Brief Announcement: Asynchronous Approximate Agreement with Quadratic Communication

Mose Mizrahi Erbes
ETH Zürich, Switzerland

Roger Wattenhofer
ETH Zürich, Switzerland

Abstract

We consider an asynchronous network of n message-sending parties, up to t of which are byzantine. We study approximate agreement, where the parties obtain approximately equal outputs in the convex hull of their inputs. In their seminal work, Abraham, Amit and Dolev [OPODIS '04] solve this problem in $\mathbb{R}$ with the optimal resilience $t < \frac{n}{3}$ with a protocol where each party reliably broadcasts a value in every iteration. This takes $\Theta(n^2)$ messages per reliable broadcast, or $\Theta(n^3)$ messages per iteration.

In this work, we forgo reliable broadcast to achieve asynchronous approximate agreement against $t < \frac{n}{3}$ faults with quadratic communication. In a tree with the maximum degree Δ and the centroid decomposition height h, we achieve edge agreement in at most $6h + 1$ rounds with $\mathcal{O}(n^2)$ messages of size $\mathcal{O}(\log \Delta + \log h)$ per round. We do this by designing a 6-round multivalued 2-graded consensus protocol and using it to recursively reduce the task to edge agreement in a subtree with a smaller centroid decomposition height. Then, we achieve edge agreement in the infinite path $\mathbb{Z}$, again with the help of 2-graded consensus. Finally, we show that our edge agreement protocol enables ε-agreement in $\mathbb{R}$ in $6 \log_2 \frac{M}{\varepsilon} + \mathcal{O}(\log \log \frac{M}{\varepsilon})$ rounds with $\mathcal{O}(n^2 \log \frac{M}{\varepsilon})$ messages and $\mathcal{O}(n^2 \log \frac{M}{\varepsilon} \log \log \frac{M}{\varepsilon})$ bits of communication, where M is the maximum non-byzantine input magnitude.

2012 ACM Subject Classification Theory of computation → Distributed algorithms

Keywords and phrases Approximate agreement, byzantine fault tolerance, communication complexity

Digital Object Identifier 10.4230/LIPIcs.DISC.2025.61

Related Version *Full Version*: `https://arxiv.org/abs/2408.05495` [10]

1 Introduction

We consider a fully connected asynchronous network of n message-passing parties $P_1, \ldots, P_n$. Up to t of these parties are corrupted in a byzantine manner, while the rest are honest.

In an approximate (convex) agreement problem, the parties output approximately equal values in the convex hull of their inputs. The most classical example is approximate agreement in $\mathbb{R}$, where the inputs/outputs are in $\mathbb{R}$, and for some parameter $\varepsilon > 0$ the following hold:

- ***validity:*** Each honest party output is between the minimum and maximum honest inputs.
- ***ε-agreement:*** If any honest parties P_i and P_j output y_i and y_j, then $|y_i - y_j| \leq \varepsilon$.

Approximate agreement in $\mathbb{R}$ was introduced in 1985 by Dolev, Lynch, Pinter, Stark and Weihl [8]. Like byzantine agreement, in synchronous networks it is possible against $t < \frac{n}{2}$ faults with setup [12], but only possible when $t < \frac{n}{3}$ if the network is asynchronous [1] or if perfect (signature-free) security is desired [8]. What sets approximate agreement apart is that it is determinism-friendly. While deterministic byzantine agreement takes $t+1$ rounds in synchrony [7] and is impossible against just one crash in asynchrony [11], approximate agreement does not share these limitations. Thus, approximate agreement protocols are customarily deterministic.

In [8], Dolev et al. achieve ε-agreement in $\mathbb{R}$ with a perfectly secure synchronous protocol secure against $t < \frac{n}{3}$ corruptions. Simplifying things slightly, in their protocol the parties estimate the spread S of their inputs (the maximum difference between any two inputs), and

run for $\lceil \log_2 \frac{S}{\varepsilon} \rceil$ rounds. In each round each party sends its value to every other party, and with this the parties halve the diameter of their values. After $\lceil \log_2 \frac{S}{\varepsilon} \rceil$ rounds, the spread is at most $2^{-\lceil \log_2(S/\varepsilon) \rceil} \leq \frac{\varepsilon}{S}$ of what it initially was, and thus ε-agreement is achieved. They present an asynchronous version of this protocol as well, but only with the resilience $t < \frac{n}{5}$ as the parties can no longer wait to receive the value of each honest party in every iteration.

Asynchronous approximate agreement in $\mathbb{R}$ with the optimal resilience $t < \frac{n}{3}$ was first achieved in 2004 by Abraham, Amit and Dolev [1], with a protocol that consists of $\mathcal{O}(\log \frac{S}{\varepsilon})$ constant-round iterations. Each iteration involves one witness technique application where each party reliably broadcasts its current value in $\mathbb{R}$ (with a reliable broadcast protocol such as Bracha's [5]), and obtains at least $n - t$ reliably broadcast values. The technique ensures that every two parties obtain the values of at least $n-t$ common parties. Since the technique's introduction in [1], most asynchronous approximate agreement protocols have depended on it. Some examples are [1, 12] for agreement in $\mathbb{R}$, [14] for agreement in $\mathbb{R}^d$ when $d \geq 2$ and [15] for agreement in graphs (trees, chordal graphs, cycle-free semilattices). Note that this list is not exhaustive; we mention some other examples in the full version [10].

The witness technique requires the parties to reliably broadcast their inputs. Since reliable broadcast requires $\Omega(n^2)$ messages for deterministic [9] or strongly adaptive [2] security against $t = \Omega(n)$ faults, the witness technique requires $\Omega(n^3)$ messages to be sent. Hence, the optimally resilient approximate agreement protocol of Abraham, Amit and Dolev [1] costs $\Theta(n^3)$ messages per iteration. This is the case despite asynchronous approximate agreement being possible with $\Theta(n^2)$ messages per iteration, as demonstrated by the protocol of Dolev et al. [8] which suboptimally tolerates $t < \frac{n}{5}$ faults. Therefore, we ask the following question: *Is there an asynchronous approximate agreement protocol that optimally tolerates $t < \frac{n}{3}$ faults with only a quadratic (proportional to n^2) amount of communication?*

In this work, we answer this question affirmatively by abandoning the witness technique. First, we achieve edge agreement (a discrete form of approximate agreement) in finite trees [15] with the optimal resilience $t < \frac{n}{3}$ via multivalued 2-graded consensus iterations. Then, we extend our protocol to achieve edge agreement in the infinite path $\mathbb{Z}$. Finally, we show that edge agreement in $\mathbb{Z}$ implies ε-agreement in $\mathbb{R}$ by reducing the latter to the former. Our final protocol for ε-agreement in $\mathbb{R}$ takes $6 \log_2 \frac{M}{\varepsilon} + \mathcal{O}(\log \log \frac{M}{\varepsilon})$ rounds (where M is the maximum honest input magnitude), with $\mathcal{O}(n^2)$ messages of size $\mathcal{O}(\log \log \frac{M}{\varepsilon})$ sent per round.

Our work is inspired by [13], which achieves exact convex agreement in $\mathbb{Z}$ with byzantine agreement iterations in a synchronous network. We instead achieve edge agreement in $\mathbb{Z}$ with graded consensus, which is much simpler than byzantine agreement, especially in asynchronous networks. Note though that [13] supports large inputs with less communication than us.

2 Model & Definitions

We consider an asynchronous network of n message-sending parties $P_1, P_2, \ldots, P_n$, fully connected via reliable and authenticated channels. An adversary corrupts up to $t < \frac{n}{3}$ parties, making them byzantine, and these parties become controlled by the adversary. The adversary adaptively chooses the parties it wants to corrupt during protocol execution, depending on the messages sent over the network. If a party is never corrupted, then we call it honest.

The parties do not have synchronized clocks. The adversary can schedule messages as it sees fit, and it is only required to eventually deliver messages with honest senders. If a party sends a message, then the adversary may corrupt the party instead of delivering the message.

To define asynchronous round complexity, we imagine an external clock. If a protocol runs in R rounds, then the time elapsed between when every honest party running the protocol knows its input and when every honest party outputs/terminates is at most $R\Delta$, where Δ is the maximum honest message delay in the protocol's execution.

Edge Agreement in a Tree

In edge agreement in a tree graph $T = (V, E)$, each party P_i acquires an input vertex $v_i \in V$, and outputs a vertex $y_i \in V$. We want the following properties:

- ***edge agreement:*** Every two honest output vertices are either equal or adjacent in T.
- ***convex validity:*** For every honest output y, there exist some (possibly equal) honest inputs v_y and v'_y such that y is on the path which connects v_y and v'_y in T.

Edge agreement in a tree generalizes edge agreement in a path, which is essentially the same task as approximate agreement in an interval in $\mathbb{R}$, but with the input/output domain restricted to the integers (with adjacent integers representing adjacent path vertices).

Graded Consensus

In k-graded consensus, each party P_i acquires an input v_i in an input domain $\mathcal{M}$, and outputs some value-grade pair $(y_i, g_i) \in (\mathcal{M} \times \{1, \ldots, k\}) \cup \{(\perp, 0)\}$. The following must hold:

- ***agreement:*** If any honest parties P_i and P_j output (y_i, g_i) and (y_j, g_j), then $|g_i - g_j| \leq 1$, and if $\min(g_i, g_j) \geq 1$, then $y_i = y_j$.
- ***intrusion tolerance:*** If $(y, g) \neq (\perp, 0)$ is an honest output, then y is an honest input.
- ***validity:*** If the honest parties have a common input $m \in \mathcal{M}$, then they all output (m, k).

As [3] has noted before, k-graded consensus is equivalent to edge agreement in a particular tree when the inputs must all be leaves of the tree.

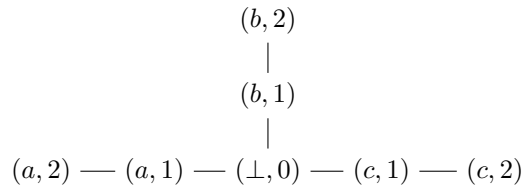

Figure 1 Edge agreement in a spider tree with the center $(\perp, 0)$ and a path $((m, 1), \ldots, (m, k))$ attached to it for each $m \in \mathcal{M}$ is equivalent to k-graded consensus when the parties can only have leaf edge agreement inputs, with each leaf input (m, k) in bijection with the k-graded consensus input m.

In the full version [10], we construct a family of multivalued 2^k-graded consensus protocols $\mathsf{GC}_{2^0}, \mathsf{GC}_{2^1}, \mathsf{GC}_{2^2}, \ldots$ which each take $3k + 3$ rounds, with $\mathcal{O}(n^2)$ messages of size $\mathcal{O}(\log k + \log |\mathcal{M}|)$ per round. These are the most efficient multivalued graded consensus protocols we are aware of. In the full version we make use of the 6-round 2-graded consensus protocol GC_2, while in this brief announcement we present a simplified approximate agreement protocol that uses 3-graded consensus, which the parties can reach by running the 9-round 4-graded consensus protocol GC_4 and subtracting 1 from their output grades if they obtain the grade 4.

3 Overview & Contributions

Our first contribution is a protocol for edge agreement in finite trees. Against byzantine faults, this problem was first studied by Nowak and Rybicki [15]. It generalizes both edge agreement in finite paths (the discrete version of ε-agreement in $[0, 1]$) and graded consensus.

Nowak and Rybicki achieve edge agreement in a finite tree $T = (V, E)$ of diameter D with $\lceil \log_2 D \rceil + 1$ constant-round witness technique iterations and thus $\Theta(n^3 \log D(\log |V| + \log n))$ bits of communication, where the $\log n$ term is due to the party IDs that identify each reliable

broadcast's sender party. Meanwhile, we achieve edge agreement with at most $h(T)$ iterations ($6h(T)+1$ rounds), where $h(T)$ (T's centroid decomposition [16] height) is a property we define in the full version [10]. The integer value $h(T)$ can be anywhere in $[\lfloor \log_2 D \rfloor, \lceil \log_2 |V| \rceil]$, which means that our protocol's round complexity is for some trees (though not for spider trees, trees with $\mathcal{O}(D)$ vertices etc.) worse than Nowak and Rybicki's. However, our iterations only cost $\mathcal{O}(n^2)$ messages, each of size at most $\mathcal{O}(\log \Delta + \log(h(T)))$ where Δ is T's maximum degree. So, our protocol requires roughly n times less communication when $h(T) \approx \log_2 D$.

In the full version [10], we present a parametrized recursive protocol $\mathsf{TC}(T)$ that achieves edge agreement in any given finite tree T. On a high level, it works as follows:

1. If T has 1 or 2 vertices, then each party outputs its input vertex. This is the base case.
2. If T has $s \geq 3$ vertices, then the parties let σ be a centroid vertex of T (whose deletion from T results in a forest whose components all have at most $s/2$ vertices), and let $w_1, \ldots, w_d$ be σ's neighbors sorted by vertex index. Then, they run 2-graded consensus, where each party's input is either σ (if its edge agreement input is σ) or some neighbor w_k of σ (if its edge agreement input is in H_k, which is how we refer to the tree component of $T \setminus \{\sigma\}$ that contains w_k). If the parties reach consensus on σ, then they output σ. Otherwise, if they reach consensus on some neighbor w_k of σ, then the parties with input vertices outside H_k adopt the new input w_k, and we reduce the task to edge agreement in the subtree H_k.

There is a snag. The explanation above only works if the parties actually reach unanimous agreement on either σ or on one of its neighbors w_k. However, 2-graded consensus does not guarantee this, since some parties might output $(\perp, 0)$ from it. What allows us to overcome this issue is that if anybody outputs $(\perp, 0)$, then the parties all learn that they ran 2-graded consensus with differing inputs, and thus learn that σ is a safe output vertex w.r.t convex validity.

Our approach for finite trees above corresponds to binary search when the tree is a path. For example, the parties reach edge agreement in the path $(0, \ldots, 8)$ by either directly agreeing on 4, or by reducing the problem to edge agreement in either $(0, \ldots 3)$ or $(5, \ldots, 8)$. Binary search does not support the infinite path $\mathbb{Z}$. Fortunately, 2-graded consensus also enables exponential search. In the full version [10], we present a protocol for edge agreement in $\mathbb{Z}$ where we use 2-graded consensus to implement a strategy based on (doubly) exponential search.

When the maximum honest input magnitude is M, our protocol for edge agreement in $\mathbb{Z}$ takes $6 \log_2 M + \mathcal{O}(\log \log M)$ rounds, with $\mathcal{O}(n^2)$ messages of size $\mathcal{O}(\log \log M)$ per round. In the full version [10], we reduce ε-agreement in $\mathbb{R}$ to edge agreement in $\mathbb{Z}$ to show that this implies ε-agreement in $\mathbb{R}$ in $6 \log_2 \frac{M}{\varepsilon} + \mathcal{O}(\log \log \frac{M}{\varepsilon})$ rounds with $\mathcal{O}(n^2 \log \frac{M}{\varepsilon})$ messages and $\mathcal{O}(n^2 \log \frac{M}{\varepsilon} \log \log \frac{M}{\varepsilon})$ bits of communication in total. Note that the factor 6 in the round complexity here stems from us using our 6-round 2-graded consensus protocol.

In terms of message and communication (though not round) complexity, our protocol for ε-agreement in $\mathbb{R}$ is more efficient than that of Abraham et al. [1], who achieve ε-agreement in $\mathbb{R}$ with $\mathcal{O}(\log \frac{S}{\varepsilon})$ constant-round witness technique iterations (where S is the honest input spread, i.e. the maximum difference of any honest inputs), and with $\Theta(n^3 \log \frac{S}{\varepsilon})$ messages in total.

Another notable protocol is Delphi, by Bandarupalli, Bhat, Bagchi, Kate, Liu-Zhang and Reiter [4]. To efficiently achieve ε-agreement with ℓ-bit inputs in $\mathbb{R}$, they assume an input distribution (normal distribution for the following), and when the honest input spread is S they achieve ε-agreement except with probability $2^{-\lambda}$ in $\mathcal{O}(\log(\frac{S}{\varepsilon} \log \frac{S}{\varepsilon}) + \log(\lambda \log n))$ rounds with $\mathcal{O}(\ell n^2 \frac{S}{\varepsilon}(\log(\frac{S}{\varepsilon} \log \frac{D}{\varepsilon}) + \log(\lambda \log n)))$ bits of communication, while relaxing validity by allowing outputs outside the range of the honest inputs by at most S. They use the security parameter λ here to assume bounds on S that hold except with $2^{-\lambda}$ probability thanks to their input distribution assumptions. In comparison, we achieve ε-agreement in $\mathbb{R}$ without relaxing validity or assuming any input bounds. As Table 1 shows, our protocol is also more efficient, in particular since Delphi requires a cubic amount of communication per round when $S \geq n \cdot \varepsilon$.

Table 1 Comparison of protocols for asynchronous ε-agreement in $\mathbb{R}$ when the parties have inputs in $[0,1]$. If v_{lo} and v_{hi} are the minimum and maximum honest inputs, then $S = v_{\mathsf{hi}} - v_{\mathsf{lo}}$ and $M = v_{\mathsf{hi}}$. To make the comparisons simple, we assume for [8], [1] and [4] that the inputs are all multiples of ε.

Threshold	Bits Sent / Round	Round Complexity	Relaxation[a]	Source
$t < \frac{n}{5}$	$\mathcal{O}(n^2 \log \frac{1}{\varepsilon})$	$\lceil \log_2 \frac{1}{\varepsilon} \rceil$	0	[8]
$t < \frac{n}{3}$	$\mathcal{O}(n^3 \log \frac{n}{\varepsilon})$[b]	$\mathcal{O}(\log \frac{S}{\varepsilon})$	0	[1]
$t < \frac{n}{3}$	$\mathcal{O}(n^2 \min(\frac{S}{\varepsilon}, n \log \frac{1}{\varepsilon}))$	$\mathcal{O}(\log(\frac{\log(1/\varepsilon)\min(1/\varepsilon,n)}{\varepsilon}))$	S	[4]
$t < \frac{n}{3}$	$\mathcal{O}(n^2 \log\log \frac{M}{\varepsilon})$[c]	$\mathcal{O}(\log \frac{M}{\varepsilon})$	0	this work

a) The relaxation is how far an honest output is allowed to be from the honest input range $[v_{\mathsf{lo}}, v_{\mathsf{hi}}]$.
b) The first few rounds of [1] estimate the spread S, and this costs $\Theta(n^4 \log \frac{1}{\varepsilon})$ bits of communication. However, this can be reduced to $\Theta(n^3 \log \frac{n}{\varepsilon})$ with modern reliable broadcast protocols [6].
c) The $\log\log \frac{M}{\varepsilon}$ factor here is for tags that distinguish messages sent in different protocol iterations.

4 Simplified Approximate Agreement in $[0,1]$

In this section, we present a simplified ε-agreement protocol that is better suited for a brief announcement than the more complicated full protocol in the full version [10]. While the full protocol uses 2-graded consensus, the simplified protocol in this section uses 3-graded consensus, which makes it less round-efficient since (with our graded consensus constructions) 3-graded consensus takes 9 rounds while 2-graded consensus takes 6. Moreover, the simplified protocol only supports inputs in bounded intervals rather than inputs in $\mathbb{R}$. Still, the protocols share the same core idea, which is that the parties repeatedly bisect the interval where their values reside by reaching graded consensus on whether their values are low or high.

In $\mathsf{Apx}(a,b)$, the parameterized approximate agreement protocol we present in this section, the parties reach ε-agreement in an interval $[a,b]$ by using 3-graded consensus to recursively reduce ε-agreement in $[a,b]$ to ε-agreement in either $[a, \frac{a+b}{2}]$ (reached via $\mathsf{Apx}(a, \frac{a+b}{2})$) or $[\frac{a+b}{2}, b]$ (reached via $\mathsf{Apx}(\frac{a+b}{2}, v)$). The parties reach 3-graded consensus on whether their inputs are below the midpoint $\frac{a+b}{2}$ or not. If they all have inputs below (resp. above) $\frac{a+b}{2}$, then they unanimously agree that this is the case, and so they run $\mathsf{Apx}(a, \frac{a+b}{2})$ (resp. $\mathsf{Apx}(\frac{a+b}{2}, b)$) to reach approximate agreement. Otherwise, $\frac{a+b}{2}$ is in the honest input range, and this fact lets us assign an appropriate behavior to each 3-graded consensus output so that the parties reach approximate agreement no matter which two 3-graded consensus outputs they settle on.

When the parties run $\mathsf{Apx}(0,1)$ for ε-agreement in $[0,1]$, they reach the base case of a recursive $\mathsf{Apx}(a,b)$ instance where $b-a \leq \varepsilon$ (where they can just output their inputs) with $\lceil \log_2 \frac{1}{\varepsilon} \rceil$ recursive Apx calls, or in other words $\lceil \log_2 \frac{1}{\varepsilon} \rceil$ iterations of 3-graded consensus. In total, this costs $9\lceil \log_2 \frac{1}{\varepsilon} \rceil$ rounds, $\mathcal{O}(n^2 \log \frac{1}{\varepsilon})$ messages and $\mathcal{O}(n^2 \log \frac{1}{\varepsilon} \log\log \frac{1}{\varepsilon})$ bits of communication. Here, we have a $\log\log \frac{1}{\varepsilon}$ factor because the parties have to be able to tell to which 3-graded consensus iteration each message belongs to, and this requires $\mathcal{O}(\log\log \frac{1}{\varepsilon})$-bit message tags.

Note that Apx does not allow the parties to terminate (stop sending messages) after they output, since some parties not sending the messages they are supposed to send could lead to some other parties never obtaining outputs. We address this shortcoming in the full version [10] with a simple constant-round quadratic-complexity termination procedure.

Protocol $\mathsf{Apx}(a, b)$

Code for a party P_i with the input v_i

if $b - a \leq \varepsilon$ **then**
  output v_i and do not run the rest of the protocol
run an instance of a 3-graded consensus protocol GC_3 with the other parties where your input is `LEFT` if $v_i < \frac{a+b}{2}$, and `RIGHT` if $v_i \geq \frac{a+b}{2}$
wait until you output some (k, g) from GC_3
if $g = 3$ **then**
  let $v_i^{\mathsf{next}} \leftarrow \min(v_i, \frac{a+b}{2})$ if $k = \texttt{LEFT}$, and let $v_i^{\mathsf{next}} \leftarrow \max(v_i, \frac{a+b}{2})$ if $k = \texttt{RIGHT}$
else
  $v_i^{\mathsf{next}} \leftarrow \frac{a+b}{2}$
if $g \leq 1$ **then**
  output $\frac{a+b}{2}$
if $k = \texttt{LEFT}$ **then**
  run an instance $\mathsf{Apx}(a, \frac{a+b}{2})$ with the other parties where your input is v_i^{next}
  when you output y from $\mathsf{Apx}(a, \frac{a+b}{2})$, output y from $\mathsf{Apx}(a, b)$ if $g \geq 2$
else if $k = \texttt{RIGHT}$ **then**
  run an instance $\mathsf{Apx}(\frac{a+b}{2}, b)$ with the other parties where your input is v_i^{next}
  when you output y from $\mathsf{Apx}(\frac{a+b}{2}, b)$, output y from $\mathsf{Apx}(a, b)$ if $g \geq 2$

If the parties all run $\mathsf{Apx}(a, b)$ with inputs in $[a, \frac{a+b}{2})$, then each party P_i runs GC_3 with the input `LEFT`, outputs $(\texttt{LEFT}, 3)$ from GC_3, and obtains its final output from an $\mathsf{Apx}(a, \frac{a+b}{2})$ instance (that everybody runs) where its input is $v_i^{\mathsf{next}} = v_i$. So, $\mathsf{Apx}(a, b)$'s security recursively follows from $\mathsf{Apx}(a, \frac{a+b}{2})$'s security. Likewise, if the parties all run $\mathsf{Apx}(a, b)$ with inputs in $[\frac{a+b}{2}, b]$, then they recursively reach ε-agreement via $\mathsf{Apx}(\frac{a+b}{2}, b)$.

On the other hand, if neither $[a, \frac{a+b}{2})$ nor $[\frac{a+b}{2}, b]$ contain all inputs, then $\frac{a+b}{2}$ is a safe output value w.r.t. validity. Knowing this, we can prove $\mathsf{Apx}(a, b)$'s security via case analysis.

- If the parties all output $(\texttt{LEFT}, 3)$ or $(\texttt{LEFT}, 2)$ from GC_3, then they all run $\mathsf{Apx}(a, \frac{a+b}{2})$ with inputs in $[a, \frac{a+b}{2}]$ and obtain their outputs from it. So, security follows from $\mathsf{Apx}(a, \frac{a+b}{2})$. Some parties (those with $\mathsf{Apx}(a, b)$ inputs above $\frac{a+b}{2}$ and those with the GC_3 grade 2) run $\mathsf{Apx}(a, \frac{a+b}{2})$ with the input $v_i^{\mathsf{next}} = \frac{a+b}{2}$ instead of v_i. This is fine because $\frac{a+b}{2}$ is safe.
- If the parties all output $(\texttt{LEFT}, 2)$ or $(\texttt{LEFT}, 1)$ from GC_3, then they all run $\mathsf{Apx}(a, \frac{a+b}{2})$ with the input $\frac{a+b}{2}$, and thus all output $\frac{a+b}{2}$ from it. The parties with the GC_3 grade 2 output $\frac{a+b}{2}$ from $\mathsf{Apx}(a, b)$ once they output this from $\mathsf{Apx}(a, \frac{a+b}{2})$, while the ones with the GC_3 grade 1 output $\frac{a+b}{2}$ from $\mathsf{Apx}(a, b)$ directly after they obtain the GC_3 grade 1.
- If the parties all output $(\texttt{LEFT}, 1)$ or $(\perp, 0)$ from GC_3, then they all directly output $\frac{a+b}{2}$ from $\mathsf{Apx}(a, b)$ after they output from GC_3. Here, it does not matter that the parties with the GC_3 grade 1 run $\mathsf{Apx}(a, \frac{a+b}{2})$ while the rest do not, because the parties with the GC_3 grade 1 do not care about their $\mathsf{Apx}(a, \frac{a+b}{2})$ outputs.
- The cases where some parties obtain the GC_3 value `RIGHT` are similar to the cases above.

Future Work. It remains open to design a protocol for ε-agreement in $\mathbb{R}$ that tolerates $t < \frac{n}{3}$ faults in $\mathcal{O}(\log \frac{S}{\varepsilon})$ rounds (where S is the honest input spread) with quadratic communication. We do not know of any such protocol for even synchronous networks, let alone asynchronous ones. The classical synchronous protocol in [8] which at first seems to fit the bill in fact takes as many rounds as the adversary desires because its round complexity scales with the spread of *all* inputs, including fake byzantine ones. It would be a good first step to solve this issue.

References

1 Ittai Abraham, Yonatan Amit, and Danny Dolev. Optimal resilience asynchronous approximate agreement. In *Proceedings of the 8th International Conference on Principles of Distributed Systems*, OPODIS '04, pages 229–239, Berlin, Heidelberg, 2004. Springer-Verlag. doi:10.1007/11516798_17.

2 Ittai Abraham, T-H. Hubert Chan, Danny Dolev, Kartik Nayak, Rafael Pass, Ling Ren, and Elaine Shi. Communication complexity of byzantine agreement, revisited. In *Proceedings of the 2019 ACM Symposium on Principles of Distributed Computing*, PODC '19, pages 317–326, New York, NY, USA, 2019. Association for Computing Machinery. doi:10.1145/3293611.3331629.

3 Hagit Attiya and Jennifer L. Welch. Multi-Valued Connected Consensus: A New Perspective on Crusader Agreement and Adopt-Commit. In *27th International Conference on Principles of Distributed Systems (OPODIS 2023)*, volume 286 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 6:1–6:23, Dagstuhl, Germany, 2024. Schloss Dagstuhl – Leibniz-Zentrum für Informatik. doi:10.4230/LIPIcs.OPODIS.2023.6.

4 Akhil Bandarupalli, Adithya Bhat, Saurabh Bagchi, Aniket Kate, Chen-Da Liu-Zhang, and Michael K. Reiter. Delphi: Efficient Asynchronous Approximate Agreement for Distributed Oracles. In *2024 54th Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN)*, pages 456–469, Los Alamitos, CA, USA, June 2024. IEEE Computer Society. doi:10.1109/DSN58291.2024.00051.

5 Gabriel Bracha. Asynchronous byzantine agreement protocols. *Information and Computation*, 75(2), 1987. doi:10.1016/0890-5401(87)90054-X.

6 Jinyuan Chen. Ociorcool: Faster byzantine agreement and reliable broadcast, 2024. doi:10.48550/arXiv.2409.06008.

7 D. Dolev and H. R. Strong. Authenticated algorithms for byzantine agreement. *SIAM Journal on Computing*, 12(4):656–666, 1983. doi:10.1137/0212045.

8 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. doi:10.1145/5925.5931.

9 Danny Dolev and Rüdiger Reischuk. Bounds on information exchange for byzantine agreement. *J. ACM*, 32(1):191–204, January 1985. doi:10.1145/2455.214112.

10 Mose Mizrahi Erbes and Roger Wattenhofer. Asynchronous approximate agreement with quadratic communication, 2025. doi:10.48550/arXiv.2408.05495.

11 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. doi:10.1145/3149.214121.

12 Diana Ghinea, Chen-Da Liu-Zhang, and Roger Wattenhofer. Optimal synchronous approximate agreement with asynchronous fallback. In *Proceedings of the 2022 ACM Symposium on Principles of Distributed Computing*, PODC '22, pages 70–80, New York, NY, USA, 2022. Association for Computing Machinery. doi:10.1145/3519270.3538442.

13 Diana Ghinea, Chen-Da Liu-Zhang, and Roger Wattenhofer. Communication-optimal convex agreement. In *Proceedings of the ACM Symposium on Principles of Distributed Computing*, PODC '25, pages 39–49, New York, NY, USA, 2025. Association for Computing Machinery. doi:10.1145/3732772.3733551.

14 Hammurabi Mendes and Maurice Herlihy. Multidimensional approximate agreement in byzantine asynchronous systems. In *Proceedings of the Forty-Fifth Annual ACM Symposium on Theory of Computing*, STOC '13, pages 391–400, New York, NY, USA, 2013. Association for Computing Machinery. doi:10.1145/2488608.2488657.

15 Thomas Nowak and Joel Rybicki. Byzantine Approximate Agreement on Graphs. In *33rd International Symposium on Distributed Computing (DISC 2019)*, volume 146 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 29:1–29:17, Dagstuhl, Germany, 2019. Schloss Dagstuhl – Leibniz-Zentrum für Informatik. doi:10.4230/LIPIcs.DISC.2019.29.

16 A simple introduction to centroid decomposition. A Simple Blog, 2020. Accessed: 2025-08-15. URL: https://robert1003.github.io/2020/01/16/centroid-decomposition.html.