{R}^m$, where all outputs are within distance $\epsilon$ of each other, and are within the convex hull of the original nonfaulty inputs. This problem, when some of the nodes are adversarial, is known as the "Byzantine Asynchronous Multidimensional Approximate Agreement" problem.

Previous landmark work by Mendes et al. and Vaidya et al. presented two solutions to the problem. Both of these solutions require exponential computation by each node in each round. Furthermore, the work provides a lower bound showing that it is impossible to solve the task of approximate agreement if $n \leq (m + 2)t$, and thus the protocols assume that $n > (m + 2)t$.

We present a Byzantine Asynchronous Multidimensional Approximate Agreement protocol in the validated setting of Cachin et al. Our protocol terminates after a logarithmic number of rounds, and requires only polynomial computation in each round. Furthermore, it is resilient to $t < \frac{n}{3}$ Byzantine nodes, which we prove to be optimal in the validated setting. In other words, working on the task in the validated setting allows us to significantly improve on previous works in several significant metrics. In addition, the techniques presented in this paper can easily yield a protocol in the original non-validated setting which requires exponential computation only in the first round, and polynomial computation in every subsequent round.

CCS Concepts: • Theory of computation → Distributed algorithms;

Additional Key Words and Phrases: approximate agreement, higher dimension, asynchronous, byzantine faults, vector inputs

ACM Reference format:
Maya Dotan, Gilad Stern, and Aviv Zohar. 2022. Validated Byzantine Asynchronous Multidimensional Approximate Agreement. 1, 1, Article 1 (November 2022), 21 pages.
https://doi.org/10.1145/nnnnnn.nnmmnnn
1 INTRODUCTION

One of the elementary problems in the field of distributed computing is the consensus problem. A natural alternative to the agreement property in an asynchronous network is a weaker approximate agreement property, which bypasses the famed FLP [9] impossibility result. In this setting, all nonfaulty nodes are required to output values that are no more than $\epsilon$ apart, for some known $\epsilon$. In order to avoid trivial solutions, all nonfaulty nodes must output some value in the convex hull of the inputs they received. Requiring outputs to be in the convex hull of inputs also yields desirable properties. For example, several parties might want to agree on the value of some given commodity, i.e. output some number that represents its value. If the parties were to use a regular consensus protocol, while starting with different evaluations for the commodity’s value, they would be allowed to output any agreed upon value from the protocol. This is true, even if they originally had very similar evaluations for the commodity’s value. Outputting a value in the convex hull of their inputs guarantees that their output remains “meaningful” by making sure that it is between the highest and lowest input values. This notion can also be generalized to agreeing on vectors in higher dimensions. For example, parties might want to allocate resources between several different avenues. If that is the case, again we would naturally want any agreed upon allocation to be “in between” the originally suggested allocations, i.e. in its convex hull. Furthermore, we might also want to impose an additional validity restriction stating that the agreed upon value is indeed a full allocation of the resources, giving all of the resources, but not more than the original amount.

In 1986, Dolev et al. [7] presented an optimally resilient approximate agreement protocol in the presence of Byzantine faults in synchronous networks. In 2004, Abraham et al. [1] constructed an optimally resilient Approximate Agreement protocol in the presence of Byzantine faults in asynchronous networks for scalar inputs. A natural generalization is Approximate Agreement protocols for inputs that can be multidimensional, i.e. in $\mathbb{R}^m$ for some value $m > 1$. Mendes et al. [13] constructed two protocols solving Byzantine Asynchronous Approximate Agreement, and showed that no such protocol can exist if the number of Byzantine nodes is $t$ for $n \leq t (m + 2)$. They generalize the steps shown in [1] to higher dimension in order to show a possibility result under the aforementioned constraint. They break each round in the algorithm into a step of data collection, including a “witness technique”, and a step of calculating the input for the next round. This general framework proves very useful. Our work uses similar tools, however the calculation of the input for the next round is different. We combine a simple geometric primitive with an enhanced witness technique that guarantees inter-round consistency of votes. This in turn reduces the overall computation from exponential to polynomial in every round, while maintaining the correctness of votes.

Mendes et al.’s work uses the notion of a “safe area” to compute votes in future rounds. Their work provides important insights into the problem, but unfortunately the calculation of a “safe area” requires exponential computation. In [10], Függer et al. present a protocol for Byzantine Asynchronous Multidimensional Approximate Agreement, which makes the communication and round complexity independent of the dimension $m$. Their work changes the voting rule but uses the same notion of a safe area, and thus the computation remains exponential. In all of these works, as the dimension grows, the assumptions on the fraction of faulty nodes becomes more and more stringent. The exponential computation and the assumption on the fraction of faulty nodes severely limit the use of those protocols in real-world applications.

Our main contributions in this paper are twofold. First of all, we present the first Byzantine Asynchronous Multidimensional Approximate Agreement protocol in the validated setting of Cachin et al. [6], which requires only polynomial computation, requires only a logarithmic number of rounds and is resilient to up to $t$ Byzantine nodes such that $n \geq 3t + 1$. Note that this ratio is constant, and does not increase with the dimension $m$, forgoing the strict requirements
Validated Byzantine Asynchronous Multidimensional Approximate Agreement

on the number of faulty nodes implied by the lower bound of Mendes et al. Secondly, we discuss how to use the ideas and techniques presented in the paper in order to construct a protocol in the non-validated setting of Mendes et al., which only requires polynomial computation in the first round and is fully polynomial afterwards. This new algorithm is conceptually simple and computable in polynomial time, requiring only the repeated deletion of the furthest points in a set and computing the average of a set of points.

The validated problem is nearly identical to Byzantine Asynchronous Multidimensional Approximate Agreement, except that it is set in the practical “validated” setting. In this setting there is an additional assumption that there exists some “external validity” function which allows for nodes to check if a value is “valid” as an input. The nodes are then required to output a value in the convex hull of the “valid” inputs. The validated setting suggested in [6] is very natural for real-world applications. Consensus tasks have been extensively researched in this validated setting [2–4, 6, 11], which has proved to be useful and interesting. For example, we can think of the consensus protocol as being run on servers which receive signed inputs from clients, and only correctly signed inputs are considered “valid”. In particular, this setting exists in blockchain systems, where the clients’ signing rights can be seen as an external validity function. Note that in this formulation there is no notion of a “nonfaulty input”, only an input which can be validated. Another type of an external validity function can make sure that the input “makes sense”. Such a function can be formulated from the example of resource allocation shown above. As stated there, when agreeing on the allocation of resources, we want to make sure that they were fully allocated, and that the total allocated amount isn’t greater than the total amount of resources. A fitting external validity function in this case would be to allow only values within the \( m + 1 \)-dimensional simplex. This validity function may be cryptographic or otherwise, and our solution uses the function as a black-box, regardless of the choice of implementation.

The vote () algorithm is easy to understand and implement. While simple, it turns out to be strong enough to replace the prohibitively expensive safe area calculations in the validated setting. Even more surprising is the the fact that it is possible to use only a single round of safe area calculations and use the vote () algorithm in the rest of the protocol, using the original round in order to guarantee validity.

Our protocol requires a number of rounds logarithmic in the diameter of the set of valid inputs and \( \frac{1}{m} \), and each round consists of a constant number of broadcasts by each node. The computation performed in each round is polynomial. This means the message and round complexity is similar to those of the protocol shown in the scalar case [1], and importantly the complexity does not increase with the dimension \( m \). The general structure of the protocol is extremely similar to the protocol of [1], similarly to the protocols of [12–14], but it uses different voting and round approximation rules. Note that the voting and round approximation rules can be replaced with any other rules that guarantees the same properties by just adjusting the way nodes check consistency with previous rounds.

As remarked above, without the use of cryptography, one possible implementation of external validity is using only one exponential time round of Mendes et al.’s safe area calculation [13] which works in the traditional setting, and then continuing with our polynomial-computation protocol. In this solution, the votes are guaranteed to be in the convex hull of nonfaulty inputs after the first round, and since our protocol validates inter-round consistency of the votes, they will continue being in the convex hull of the nonfaulty inputs throughout the rest of the protocol. This means our protocol with slight adjustments can be used in the traditional setting, yielding a fully polynomial solution in all rounds except for the first (exponential) one.
2 MODEL AND BASIC DEFINITIONS

As stated above we will solve the Validate Byzantine Asynchronous Multidimensional ε-Agreement problem. In our model, there are $n$ nodes with point-to-point channels between every pair. The nodes can send messages to each other, which are guaranteed to arrive, but can arrive after any finite time. The adversary can control up to $t < \frac{n}{3}$ nodes, causing them to arbitrarily deviate from the protocol. Classically, Byzantine Asynchronous ε-Agreement is defined as follows:

Definition 2.1. A protocol for Byzantine Asynchronous ε-Agreement over $\mathbb{R}^m$ has the following properties:

- **Termination.** All nonfaulty nodes complete the protocol.
- **Correctness.** Two nonfaulty nodes $i$ and $j$ that complete the protocol output $y_i, y_j \in \mathbb{R}^m$ such that $\text{dist}(y_i, y_j) \leq \varepsilon$.
- **Validity.** Let $x_i$ be node $i$’s input and let $I$ be the set of nonfaulty nodes. The output $y_j$ of a nonfaulty node $j$ is in $\text{conv}(\{x_i | i \in I\})$.

Following ideas presented in [6] and the model described in [3] we assume there exists some external function $\text{ex\_val}()$, which all nonfaulty nodes can run as a black box. The purpose of the function is to determine whether a value is a valid input. In general, we say that a value $v$ is a valid input if $\text{ex\_val}(v) = true$. We think of a setting in which all nodes, including the Byzantine ones, receive valid inputs in $\mathbb{R}^m$. Assuming a function $\text{ex\_val}()$, we define Validated Byzantine Asynchronous ε-Agreement:

Definition 2.2. A protocol for Validated Byzantine Asynchronous ε-Agreement over $\mathbb{R}^m$ has the following properties:

- **Termination.** Assume that nonfaulty nodes receive only valid inputs. Then all nonfaulty nodes complete the protocol.
- **Correctness.** Two nonfaulty nodes $i$ and $j$ that complete the protocol output $y_i, y_j \in \mathbb{R}^m$ such that $\text{dist}(y_i, y_j) \leq \varepsilon$.
- **Validity.** The output $y_j$ of a nonfaulty node $j$ is in $\text{conv}(\{v \in \mathbb{R}^m | \text{ex\_val}(v) = \text{true}\})$.

In exact Byzantine Agreement protocols, a Validity property which states that all nonfaulty nodes output some value $v$ such that $\text{ex\_val}(v) = true$ still allows for trivial solutions, by checking all values and outputting some valid value deterministically. However, exhaustively searching over all valid values in $\mathbb{R}^m$ is impractical so the simple Validity property proposed above is enough to rule out trivial solutions in our setting. Note that if the external validity function can be an arbitrary function $\text{ex\_val}: \mathbb{R}^m \rightarrow \{true, false\}$, then having black-box access to it does not allow for practical ways to find a predefined valid value.

The first part of the paper will contain some basic definitions, and the analysis of the basic procedures we will use throughout this work. In broad strokes, throughout each round of the protocol each node aggregates the points it has seen, and calculates a new point called a “vote” to send in the next round. A vote calculated using these procedures lies in the convex hull of the aggregated points which is crucial for solving the problem of Validated Approximate Agreement. In the initialization round, nodes eliminate extreme points before calculating a vote for the next round of the protocol which lies in the convex hull of the remaining points. Later in the section, we prove some useful properties of these algorithms. We will later use these properties to show that the diameter of the set of nonfaulty votes shrinks exponentially fast.
2.1 Basic Definitions and Algorithms

Definition 2.3 (Diameter). Let \( A \) be a finite set of points. We define the diameter of \( A \) as:
\[
\text{diam} \left( A \right) := \max \{ \text{dist} \left( x, y \right) | x, y \in A \} .
\]

Where the \( \text{dist} \) function is simply the Euclidean distance.

Definition 2.4 (Convex Hull). Let \( A \in \mathbb{R}^n \) be some set. The convex hull of \( A \) is the set of all possible finite convex combinations of points in \( A \). We denote the convex hull of \( A \) by \( \text{conv} \left( A \right) \). So:
\[
\text{conv} \left( A \right) = \{ \sum_{i=1}^{k} a_i x_i | k \in [N] \land \sum_{i=1}^{k} a_i = 1 \land \forall i \in [k] a_i \in A \land \forall i \in [k] 0 \leq a_i \leq 1 \} .
\]

Definition 2.5. Let \( A \) be a finite multiset of points in \( \mathbb{R}^n \). We denote:
\[
\text{furthest} \left( A \right) = \left\langle \max \{ \| x - y \| \} | x, y \in A \right\rangle .
\]

I.e. \( \text{furthest} \left( A \right) \) is the maximal distance pair in \( A \).

Note that \( \text{furthest} \left( A \right) \) is defined to be a pair of points with maximal distance in \( A \). We assume that there is a deterministic tie-breaking rule for ordering points by distance (for instance, the lexicographic tie-breaking rule on the pair of points). Hence from now on when we say "Maximal distance pair" we mean maximal distance according to Euclidean distance and lexicographic tie-breaking, and \( \text{furthest} \left( A \right) \) is defined to use that same tie-breaking rule.

For a set of points \( A \) we define the procedures \( \text{Elim}' \left( A \right) \) and \( \text{vote} \left( A \right) \). The first, \( \text{Elim}' \left( A \right) \) takes a set of points \( A \) in \( \mathbb{R}^n \) and iteratively removes the two maximal-distance pair of points in \( A \) \( t \) times. This can be done by computing the distance between each pair of points, sorting them, and then going over the sorted list and repeatedly removing pairs from which neither point has previously been deleted until \( t \) pairs are deleted. \( \text{Elim}' \left( A \right) \), presented in algorithm 1, is used in the initialization round in order to help nodes estimate the distance between points they have seen in this round and votes of all nodes in the next round. The second procedure, \( \text{vote} \left( A \right) \), which simply computes the average of all points in \( A \). In each round other than the initialization round nodes collect information from other nodes, and then compute their votes for next round using the \( \text{vote} () \) procedure presented in algorithm 2.

Algorithm 1 \( \text{Elim}' \left( A \right) \)

\begin{algorithm}
\textbf{Input:} A set of points \( A \)  
\textbf{Output:} A new set of points \( A' \)
\begin{algorithmic}
\State \textbf{procedure} \( \text{Elim}' \left( A \right) \)
\For {\( i \leftarrow 1, \ldots, t \) do}
\State \( A \leftarrow A \setminus \text{furthest} \left( A \right) \)
\EndFor
\State \textbf{return} \( A \)
\end{algorithmic}
\end{algorithm}

Algorithm 2 \( \text{vote} \left( A \right) \)

\begin{algorithm}
\textbf{Input:} A multiset of points \( A \)  
\textbf{Output:} A point \( o \) in \( \text{conv} \left( A \right) \)
\begin{algorithmic}
\State \textbf{procedure} \( \text{vote} \left( A \right) \)
\State \textbf{return} \( \sum_{x \in A} x \in \text{conv} \left( A \right) \frac{1}{|A|} \)
\end{algorithmic}
\end{algorithm}
2.2 Properties of the Vote Procedure

As stated above, in each round nodes send each other votes and check that they’re consistent with votes from previous rounds. Nodes only accept votes if they’re found to be consistent. In addition, in the initialization round of the protocol, nodes share information in order to approximate the distance between points in the next round. Each node must be able to bound that distance only using the information it sees in the initialization round.

In this subsection we show that the algorithms presented in the previous subsection uphold the following properties:

1. If non-faulty nodes have a “large enough” intersection in the initialization round, then they are able to bound the distance between any two points in the next round, and calculate the number of rounds that they need to run in order to guarantee validated approximate agreement.
2. When nodes check that votes have been computed according to vote (), the diameter of accepted values shrinks by a constant factor in each round, and all non-faulty nodes always vote inside the convex hull of the initial validated set.

This implies that nodes only need to go through a logarithmic number of rounds (in the diameter of the original votes and ) to ensure that all of the votes are at most -distance apart. Intuitively, all nonfaulty nodes need to make sure that each vote was calculated based on a “large enough” set, and that every other vote is based on a set with a “large enough” intersection (the exact definitions of both conditions are stated in the relevant lemmas). We show that if these two conditions hold, the desired multiplicative shrinking takes place.

In all of the lemmas and claims in this section we assume that and .

The first lemma shows that non-faulty nodes (who validated inter-round consistency) always vote inside the convex hull of the nodes validated in the initialization round.

Lemma 2.6. Let be a multiset of points in such that . Let such that . Then vote () ∈ conv () and vote () ∈ conv ()

Proof. Remember that vote () = u∈U 1 |u| u ∈ conv () and that . In addition, , and thus conv () ⊆ conv (). As before, vote () ∈ conv () ⊆ conv () The claim follows.

The following lemma is used in bounding the distance between votes after the initialization round. In the lemma, , are to be thought of as the votes that two nodes receive before completing the initialization round. If a node completes the initialization round while having seen the votes in , it will compute a vote inside the convex hull of for the next round. In this context, the lemma shows that a node receiving the votes in can bound the distance between any of the votes it has seen and the vote of other nodes in the first round, which lie within . This also allows the node to bound the distance between any two nodes’ votes in the next round.

Lemma 2.7. Let be multisets of points in such that |U ∪ V| ≤ n and |U ∩ V| ≥ n − t, then diam (U) ≥ diam (Elim t(V)) and U ∩ Elim t(V) ≠ ∅.

Proof. First note that |V ∩ U| ≥ n − t and thus |U|, |V| ≥ n − t ≥ 2t + 1, meaning that Elim t(V) is well defined and is the result of removing t pairs of points from V. For the first part of the lemma, for every i ∈ [t], denote (pi, qi) to be the pair deleted in the i’th iteration of Elim t(V). For each such i, let di = dist (pi, qi). First of all, d1 ≥ d2 ≥ ... ≥ d1 ≥ diam (Elim t(V)), because in iteration i the furthest distance pair is removed, leaving only points whose distance is no greater than di. If for any i ∈ [t], pi, qi ∈ U, then diam (U) ≥ dist (pi, qi) ≥ diam (Elim t(V)). Otherwise, for every i ∈ [t], either pi ∉ U or qi ∉ U. This means that in total at least different points from V \ U have
be deleted from $V$ throughout $\text{Elim}^t(V)$. By assumption $n \geq |U \cup V| = |U \cap V| + |U \setminus V| + |V \setminus U| \geq n - t + |V \setminus U|$, and thus $t \geq |V \setminus U|$. In other words, all points from $V \setminus U$ have been deleted from $V$ throughout the process of $\text{Elim}^t(V)$, leaving only points from $V \cap U$. Therefore, in this case $\text{Elim}^t(V) \subseteq U \cap V \subseteq U$, and thus $\text{diam}(\text{Elim}^t(V)) \leq \text{diam}(U)$.

For the second part of the lemma, by assumption $|U \cap V| \geq n - t \geq 2t + 1$. Only $2t$ points are removed from $V$ during $\text{Elim}^t(V)$, so $U \cap \text{Elim}^t(V) \neq \emptyset$ as well. □

The next two lemmas are used to show that the diameter of all accepted values shrinks by a factor of 2 in each round.

This means that a logarithmic number of rounds is required to reach the desired diameter. Theorem 2.8 is a technical lemma only to be used in the next lemma and shows that voting according to the set $A$ or the set $A$ with $A$ added to it yield the same vote. In theorem 2.9, read $P$ as the set of all accepted votes in a given round, and $U, V$ as two sets of votes collected by two nodes. In this context, the lemma shows that any two nodes calculating votes according to sets with a large enough intersection will have close votes in the next round. In the protocol, nodes guarantee that any two accepted votes are computed with respect to sets with a large intersection, meaning that the diameter of all accepted votes shrinks by a constant factor in each round.

**Lemma 2.8.** Let $A$ be a multiset of points in $\mathbb{R}^m$, let $l \in \mathbb{N}$ and let $v =$ vote $(A)$. Define $B$ to be the multiset $A$ with the value $v$ added $l$ times. Then vote $(A) =$ vote $(B)$.

**Lemma 2.9.** Let $P$ be a multiset of points in $\mathbb{R}^m$ such that $|P| \leq n$. Let $U, V \subseteq P$ such that $|U \cap V| \geq n - t$. Then $\text{dist}(\text{vote } (U), \text{vote } (V)) \leq \frac{1}{4}\text{diam } (P)$.

The proof of theorems 2.8 and 2.9 is provided in the appendix.

### 3 Validated Byzantine Asynchronous Multidimensional Approximate Agreement

In this section, we take the algorithms from above and construct a Validated Byzantine Asynchronous Multidimensional Approximate Agreement protocol with them. We begin by formally defining the setting of the problem, and then follow with an explicit algorithm for the solution. The protocol follows the general framework of the AAD protocol described in [1], which has also proven to be useful in Mendes et al.’s work [13]. In this framework, the protocol is divided into 3 main conceptual parts.

The first part is an initialization and round-estimation protocol. In this protocol, all nodes collect other nodes’ initial data, estimate how many rounds the protocol must run in order to reach $\varepsilon$-agreement, and then output a vote to suggest in the next vote. In our protocol, the initialization and round-estimation takes place in Algorithm 7, which is called in line 9 of Algorithm 3. After initialization, all nodes participate in a loop consisting of the next two parts. First of all, in the loop each node broadcasts its current vote, and collects votes using an idea called a witness technique. The witness technique is a simple two-round protocol in which all nodes first broadcast votes, wait to receive $n - t$ votes from other nodes, and then broadcast a report of the $n - t$ votes they collected. Once the nodes see that $n - t$ of the reports they received contain all of the values they received, they complete the witness technique. The protocol guarantees that every pair of nonfaulty nodes that complete it have seen at least $n - t$ common values.

The main job of the loop in line 12 is to execute the witness technique in each round, with adjustments made to check that values are consistent with information from previous rounds. Finally, after completing the witness technique in each round, every node computes its vote for next round using our voting rule and starts the next round. In our protocol, this stage takes place in line 23 of Algorithm 3. In the non-validated setting it is crucial that this voting rule
outputs a point in the convex hull of the values received from nonfaulty nodes. This means that if the faulty nodes send values outside the convex hull of the nonfaulty values, these values need to somehow be ignored. Ideally, the voting rule is one such that if all nonfaulty nodes use it to calculate their votes, then the diameter of the convex hull of nonfaulty votes shrinks by a constant multiplicative factor in each round. If that is the case, the exponential shrinking yields a logarithmic round requirement.

The protocol presented in this work slightly adjusts the general framework of the AAD protocol. In the initialization round of the protocol, nodes check whether the suggested values are externally valid. Afterwards, our protocol employs a witness technique similar to the one in AAD, with the added functionality that nodes also check that the values sent in a given round are consistent with values received in previous rounds. More precisely, when sending a vote, nodes must also provide the set of values and reports which were used to compute the vote in the previous round. Before accepting any such message, nodes check that each vote was computed correctly, and recursively check that the values and reports from previous rounds are correct. This allows nodes to guarantee that all of the votes are consistent between rounds. In addition, whereas in the previous protocol only nonfaulty nodes must compute votes based on sets with a large intersection, this adjustment requires faulty nodes to also do so, or have their votes rejected. This process makes sure that only “valid” values are accepted in late rounds, and thus can be thought of as an external validity function for those rounds. This stronger witness technique could be used in additional settings, and we discuss this idea in Section 4. Finally, the voting rule in our protocol is the vote( ) algorithm described in previous sections. The algorithm satisfies the desired property of multiplicative shrinkage between rounds as shown in Lemma 3.9. In the first round external validity is verified. In later rounds inter-round consistency is verified, which guarantees that votes are within the convex hull of the previous round’s votes. Overall this means that votes are guaranteed to be inside the convex hull of the externally-valid values in every round. Algorithm 3 describes the exact behavior of each node. In our solution we assume the existence of broadcast channels for each node. Each node can send a broadcast accompanied by a tag.

Definition 3.1. A broadcast channel has the following properties:

- **Validity.** A nonfaulty node i receives a broadcast from a nonfaulty node j with a given tag, if and only if j sent that broadcast with that tag.
- **Liveness.** If some nonfaulty node i receives a broadcast from node j with a given tag, every nonfaulty node eventually receives that broadcast from j with that tag.
- **Uniqueness.** If two nonfaulty nodes receive two messages m, m’ from the same node with the same tag, then 

In our protocol, each message’s tag is comprised of the current round number and the type of message. The type of message (e.g. a “value” message) is the first element of each broadcast. These channels can be simulated using the information theoretically secure Reliable Broadcast protocol described in [5].

We will start by setting some conventions which will help us discuss the protocol.

Definition 3.2. We say that node i is “in round s” if its local r variable equals s. Furthermore, we say that node i "reached round s” if at any point throughout the protocol its local r variable equals s. Similarly, we say that node i "completed round s” if at any point throughout the protocol its local r variable is greater than s.

Definition 3.3. For every variable x defined in the described algorithms, we define x_i to be node i’s x variable. For example, termination_times_i is node i’s termination_times set.

Manuscript submitted to ACM
Algorithm 3 MultiDimApproxAgree(\(v^0, \epsilon\))

| Line | Code for node \(i\)                                      |
|------|---------------------------------------------------------|
| 1    | global \(r \leftarrow 0\)                              |
| 2    | global \(\text{termination\_times} \leftarrow \emptyset\) |
| 3    | global \(\text{waiting\_values} \leftarrow \emptyset\)  |
| 4    | global \(\text{waiting\_reports} \leftarrow \emptyset\) |
| 5    | global \(\forall r \geq 0 \text{values}^r \leftarrow \emptyset\) |
| 6    | global \(\forall r \geq 0 \text{reports}^r \leftarrow \emptyset\) |
| 7    | global \(\text{halt} \leftarrow \infty\)               |
| 8    | continually run \(\text{process\_messages}\) in the background |
| 9    | call \(\text{init\_node}\)()                           |
| 10   | \(r \leftarrow 1\)                                     |
| 11   | broadcast \("\text{value}^r, v^1, \text{values}^0, \text{reports}^0, 1\)\) |
| 12   | while \(r < \text{halt}\) do                           |
| 13   | for \(M = (\"\text{value}^s, v^s, \text{rec\_vals}^s, \text{rec\_reps}^s, s\)\) \(\in \text{waiting\_val\_messages}\) do |
| 14   | if \(\text{value\_message\_ready}(M)\) then            |
| 15   | \(\text{values}^s\).add\(\(v^s\)\)                     |
| 16   | if \(s = r\) and \(|\text{values}^s| = n - t\) then    |
| 17   | broadcast \("\text{report}, \text{values}^r, r\)\)       |
| 18   | for \(M = (\"\text{report}, \text{rec\_vals}^s, s\)\) \(\in \text{waiting\_reports}\) do |
| 19   | if \(\text{report\_message\_ready}(M)\) then            |
| 20   | \(\text{reports}^s\).add\(\(\text{rec\_vals}^s\)\)       |
| 21   | if \(s = r\) and \(|\text{reports}^r| = n - t\) then    |
| 22   | \(r \leftarrow r + 1\)                                 |
| 23   | \(v^r \leftarrow \text{vote}(\text{values}^{r-1})\)    |
| 24   | broadcast \("\text{value}, v^r, \text{values}^{r-1}, \text{reports}^{r-1}, r\)\) |
| 25   | output \(v^r\) and terminate                           |

Algorithm 4 \(\text{process\_messages}\)()

| Line | Code                                                                 |
|------|----------------------------------------------------------------------|
| 1    | upon receiving \("\text{init\_value}, v\) from node \(j\), do       |
| 2    | if \(\text{ex\_val}(v) = \text{true}\) then                         |
| 3    | \(\text{values}^0\).add\(\(v\)\)                                   |
| 4    | if \(|\text{values}^0| = n - t\) then                                |
| 5    | broadcast \("\text{report}, \text{values}^0, 0\)\)                  |
| 6    | upon receiving \("\text{enough}, e\) from \(j\), do                |
| 7    | \(\text{termination\_times}.add(\(e\)\)\)                         |
| 8    | if \(|\text{termination\_times}| \geq n - t\) then                  |
| 9    | set \(\text{halt}\) to be the \(t + 1\)th smallest value in \(\text{termination\_times}\) |
| 10   | upon receiving \(M = (\"\text{value}, v^r, \text{rec\_vals}^s, \text{rec\_reps}^s, s\)\) from \(j\), do |
| 11   | \(\text{waiting\_values}.add(M)\)                                  |
| 12   | upon receiving \(M = (\"\text{report}, \text{rec\_vals}^s, s\)\) from \(j\), do |
| 13   | \(\text{waiting\_reports}.add(M)\)                                  |

From this point on, our main goal will be to prove this main theorem:
The first set of lemmas shows that in each round some progress is made. In broad strokes, the lemmas show that no nonfaulty node gets stuck: at any point it time it either eventually completes the round it is in, or completes the relevant round reached, enough values were sent, or enough reports were sent.

### Algorithm 5: Value Message Ready

```plaintext
Algorithm 5 \text{value\_message\_ready}(M = (\text{"value"}, v^i, rec\_vals^{s-1}, rec\_reps^{s-1}, s))

1. if the following conditions hold
2. \( r \geq s \), \( \triangleright \) relevant round reached
3. \( |\text{rec\_vals}^{s-1}| \geq n - t \), \( \triangleright \) enough values were sent
4. \( |\text{rec\_reps}^{s-1}| \geq n - t \), \( \triangleright \) enough reports were sent
5. \( \forall v_{\text{vals}} \in \text{rec\_reps}^{s-1}, v_{\text{vals}} \subseteq \text{rec\_vals}^{s-1} \), \( \triangleright \) reports are also in values
6. \( \text{rec\_vals}^{s-1} \subseteq \text{values}^{s-1} \), \( \triangleright \) i accepted the sent values
7. \( \text{rec\_reps}^{s-1} \subseteq \text{reports}^{s-1} \), \( \triangleright \) i accepted the sent reports
8. then
9. if \( s = 1 \) and \( v^j_i \in \text{cons}(\text{Elim}(\text{rec\_vals}^{s-1})) \) then
10. return \text{true}
11. else if \( s > 1 \) and \( v^j_i = \text{vote}(\text{rec\_vals}^{s-1}) \) then
12. return \text{true}
13. return \text{false}
```

### Algorithm 6: Report Message Ready

```plaintext
Algorithm 6 \text{report\_message\_ready}(M = (\text{"report"}, rec\_vals^{s}, s))

1. if the following conditions hold
2. \( r \geq s \), \( \triangleright \) relevant round reached
3. \( |\text{rec\_vals}^{s}| \geq n - t \), \( \triangleright \) enough values were sent
4. \( \text{rec\_vals}^{s} \subseteq \text{values}^{s} \), \( \triangleright \) i accepted the sent values
5. then
6. return \text{true}
7. else
8. return \text{false}
```

### Algorithm 7: Init Node

```plaintext
Algorithm 7 \text{init\_node()}

1. broadcast (\text{"init\_value"}, v^0)
2. while \( |\text{termination\_times}| < n - t \) and \( |\text{reports}^{0}| < n - t \) do
3. for \( M = (\text{"report"}, \text{rec\_vals}^{0}, 0) \in \text{waiting\_reports} \) do
4. if \text{report\_message\_ready}(M) then
5. \text{reports}^{0}.\text{add}(\text{rec\_vals}^{0})
6. if \( |\text{reports}^{0}| = n - t \) then
7. \( v^j_i \leftarrow \text{vote}(\text{Elim}(\text{values}^{0})) \)
8. \( \text{enough} \leftarrow \lceil \log_2(\text{\text{diam}}(\text{values}^{0})/\epsilon) \rceil + 1 \)
9. broadcast (\text{"enough"}, enough)
```

**Theorem 3.4.** \text{MultiDimApproxAgree}(\cdot, \epsilon) is a Validated Byzantine Asynchronous \( \epsilon \)-Agreement protocol over \( \mathbb{R}^m \) resilient to up to \( t = \frac{n-1}{2} \) Byzantine nodes.

This algorithm is optimally resilient and a sketch for a proof of the lower bound can be found in section A of the appendix. In order to prove our main theorem, we will prove several lemmas about the properties of the protocol.

### 3.1 Liveness Lemmas

The first set of lemmas shows that in each round some progress is made. In broad strokes, the lemmas show that no nonfaulty node gets stuck: at any point it time it either eventually completes the round it is in, or completes the relevant round reached, enough values were sent, or enough reports were sent.
protocol. The ideas in the protocol are very similar to the ones in Abraham et al. [1], with adjustments made for the externally valid setting. Therefore, the proofs also follow similar logic, and are provided in the appendix.

**Lemma 3.5 (Eventual Viewpoint Consistency).** Let $i, j$ be two nonfaulty nodes that reach round $r$ of the protocol. Then either (1) from some point on, for every $s \leq r$, $values_i^s \subseteq values_j^s$ and $reports_i^s \subseteq reports_j^s$ or (2) $i$ eventually completes the protocol and terminates.

**Lemma 3.6 (Init Termination).** If all nonfaulty nodes participate in the protocol and have valid inputs, then they all complete init_node.

**Lemma 3.7 (Iterations Eventually Terminate).** If all nonfaulty nodes participate in the protocol, reach round $r$ of the protocol for $r \geq 1$, and none of them terminate during it, then they all complete the round.

### 3.2 Shrinking Diameter Lemmas

The lemmas in this section show that the diameter of the set of all points that nonfaulty nodes receive and add to their respective $values$ sets shrinks by a constant multiplicative in each round. Our strategy will be to use the results from Section 2.2, and show that the local sets that all nonfaulty nodes meet the conditions of the claims we proved there, which allows us to use theorem 2.9. The first lemma shows that using the witness technique guarantees that nodes only accept votes which were computed according to sets with a large intersection.

**Lemma 3.8 (Reported Viewpoints of Accepted Values Are Close).** Let $i, j$ be two nonfaulty nodes and observe some round $r \geq 1$. If node $i$ added $v_p$ to $values_i^r$ as a result of receiving (“value”, $v_p$, $rec_vals_p$, $rec_reps_p$, $r$) from $p$, and node $j$ added $v_q$ to $values_j^r$ as a result of receiving (“value”, $v_q$, $rec_vals_q$, $rec_reps_q$, $r$) from $q$, then $|rec_vals_p \cap rec_vals_q| \geq n - t$.

**Proof.** Nodes $i$ and $j$ added $v_p$ and $v_q$ to $values_i^r$ and $values_j^r$ respectively, so they received

(“value”, $v_p$, $rec_vals_p$, $rec_reps_p$, $r$) and (“value”, $v_q$, $rec_vals_q$, $rec_reps_q$, $r$) messages respectively. Since $i, j$ added those points we know that eventually $rec_reps_p \subseteq reports_i^{r-1}$ and $rec_reps_q \subseteq reports_j^{r-1}$ and that $|rec_reps_p| \geq n - t$, $|rec_reps_q| \geq n - t$. Every element $S \in reports_i^{r-1}$ was added as a result of receiving a “report” message from a unique node $l$ for that round, after checking that $|S| \geq n - t$. Since no node can send two “report” messages per round, this means that the total number of values in $rec_reps_p \cup rec_reps_q$ is no greater than $n$. Now, $2(n - t) \geq n + t + 1$, so $rec_reps_p \cap rec_reps_q \neq \emptyset$. Observe some set $S \in rec_reps_p \cap rec_reps_q$. Seeing as $i, j$ added $v_p$ and $v_q$ to their $values$ sets, it must also be the case that $S \subseteq rec_vals_p \subseteq values_i^{r-1}$ and that $S \subseteq rec_vals_q \subseteq values_j^{r-1}$. As shown above, every such $S$ is of size at least $n - t$, so $|rec_vals_p \cap rec_vals_q| \geq n - t$, because they both contain the set $S$. \hfill \Box

**Lemma 3.9 (Shrinking Diameter).** Let $r \geq 1$ be some round number, and let $G$ be the set of all nonfaulty nodes. Observe the sets $values_i^r$ after the nonfaulty nodes stop adding elements to them. Also, observe the set $values^r = \bigcup_{i \in G} values_i^r$, consisting of all points the nonfaulty nodes ever accept in round $r$. Then:

$$diam\left(values^{r+1}\right) \leq \frac{1}{2} diam\left(values^r\right).$$

\[1\] Each $values$ set can contain no more than $n$ elements, so such a point must exist.
Proof. Observe two points \( v_p, v_q \in \text{values}_{i+1} \). Assume that \( v_p \in \text{values}_{i+1}^{f} \) and \( v_q \in \text{values}_{i+1}^{f} \) for some \( i, j \in G \). Node \( i \) added \( v_p \) to \( \text{values}_{i+1}^{f} \) after receiving the message \("value", v_p, rec_vals_p, rec_reps_p, r + 1\) from \( p \). Similarly, node \( j \) added \( v_q \) to \( \text{values}_{i+1}^{f} \) after receiving the message \("value", v_q, rec_vals_q, rec_reps_q, r + 1\) from \( q \). Seeing as \( i, j \) added the values \( v_p, v_q \) to their respective \( \text{values} \) sets, they found that \( |\text{rec_vals}_p|, |\text{rec_vals}_q| \geq n - t \) and \( v_p = \text{vote}(\text{rec_vals}_p), v_q = \text{vote}(\text{rec_vals}_q) \). Furthermore, \( \text{rec_vals}_p \subseteq \text{values}_{i}^{f}, \text{rec_vals}_q \subseteq \text{values}_{j}^{f} \) and thus \( \text{rec_vals}_p, \text{rec_vals}_q \subseteq \text{values}^{f} \). From Lemma 3.8, we know that \( |\text{rec_vals}_p \cap \text{rec_vals}_q| \geq n - t \). Combining all of those observations we can see that the conditions of Lemma 2.9 hold. Therefore:

\[
\text{dist}(v_p, v_q) = \text{dist}(\text{vote}(\text{rec_vals}_p), \text{vote}(\text{rec_vals}_q)) \\
\leq \frac{1}{2} \text{diam (values)}^{f} .
\]

(1)

Since we dealt with two arbitrary points \( v_p, v_q \in \text{values}_{i+1}^{f} \), we conclude that for every \( r \geq 1 \):

\[
\text{diam (values}^{f + 1}) \leq \frac{1}{2} \text{diam (values}^{f}) .
\]

(2)

3.3 Initial Diameter Approximation Lemmas

The lemmas in this section show that we can use the diameter of every nonfaulty node’s \( \text{values} \) set in the end of \text{init_node} to bound the distance between every two points accepted by any two nonfaulty nodes in round 1. This in turn means that any nonfaulty node’s perceived diameter in round 0 can be used to calculate the number of needed rounds.

Lemma 3.10 (Initial Diameter Approximation). Let \( i, j, k \) be three nonfaulty nodes that started the first round of the protocol. Let \( v_p \in \text{values}^{i} \) and \( v_q \in \text{values}^{j} \). Then \( \text{dist}(v_p, v_q) \leq 3 \text{diam (values}_{k}^{0}) \) at the time \( k \) completes \text{init_node}.

Proof. Since \( v_p \in \text{values}^{i} \), node \( i \) received a \("value", v_p, rec_vals_p, rec_reps_p, 1\) broadcast from node \( p \) such that eventually the message satisfies the conditions of Algorithm 5, meaning that \( \text{rec_reps}_p, \text{reports}_p \subseteq \text{values}_{i}^{f} \) and that \( |\text{rec_reps}_p| \geq n - t \). Observe the sets \( \text{vals}_{k}^{0}, \text{reports}_{k}^{0} \) at the time \( k \) completes \text{init_node}. At that time we know that both sets are at least of size \( n - t \). Applying a counting argument similar to the one in Lemma 3.8 we find that \( |\text{rec_reps}_p \cap \text{reports}_{k}^{0}| \neq 0 \) and thus \( |\text{vals}_p^{0} \cap \text{vals}_{k}^{0}| \geq n - t \).

Nodes only add values to \( \text{vals}_{k}^{0} \) after receiving an \"init_value\" broadcast. Nodes can only send one such broadcast, so the union of all such sets contains \( n \) values at most. Note that \( \text{rec_vals}_p \subseteq \text{vals}_{k}^{0}, \text{rec_vals}_q \subseteq \text{vals}_{k}^{0} \). Finally, \( i \) and \( j \) check that \( v_p \in \text{conv}(\text{Elim}^{f}(\text{rec_vals}_p)) \) and \( v_q \in \text{conv}(\text{Elim}^{f}(\text{rec_vals}_q)) \). Overall, \( \text{vals}_{k}^{0} \cap \text{rec_vals}_p \geq n - t \) and \( |\text{vals}_p^{0} \cup \text{rec_vals}_p| \geq n \). From theorem 2.7, \( \text{diam (vals}_{k}^{0}) \geq \text{diam (Elim}^{f}(\text{rec_vals}_p)) \) and that \( \text{vals}_{k}^{0} \cap \text{Elim}^{f}(\text{rec_vals}_p) \neq \emptyset \). Similarly, \( \text{diam (vals}_{k}^{0}) \geq \text{diam (Elim}^{f}(\text{rec_vals}_q)) \) and \( \text{vals}_{k}^{0} \cap \text{Elim}^{f}(\text{rec_vals}_q) \neq \emptyset \). Set two points \( x \in \text{vals}_{k}^{0} \cap \text{Elim}^{f}(\text{rec_vals}_p) \) and \( y \in \text{vals}_{k}^{0} \cap \text{Elim}^{f}(\text{rec_vals}_q) \). From the convexity of the distance function, \( \text{dist}(x, y) \leq \text{diam (Elim}^{f}(\text{rec_vals}_p)) \) and \( \text{dist}(x, y) \leq \text{diam (Elim}^{f}(\text{rec_vals}_q)) \). In addition, \( x, y \in \text{vals}_{k}^{0} \), so \( \text{dist}(x, y) \leq \text{diam (vals}_{k}^{0}) \). Combining these observations and using the triangle inequality, we get:
we know that all of them complete the \textit{init\_node} call. First we will show that at least one nonfaulty node terminates throughout the protocol. Assume by way of contradiction that none of them do. Using Lemma 3.7 and simple induction we can show that for every \( r \in \mathbb{N} \), all nonfaulty nodes eventually reach round \( r \). Note that at the time \( i \) completes \textit{init\_node}, \(|\text{termination\_times}| \geq n - t \) and thus \( \text{halt}_i \) is well defined. Since the \( t + 1 \)'th smallest element in \textit{termination\_times} is monotonically decreasing (adding elements cannot increase the \( t + 1 \)'th smallest values), as \( i \) continues participating in the protocol \( \text{halt}_i \) can only become smaller. That means that eventually, every nonfaulty node \( i \) will see that \( r \geq \text{halt}_i \) and terminate.

Now assume some nonfaulty node terminates. Let \( s \) be the smallest round number such that some nonfaulty node terminates in round \( s \), and let \( j \) be such a nonfaulty node. Now assume by way of contradiction that some nonfaulty node \( i \) never terminates. Since no nonfaulty node terminates before round \( s \), from Lemma 3.7 we know that all nonfaulty nodes eventually reach it. Node \( j \) terminated in round \( s \), which means that at the time it terminated, it must have found that \( s \geq \text{halt}_j \). Eventually, \( i \) will receive all of the ("enough", \( \epsilon \)) messages that \( j \) received, and therefore at that time \textit{termination\_times} \( j \subseteq \text{termination\_times} \). Since the \( t + 1 \)'th smallest element is monotonically decreasing, at that time \( s \geq \text{halt}_j \geq \text{halt}_i \) and \( i \) will terminate.

\textbf{Validity.} Let \( C = \text{conv} \{(v \in \mathbb{R}^m | \text{ex\_val}(v) = \text{true})\} \). We will prove by induction that for every nonfaulty \( i \) and \( r \geq 0 \), \( \text{values}^r_i \subseteq C \) and \( v_{i+1}^r \in C \). For \( r = 0 \), let \( u \in \text{values}^0_i \). Node \( i \) would have only added \( u \) to \( \text{values}^0_i \) if \( \text{ex\_val}(u) = \text{true} \) and thus \( u \in C \). After completing \textit{init\_node}, \( v_1^i = \text{vote} \left( \text{Elim}^i(\text{values}^0_i) \right) \). Note that \( \text{Elim}^i(\text{values}^0_i) \subseteq \text{values}^0_i \). From theorem 2.6, vote \( \left( \text{Elim}^i(\text{values}^0_i) \right) \in \text{conv} \left( \text{values}^0_i \right) \).

Next, we will prove the case for \( r = 1 \). Observe any \( v_j \in \text{values}^1_i \). Node \( i \) would have only added the value to \( \text{values}^1_i \) if it received ("value", \( v_j \), \( \text{rec\_vals}_i \), \( \text{rec\_reps}_i \), \( r \)) from \( j \), and eventually found that \( v_j \in \text{conv} \left( \text{Elim}^i(\text{rec\_vals}_i) \right) \) and that \( \text{rec\_vals}_i \subseteq \text{values}^0_i \). As shown above, \( \text{values}^0_i \subseteq C \), and since \( \text{Elim}^i() \) only removes points, \( \text{Elim}^i(\text{rec\_vals}_i) \subseteq \text{rec\_vals}_i \). Therefore, \( v_j \in \text{conv} \left( \text{Elim}^i(\text{rec\_vals}_i) \right) \subseteq \text{conv} \left( \text{values}^0_i \right) \subseteq C \). In addition, \( i \) computes \( v_{i+1}^r = \text{vote} \left( \text{values}^r_i \right) \).

From theorem 2.6, \( v_{i+1}^r \in \text{conv} \left( \text{values}^r_i \right) \subseteq C \).

Now observe some \( r > 1 \), and assume the claim holds for \( r - 1 \). Observe any \( v_j \in \text{values}^r_i \). Since \( i \) added \( v_j \) to \( \text{values}^r_i \), it received ("value", \( v_j \), \( \text{rec\_vals}_i \), \( \text{rec\_reps}_i \), \( r \)) from \( j \), and eventually found that \( \text{rec\_vals}_i \subseteq \text{values}^{r-1}_i \).
and \( v_j = \text{vote}(\text{rec_vals}_j) \). From theorem 2.6, \( v_j \in \text{conv}(\text{rec_vals}_j) \subseteq \text{conv}(\text{values}_i^{r-1}) \subseteq C \). Finally, \( i \) computes \( v_i^{r+1} = \text{vote}(\text{values}_i^r) \). Again, following the same logic, we find that \( v_i^{r+1} \in C \) as required.

If a nonfaulty node \( i \) terminated, it must have first completed \( \text{init}_i \), computed \( v_i^1 \) and set \( r = 1 \). Throughout the protocol, \( i \) only increments \( r \), so when it completes the protocol, it outputs \( v_i^r \) for \( r \geq 1 \). As shown above, for every \( r \geq 1 \) \( v_i^r \in C \), as required.

**Correctness.** For every nonfaulty \( i \) and round \( r \geq 1 \) we will observe the set \( \text{values}_i^r \) at a point in time where \( i \) doesn’t add any more values to it (i.e., at a time where it is maximal in size). In addition, for every nonfaulty \( i \) we will define \( \text{values}_i^{\text{init}} \) to be the set \( \text{values}_i^0 \) at the time \( i \) computes \( \text{enough}_i \) in line 8 of \( \text{init}_i \). Define \( G \) to be the set of nonfaulty nodes and for every \( r \geq 1 \) define \( \text{values}^r = \bigcup_{i \in G} \text{values}_i^r \). First we would like to show that for every nonfaulty \( i \), 
\[
3\text{diam}(\text{values}_i^{\text{init}}) \geq \text{diam}(\text{values}_i^1).
\]

Observe the values \( v_p, v_q \in \text{values}_i^1 \) such that \( \text{dist}(v_p, v_q) = \text{diam}(\text{values}_i^1) \). Assume that \( v_p \in \text{values}_j^1 \) and \( v_q \in \text{values}_k^1 \) for some \( j, k \in G \). By Lemma 3.10 we know that \( \text{diam}(\text{values}_i^1) = \text{dist}(v_p, v_q) \leq 3\text{diam}(\text{values}_i^{\text{init}}) \).

We now turn to observe the sets \( \text{values}^r \) for every \( r \geq 1 \). Using Lemma 3.9, we know that for every \( r \geq 1 \):
\[
\text{diam}(\text{values}^{r+1}) \leq \frac{1}{2} \text{diam}(\text{values}^r).
\]

Combining the two previous observations, we now know that for every nonfaulty \( i \) and round \( r \geq 1 \):
\[
\frac{1}{2}(r-1) \cdot 3\text{diam}(\text{values}_i^{\text{init}}) \geq \frac{1}{2}(r-1) \cdot \text{diam}(\text{values}_i^1) \geq \text{diam}(\text{values}_i^r).
\]

Setting \( r \geq \text{enough}_i \), we find that:
\[
\text{diam}(\text{values}_i^r) \leq 3\text{diam}(\text{values}_i^{\text{init}}) \cdot \frac{1}{2}(r-1)
\]
\[
\leq 3\text{diam}(\text{values}_i^{\text{init}}) \cdot \frac{1}{2}(\log_3(3\text{diam}(\text{values}_i^{\text{init}})) + 1 - 1)
\]
\[
\leq 3\text{diam}(\text{values}_i^{\text{init}}) \cdot \frac{1}{2}(\log_3(3\text{diam}(\text{values}_i^{\text{init}})) + 1)
\]
\[
= 3\text{diam}(\text{values}_i^{\text{init}}) \cdot \frac{\varepsilon}{3\text{diam}(\text{values}_i^{\text{init}})} = \varepsilon.
\]

Finally, observe the smallest round \( s \) in which some nonfaulty node terminates, and let \( i \) be such a node. First, clearly if all nonfaulty nodes would have sent their values in round \( s \), they would all eventually accept those values and add them to their respective \( \text{values}_i^s \) sets. In that case all of the arguments above would also apply to those values. This means that we can consider all of the values \( v_i^s \) output by nonfaulty nodes to be part of \( \text{values}_i^s \). Observe \( \text{halt}_i \) at the time \( i \) terminates. As shown in the proof of the Validity property, we know that for some nonfaulty node \( j \), \( s \geq \text{halt}_j \geq \text{enough}_j \). From our previous observation, we know that \( \text{diam}(\text{values}_j^s) \leq \varepsilon \). Furthermore, we’ve also shown in the proof of the Validity property that for every nonfaulty \( i \) and round \( r \geq 1 \), \( \text{values}_i^r \subseteq \text{conv}(\text{values}_i^{r-1}) \), which means that also \( \text{values}_i^r \subseteq \text{conv}(\text{values}_i^{r-1}) \). In other words all nonfaulty nodes output values in \( \text{conv}(\text{values}_i^s) \), and thus all of their outputs are no more than \( \varepsilon \) apart.

**Theorem 3.11.** Let \( V = \{v \in \mathbb{R}^m | \text{ex_val}(v) \} \) be the set of all valid inputs. Every nonfaulty node runs for \( O\left(\log\left(\frac{\text{diam}(V)}{\varepsilon}\right)\right) \) rounds.
PROOF. Observe some nonfaulty node \( i \). At the time \( i \) completes \( \text{init} \), \( |\text{termination\_times}_{i}| \geq 2t + 1 \). At that time \( \text{halt}_{i} \) is defined to be the \( t + 1 \)'th smallest value in \( \text{termination\_times}_{i} \). Since there are \( n - t \geq 2t + 1 \) values in \( \text{termination\_times}_{i} \), there are at least \( t + 1 \) values \( e \in \text{termination\_times}_{i} \) such that \( \text{halt}_{i} \leq e \). There are only \( t \) faulty nodes, so \( \text{halt}_{i} \leq e \) for a value \( e \) that some nonfaulty \( j \) sent. Node \( j \) sends the value \( \text{enough}_{j} = \lceil \log_{2} (3 \text{diam} (\text{values}_{j}) / e) \rceil + 1 \). As shown in the proofs of the Correctness and Validity properties of \( \text{MultiDimApproxAgree} \), \( \text{halt}_{i} \) can only become smaller throughout the algorithm, and \( \text{values}_{j} \subseteq V \). This leads us to conclude that \( i \) terminates after seeing that \( r \geq \lceil \log_{2} (3 \text{diam} (\text{values}_{j}) / e) \rceil + 1 \), and thus won’t run for more than \( \lceil \log_{2} (3 \text{diam} (V) / e) \rceil + 1 \) rounds. \( \square \)

4 CONCLUSIONS AND FUTURE WORK

In this work we formalized the task of Validated Byzantine Asynchronous \( \epsilon \)-Agreement over \( \mathbb{R}^{m} \), provided an efficient protocol that solves it, and proved it is optimally resilient. In future works we would like to further explore the task in the traditional setting (i.e. in which nodes output values in the convex hull of the nonfaulty inputs) and check if our method can help solve the problem efficiently. We believe that our extended witness technique is also relevant to the traditional variant of the problem. Using this technique, hybrid versions of the Mendes-Herlihy and Vaidya-Garg algorithms [13] and our method can be used to derive more computationally efficient solutions by running one initialization round which requires exponential computation. In this round, nodes will compute the Safe Area as described in [13]. Each output of that computation is in the convex hull of the nonfaulty inputs. After the initialization round, nodes can check inter-round consistency using our protocol, which will prevent faulty nodes from reporting values outside of the convex hull of nonfaulty inputs. By switching over to our validated solution, the protocol still requires a logarithmic number of rounds, with only the first one requiring exponential computation. In fact there may be many ways of running an initialization round, and we think this is an interesting future direction.

Another avenue of future research could involve also relaxing the validity property of the protocol. In addition to only requiring \( \epsilon \)-Correctness, we could conceivably require only \( \delta \)-Validity. By this we mean that nodes aren’t required to output values in the convex hull of nonfaulty inputs, but to output values that are close to nonfaulty values. One way to formalize this is by defining \( V \) to be the set of all nonfaulty inputs, and allowing parties to output values in a ball of diameter \( \delta \cdot \text{diam} (V) \) which contains all points in \( V \). Note that this condition still rules out trivial solutions by requiring the ball to contain the values in \( V \). The \( \text{MultiDimApproxAgree} \) protocol also achieves \( 3 \)-Validity in the traditional setting, i.e. when ignoring the external validity function (or setting it to always return \( \text{true} \)). This can be shown by proving the following informally stated lemma:

**Lemma 4.1.** Let \( V \) be the set of all nonfaulty inputs. When executing \( \text{MultiDimApproxAgree} \) with \( \text{ex\_val}() \) always returning \( \text{true} \), if a nonfaulty node accepts a value \( v \) in round 1, then there exists some nonfaulty input \( x \in V \) such that \( \text{dist} (v, x) \leq \text{diam} (V) \).

The lemma is formally stated and proved in the appendix. Now we know that in the first round of the protocol, all values accepted by all nonfaulty nodes are of distance \( \text{diam} (V) \) or less from the convex hull of nonfaulty inputs, which means that they are within a ball of diameter \( 3 \text{diam} (V) \) which contains \( V \). Following the proof in the rest of the paper, we can now conclude that every nonfaulty node’s output will be within that ball\(^2\). As far as we know, this is the first

---

\(^2\)As currently stated, the adversary can choose very distant points in the initialization round, inflating the number of rounds required. In order to mitigate this, two initialization rounds can be used instead: the first resulting in the distance between accepted points being no more than \( 3 \text{diam} (V) \), and the second used to compute the number of required rounds by approximating the diameter of the resulting ball.
formulation of such a property and future research might lead to a much smaller $\delta$ by using different techniques in the initial rounds.

REFERENCES

[1] Ittai Abraham, Yonatan Amit, and Danny Dolev. Optimal resilience asynchronous approximate agreement. In *International Conference On Principles Of Distributed Systems*, pages 229–239. Springer, 2004.

[2] Ittai Abraham, Philipp Jovanovic, Mary Maller, Sarah Meiklejohn, Gilad Stern, and Alin Tomescu. Reaching consensus for asynchronous distributed key generation, 2021. arXiv:2102.09641.

[3] Ittai Abraham, Dahlia Malkhi, and Alexander Spiegelman. Asymptotically optimal validated asynchronous byzantine agreement. In *Proceedings of the 2019 ACM Symposium on Principles of Distributed Computing*, PODC ’19, pages 337–346, New York, NY, USA, 2019. Association for Computing Machinery. URL: https://doi.org/10.1145/3293611.3331612.

[4] Ittai Abraham and Gilad Stern. Information theoretic hotstuff. In *OPODIS*, volume 184 of LIPIcs, pages 11:1–11:16, Dagstuhl, Germany, 2020. Schloss Dagstuhl - Leibniz-Zentrum für Informatik.

[5] Gabriel Bracha. Asynchronous byzantine agreement protocols. *Inf. Comput.*, 75(2):130–143, November 1987. URL: http://dx.doi.org/10.1016/0890-5401(87)90054-X.

[6] Christian Cachin, Klaus Kursawe, Frank Petzold, and Victor Shoup. Secure and efficient asynchronous broadcast protocols. In *Proceedings of the 21st Annual International Cryptology Conference on Advances in Cryptology, CRYPTO ’01*, pages 524–541, Berlin, Heidelberg, 2001. Springer-Verlag.

[7] Danny Dolev, Nancy A. Lynch, Shlomit S. Pinter, Eugene W. Stark, and William E. Weihl. Reaching approximate agreement in the presence of faults. *J. ACM*, 33(3):499–516, May 1986. URL: https://doi.org/10.1145/5925.5931.

[8] Cynthia Dwork, Nancy Lynch, and Larry Stockmeyer. Consensus in the presence of partial synchrony. *J. ACM*, 35(2):288–323, April 1988. URL: https://doi.org/10.1145/42282.42283.

[9] Michael J. Fischer, Nancy A. Lynch, and Michael S. Paterson. Impossibility of distributed consensus with one faulty process. *J. ACM*, 32(2):374–382, April 1985. URL: https://doi.org/10.1145/3149.214121.

[10] Matthias Függer and Thomas Nowak. Fast multidimensional asymptotic and approximate consensus. *arXiv preprint arXiv:1805.04923*, 2018.

[11] Yuan Lu, Zhenliang Lu, Qiang Tang, and Guiling Wang. Dumbo-mvba: Optimal multi-valued validated asynchronous byzantine agreement, revisited. In *Proceedings of the 39th Symposium on Principles of Distributed Computing*, PODC ’13, pages 65–73, New York, NY, USA, 2013. Association for Computing Machinery. URL: https://doi.org/10.1145/2484239.2484256.
5 APPENDIX

A LOWER BOUND SKETCH

It is natural to ask whether the presented protocol is optimally resilient, i.e., whether there could exist a Validated Byzantine Asynchronous Multidimensional $\epsilon$-Agreement protocol resilient to $t$ Byzantine nodes such that $n \leq 3t$. It is important to state that [13] proves a lower bound showing that there does not exist an approximate agreement protocol over $\mathbb{R}^m$ resilient to any $t$ such that $n \leq (m + 2)$. The protocol presented in this paper manages to circumvent this lower bound by assuming an external validity function and only requiring valid outputs to be within the convex hull of the valid inputs. We can show that no Validated Byzantine Asynchronous Multidimensional $\epsilon$-Agreement protocol can be resilient to $t \geq \frac{n}{2}$ fault nodes using ideas from [8].

Set some $t, n$ such that $n \leq 3t$. In addition, set some $\epsilon$ and $m$. Assume by way of contradiction that there exists some Validated Byzantine Asynchronous $\epsilon$-Agreement protocol over $\mathbb{R}^m$ resilient to $t$ faults. Uniformly and independently sample two values $v_1 \in (0, 1)^m$ and $v_2 \in (\epsilon + 1, \epsilon + 2)^m$ and define $\text{ex}_\text{val}(\cdot)$ as follows:

$$
\text{ex}_\text{val}(x) = \begin{cases} 
\text{true} & x \in \{v_1, v_2\} \\
\text{false} & \text{else}
\end{cases}.
$$

Let $B \subseteq [n]$ be some set of nodes of size $t$ that the adversary will control. If $n - t = 1$, the adversary will control only $t - 1$ nodes instead. Divide the remaining nodes into two sets $G_1$ and $G_2$ of sizes $\left\lfloor \frac{n-t}{2} \right\rfloor$ and $\left\lceil \frac{n-t}{2} \right\rceil$ respectively. All nodes in $G_1$ will receive $v_1$ as input and all nodes in $G_2$ will receive $v_2$ as input and run the protocol. All nodes in $B$ will communicate with nodes in $G_1$ as nonfaulty nodes would with input $v_1$. Similarly all nodes in $B$ will communicate with nodes in $G_2$ as nonfaulty nodes with input $v_2$. All messages between nodes in $B$ and $G_1$ and between nodes in $B$ and $G_2$ are delivered instantly. On the other hand, all communication between nodes in $G_1$ and $G_2$ is delayed until all of those nodes complete the protocol. Technically, this scheduling might not be valid if those nodes never complete the protocol, but we will show that this is not the case. From the point of view of all nodes in $G_1$, any execution of the protocol in this setting indistinguishable from a setting where the adversary controls all nodes in $G_2$ and instructs them to be silent. If the adversary controls the nodes in $G_2$ and the rest of the nodes communicate freely, the nodes in $G_1$ must complete the protocol. Since those two settings are indistinguishable to nodes in $G_1$, they must do complete protocol if the adversary controls the nodes in $B$ as well. The same argument can be made for nodes in $G_2$. Now, from the point of view of nodes in $G_1$, this setting is indistinguishable from a setting in which the only valid input is $v_1$. This is because the only other valid value was uniformly and independently sampled from $(\epsilon + 1, \epsilon + 2)^m$. The probability that any given value is chosen is 0, so there is no way for any node in $G_1$ to find out which other value was sampled (or even whether there exists any other value $v \in \mathbb{R}^m$ such that $\text{ex}_\text{val}(v) = \text{true}$). From the Validity property, every node in $G_1$ must then output the value $v_1$. Using the exact same arguments, every node in $G_2$ must output the value $v_2$. Note that from the way $v_1$ and $v_2$ were sampled we know that $\text{dist}(v_1, v_2) > \epsilon$, reaching a contradiction to the Correctness property of the protocol.

B PROOFS FOR TECHNICAL CLAIMS AND LEMMAS

Lemma B.1. Let $A$ be a multiset of points in $\mathbb{R}^m$, let $l \in \mathbb{N}$ and let $v = \text{vote}(A)$. Define $B$ to be the multiset $A$ with the value $v$ added $l$ times. Then $\text{vote}(A) = \text{vote}(B)$. 

Manuscript submitted to ACM
Proof. Let \(|A| = k\). The set \(B\) is the multiset \(A\) with \(v\) added \(l\) times, so \(|B| = k + l\). Then:

\[
vote(B) = \frac{1}{k + l} \sum_{u \in B} u = \frac{1}{k + l} (l \cdot v + \sum_{u \in A} u)
\]
\[
= \frac{1}{k + l} (l \left( \frac{1}{k} \sum_{u \in A} u \right) + \frac{k}{k} (\sum_{u \in A} u))
\]
\[
= \frac{1}{l + k} (l + k) \left( \frac{1}{k} \sum_{u \in A} u \right) = \frac{1}{k} \sum_{u \in A} u = vote(A)
\]

\(\square\)

Lemma B.2. Let \(P\) be a multiset of points in \(\mathbb{R}^m\) such that \(|P| \leq n\). Let \(U, V \subseteq P\) such that \(|U \cap V| \geq n - t\). Then \(dist(vote(U), vote(V)) \leq \frac{1}{2} diam(P)\).

Proof. Let \(k = |U \setminus V|\) and \(l = |V \setminus U|\). Note that:

\[
n \geq |U \cup V| = |U \cap V| + |U \setminus V| + |V \setminus U| \geq n - t + k + l
\]

and thus \(t \geq k + l\). Define \(U'\) to be the multiset \(U\) with the value \(vote(U)\) added \(l\) times and similarly define \(V'\) to be the multiset \(V\) with the value \(vote(V)\) added \(k\) times. Note that now:

\[
|U'| = |U| + l = |U \cap V| + |U \setminus V| + l = |U \cap V| + k + l
\]

and similarly, \(|V'| = |U \cap V| + k + l\), and thus \(|V'| = |U'|\). Also, \(U \cap V \subseteq U \subseteq U'\), and similarly \(U \cap V \subseteq V'\). This means that both \(U'\) and \(V'\) each consist of all elements in \(U \cap V\), and each of the sets includes \(k + l\) additional elements (some of which might be shared). Let \(U' \setminus (U \cap V) = \{u_1, \ldots, u_{k+l}\}\) and \(V' \setminus (U \cap V) = \{v_1, \ldots, v_{k+l}\}\). Note that for every set \(A\), \(vote(A) \in conv(A)\). By definition, for every \(u \in U, v \in V\), \(dist(u, v) \leq diam(P)\). Distance is a convex function in each of its arguments, so for every \(u \in U', v \in V'\) it is also true that \(dist(u, v) \leq diam(P)\).
From theorem 2.8, vote \((U) = \text{vote} (U')\) and vote \((V) = \text{vote} (V')\), so dist \((\text{vote} (U), \text{vote} (V)) = \text{dist} (\text{vote} (U'), \text{vote} (V'))\).

Combining the previous observations:

\[
\text{dist} (\text{vote} (U'), \text{vote} (V')) = \left\| \text{vote} (U') - \text{vote} (V') \right\|
\]

\[
= \frac{1}{|U'|} \sum_{u \in U'} u - \frac{1}{|V'|} \sum_{v \in V'} v
\]

\[
= \frac{1}{|U'|} \left( \sum_{u \in U' \cap (U \cap V')} u - \sum_{v \in V' \cap (U \cap V')} v \right)
\]

\[
\leq \frac{1}{n - t} \left( \sum_{i=1}^{k+1} u_i - \sum_{i=1}^{k+1} v_i \right)
\]

\[
\leq \frac{1}{n - t} \left( (k + 1) \text{diam}(P) \right)
\]

\[
< \frac{n}{2t} \cdot t \cdot \text{diam} (P) = \frac{1}{2} \text{diam} (P)
\]

**Lemma B.3 (Eventual Viewpoint Consistency).** Let \(i, j\) be two nonfaulty nodes that reach round \(r\) of the protocol. Then either (1) from some point on, for every \(s \leq r\), \(\text{values}^s_i \subseteq \text{values}^s_j\) and \(\text{reports}^s_i \subseteq \text{reports}^s_j\) or (2) \(i\) eventually completes the protocol and terminates.

**Proof.** If \(i\) completes the protocol, we are done. If that is not the case, we will prove the claim holds by induction on \(s\). For \(s = 0\), if \(\text{ex}_k \in \text{values}^0_j\), then \(j\) received an \("\text{init}_\text{value}\", v_k\) broadcast from some node \(k\) s.t. \(\text{ex}_\text{val}(v_k) = \text{true}\). From the Liveness and Uniqueness properties of the broadcast channel, \(i\) will also receive that message from node \(k\), find that \(\text{ex}_\text{val}(v_k) = \text{true}\) and add \(v_k\) to \(\text{values}^0_i\). From the Uniqueness property of the broadcast channel there can only be \(n\) such broadcasts, so the number of elements in \(\text{value}^0_i\) is bounded, and we can make this argument for very such element. In addition, if \(\text{rec}_\text{vals}_k \in \text{reports}^0_i\), then \(j\) must have received a \("\text{report}\", \text{rec}_\text{vals}_k, 0\) broadcast from \(k\) and at some point in time \(\text{rec}_\text{vals}_k \subseteq \text{values}^0_i\), \(|\text{rec}_\text{vals}_k| \geq n - t\). From the Liveness and Uniqueness properties of the broadcast channel, \(i\) will also receive the \("\text{report}\", \text{rec}_\text{vals}_k, 0\) message from \(k\). Seeing as the sets only grow, following the same logic as above eventually \(\text{rec}_\text{vals}_k \subseteq \text{values}^0_j \subseteq \text{values}^0_i\) and \(|\text{rec}_\text{vals}_k| \geq n - t\), at which point \(i\) will add \(\text{rec}_\text{vals}_k\) to \(\text{values}^0_i\).

Now assume the claim holds for some \(s - 1\). Observe some \(v_k \in \text{values}^s_j\). Since \(j\) is nonfaulty, it must have received a \("\text{value}\", c_k, \text{rec}_\text{vals}^{s-1}_k, \text{rec}_\text{reps}^{s-1}_k, s\) broadcast and added it to \(\text{waiting}_\text{reports}_j\). Then, at some point in time, \(j\) found in line 14 that the message satisfies the conditions of Algorithm 5. From the Liveness and Uniqueness properties of the broadcast protocol, \(i\) will receive that message as well and add it to \(\text{waiting}_\text{values}_i\). By assumption \(i\) reaches round \(r\), and thus also reaches round \(s \leq r\). By the induction hypothesis \(\text{values}^{s-1}_i \subseteq \text{values}^{s-1}_j\) as well as \(\text{reports}^{s-1}_i \subseteq \text{reports}^{s-1}_j\). In other words, eventually \(i\) will see that the conditions of Algorithm 5 hold in line 14 and add \(v_k\) to \(\text{values}^s_i\).
Similarly, if rec vals_k ∈ reports_j, then j received a message ("report", rec vals_k, s) and added it to waiting reports_j.

Then, j found in line 19 that rec vals_k ⊆ values^j, |rec vals_k| ≥ n - t. From the Liveness and Uniqueness properties of the broadcast channel, eventually node i will receive that broadcast from k as well and add it to waiting reports_i. We’ve already shown that eventually i will reach round s and rec vals_k ⊆ values^i ⊆ values^j, |rec vals_k| ≥ n - t. At that point i will add rec vals_k to reports^i, completing our proof.

**Lemma B.4 (Init Termination).** If all nonfaulty nodes participate in the protocol and have valid inputs, then they all complete init_node.

**Proof.** Assume all nonfaulty nodes participate in the protocol and have valid inputs. In the beginning of the protocol, they call init_node and broadcast an ("init_value", v^0_i) message. From the Validity property of broadcast channels, every nonfaulty node i will receive those messages. Since all of them have valid inputs, i will find that ex_val(v^0_i) = true for every nonfaulty j, and add v^0_j to values^i. After that, every nonfaulty i will find that |values^i| ≥ n - t and send a ("report", values^0, 0) message. Note that every nonfaulty node j that completes init_node has at least one value in its reports^j set. A nonfaulty j will only add the set rec vals to reports^0_j if |rec vals| ≥ n - t and rec vals ⊆ values^0_j. This implies that |values^0_j| ≥ n - t, which means that j must have sent some "report" message for round 0. In other words, every nonfaulty j that completes init_node also broadcast some "report" message for round 0.

Now observe some nonfaulty i. If i completes the protocol, then it must have clearly completed init_node as well. Otherwise, it received a ("report", rec vals^0, 0) message from every nonfaulty j such that rec vals^j ⊆ values^0 and |rec vals| ≥ n - t. We know from Lemma 3.5 that eventually for every nonfaulty j, values^0_j ⊆ values^i_j, at which point i will find that the conditions in line 4 of init_node hold and add rec vals^j to reports^i_j. After adding such a tuple for every nonfaulty node, i will find that |reports^i_j| = n - t and broadcast an "enough" message. From the Validity property of broadcast channels, every nonfaulty node i will receive the "enough" message sent by all nonfaulty nodes. At that point, both |termination times| ≥ n - t and |reports^i_j| ≥ n - t, and i will complete init_node.

**Lemma B.5 (Iterations Eventually Terminate).** If all nonfaulty nodes participate in the protocol, reach round r of the protocol for r ≥ 1, and none of them terminate during it, then they all complete the round.

**Proof.** In the beginning of the round, every nonfaulty node broadcasts the message ("value", v^j, values^j, reports^j, r). From the Validity property of broadcast channels, every nonfaulty node will receive the "value" message sent by every nonfaulty node in round r. Observe some nonfaulty node i. If node i never completes round r, it must never terminate because i doesn’t terminate during the round. In that case, we know from Lemma 3.5 that from some point on values^j ⊆ values^i and reports^j ⊆ reports^i for every nonfaulty node j. Therefore, if j broadcasts some message ("value", v^j, rec vals^j, rec reps^j, r), then we know that eventually rec vals^j ⊆ values^j and rec reps^j ⊆ reports^j. Note that j proceeds to round r only after |reports^j| ≥ n - t and at that point it computes v^j = vote(values^j). Furthermore, j would have only added a set rec vals to reports^j if rec vals ⊆ values^j and |rec vals| ≥ n - t. This also implies that |values^j| ≥ n - t. Combining these observations, eventually i will see that the conditions in line 14 hold, and add v^j to values^i. After adding such a value for every nonfaulty node, i will find that |values^i| ≥ n - t, and broadcast a ("report", values^i, r) message. It is important to note that if i completes round r, then |reports^i| ≥ n - t. As shown above, this must mean that |values^i| ≥ n - t and thus i sends a "report" message for round r.
Observe some nonfaulty node \( i \) again. If \( i \) completes the protocol, then it must have completed round \( r \) first. Otherwise, \( i \) never completes the protocol and therefore receives a \("report", \{ r \} \) message from every honest node \( j \) and adds it to \( \text{waiting reports} \). Following similar arguments to the ones above we know that eventually \( \text{val}_{r}^{i} \subseteq \text{app}_{r}^{i} \subseteq \text{app}_{0}^{i} \). Furthermore, \( j \) only sends a \("report" \) message after finding the conditions of line 16 hold, at which point \( |\text{app}_{r}^{j}| \geq n - t \). This means that for every honest \( j \), node \( i \) will eventually see that the conditions in line 19 hold, and add \( \text{val}_{r}^{i} \) to \( \text{reports}_{r}^{i} \). After adding a tuple for each honest node, \( i \) sees that \( |\text{reports}_{r}^{i}| \geq n - t \) and continue to the next line. Afterwards \( i \) will perform a few local computations, and start the next round.

**Lemma B.6.** Let \( V \) be the set of all nonfaulty inputs. When executing MultiDimApproxAgree with \( \	ext{ex\_val}(\cdot) \) always returning true, if a nonfaulty node accepts a value \( v \) in round 1, then there exists some nonfaulty input \( x \in V \) such that \( \text{dist}(v, x) \leq \text{diam}(V) \).

**Proof.** First of all we will more formally state the lemma. Define \( \text{ex\_val}(x) \) to be true for every \( x \in \mathbb{R}^{m} \). Let \( G \) be the set of all nonfaulty nodes and let \( V \) be the set of all nonfaulty inputs, i.e. \( V = \{x| i \in G \} \). Similarly to the proof of the Correctness property of the protocol, for every nonfaulty \( i \) observe \( \text{val}_{r}^{i} \) at a point in time where \( i \) doesn’t add any more values to it. Define \( \text{val}_{0}^{1} = \bigcup_{i \in G} \text{val}_{0}^{i} \). Then, for every \( v \in \text{val}_{0}^{1} \) there exists some \( x \in V \) such that \( \text{dist}(v, x) \leq \text{diam}(V) \).

Observe some \( v \in \text{val}_{0}^{1} \). Some nonfaulty node \( i \) must have added \( v \) to its \( \text{val}_{r}^{i} \) set, so it received a \("value", \{ r \}, \text{app}_{r}^{i}, 1 \) broadcast such that \( \text{app}_{r}^{i} \subseteq \text{val}_{0}^{i} \), \( \text{val}_{r}^{i} \geq n - t \), and \( v \in \text{conv}(\text{Elim}(\text{val}_{r}^{i})) \).

The set \( \text{val}_{0}^{i} \) only contains values which were received in an \"init\_values\" broadcast. A nonfaulty \( i \) broadcasts its input \( x_{i} \) in its \"init\_values\" broadcast. Since \( \text{val}_{0}^{i} \subseteq \text{val}_{0}^{i} \), it contains up to \( t \) values broadcast by faulty nodes and at least \( n - 2t \geq t + 1 \) values \( x_{j} \in V \). As stated above, \( v \in \text{conv}(\text{Elim}(\text{val}_{r}^{i})) \).

Using the convexity of the distance function, to complete the proof it is enough to show that there exists a point \( x \in V \) such that for every \( v_{i} \in \text{Elim}(\text{val}_{r}^{i}) \), \( \text{dist}(x, v_{i}) \leq \text{diam}(V) \). From this point on, the proof is extremely similar to the proof of theorem 2.7.

Recall that \( \text{Elim}(\text{val}_{r}^{i}) \) consists of \( t \) iterations of deleting the pair of furthest-distance points. For every \( i \in [t] \), denote \( (p_{i}, q_{i}) \) to be the pair deleted in the \( i \)’th iteration of \( \text{Elim}(\text{val}_{r}^{i}) \). If \( \text{Elim}(\text{val}_{r}^{i}) \subseteq V \), then for any point \( x \in V \) and any point \( v_{i} \in \text{Elim}(\text{val}_{r}^{i}) \), \( \text{dist}(x, v_{i}) \leq \text{diam}(V) \) by the definition of the diameter. Otherwise, there exists some \( v_{i} \in \text{Elim}(\text{val}_{r}^{i}) \) such that \( v_{i} \notin V \). Assume by way of contradiction that there is no \( i \) such that both \( p_{i}, q_{i} \in V \). In that case, at least one distinct value from \( \text{val}_{r}^{i} \) is deleted in each iteration. There are \( t \) such iterations and at most \( t \) such values, so this means that all points not in \( V \) have been deleted throughout \( \text{Elim}(\text{val}_{r}^{i}) \). In other words, \( \text{Elim}(\text{val}_{r}^{i}) \subseteq V \), reaching a contradiction. Therefore there exists some \( i \in [t] \) such that \( p_{i}, q_{i} \in V \) are the furthest-distance pair in the \( i \)’th iteration. Setting \( x = p_{i} \), we know that for any remaining point \( q_{i} \) at that moment \( \text{dist}(v_{i}, x) \leq \text{dist}(p_{i}, q_{i}) \leq \text{diam}(V) \). All points in \( \text{Elim}(\text{val}_{r}^{i}) \) must be points that have remained after \( i \) iterations, completing the proof. □