Asynchronous Approximate Agreement with Quadratic Communication

Mose Mizrahi Erbes
ETH Zurich, Switzerland

Roger Wattenhofer
ETH Zurich, Switzerland

Abstract

We study approximate agreement in an asynchronous network of n parties, up to t of which are byzantine. This an agreement task where the parties obtain approximately equal inputs in the convex hull of their inputs. In an asynchronous network, it can be solved with the optimal resilience $t < \frac{n}{3}$ by forcing the parties to reliably broadcast their messages and thus preventing inconsistent byzantine behavior. This costs $\Theta(n^2)$ messages per reliable broadcast, or $\Theta(n^3)$ messages per protocol 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 (agreement on two adjacent vertices) 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 construct a recursive edge agreement protocol. Then, we achieve edge agreement in the infinite path $\mathbb{Z}$, again by using 2-graded consensus. Finally, we show that our edge agreement protocol enables approximate agreement in $\mathbb{R}$ (with outputs that are at most some small parameter $\varepsilon > 0$ apart) in $6 \log_2 \frac{M}{\varepsilon} + \mathcal{O}(\log \log \frac{M}{\varepsilon})$ rounds with $\mathcal{O}(n^2)$ messages of size $\mathcal{O}(\log \log \frac{M}{\varepsilon})$ per round, 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.OPODIS.2025.16

Related Version *Brief Announcement*: https://doi.org/10.4230/LIPIcs.DISC.2025.61 [27]

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 suffer from byzantine faults, 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 [14]. Like byzantine agreement, in synchronous networks it is possible against $t < \frac{n}{2}$ faults with setup (public key infrastructure to enable message signing) [21], but only possible when $t < \frac{n}{3}$ if perfect (signature-free) security is desired [14] or the network is asynchronous. What sets approximate agreement apart is that it is determinism-friendly. While deterministic byzantine agreement takes $t + 1$ rounds in synchrony [13] and is impossible against just one crash in asynchrony [17], approximate agreement does not share these limitations. Thus, approximate agreement protocols are customarily and preferably deterministic.

In [14], Dolev et al. achieved 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 this

is enough for the parties to halve the spread 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 presented 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 by Coan in 1988 [10], who designed a simple protocol like the one above that tolerated $t < \frac{n}{3}$ crash faults, and then upgraded this simple protocol into a byzantine fault tolerant one. He obtained this upgrade by using Bracha's reliable broadcast [8] to prevent the byzantine parties from transmitting different values to different parties, and by introducing a scheme to ensure that the parties ignore any values that could not have been sent by protocol-following parties. This upgrade has the drawback of turning the base protocol's raw broadcasts into reliable broadcasts, causing a $\Theta(n)$-fold communication increase as reliable broadcast requires $\Omega(n^2)$ messages for deterministic [15] or strongly adaptive [2] security against $t = \Omega(n)$ faults. This means that the upgraded protocol costs $\Theta(n^3)$ messages per iteration.

Later in 2004, Abraham, Amit and Dolev achieved a similar result: A cubic-complexity asynchronous protocol for approximate agreement in $\mathbb{R}$ with the optimal resilience $t < \frac{n}{3}$. This protocol has an advantage over the previous ones: Its input spread estimation is fault tolerant, which means that its round complexity scales with the logarithm of the spread of only the honest parties' inputs, rather than the spread of *all* inputs including the byzantine ones. Abraham et al. obtained this protocol by developing the witness technique, which involves each party reliably broadcasting a value and obtaining at least $n - t$ reliably broadcast values, with the guarantee that every two parties obtain the values of at least $n - t$ common parties. Since then, most asynchronous approximate agreement protocols have depended on this technique. Some examples are [1, 21] for agreement in $\mathbb{R}$, [16, 22, 26] for agreement in $\mathbb{R}^d$ when $d \geq 2$, and [11, 29] for agreement in graphs (trees, chordal graphs, cycle-free semilattices).

The optimally resilient protocols above cost $\Theta(n^3)$ messages per round due to their use of reliable broadcast. However, asynchronous approximate agreement is possible with $\Theta(n^2)$ messages per round, as shown by the protocol in [14] which tolerates $t < \frac{n}{5}$ faults. So, we ask: *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 forgoing reliable broadcast. First, we achieve edge agreement in finite trees [29] (agreement on two adjacent vertices) 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 achieve ε-agreement in $\mathbb{R}$ by reducing it to edge agreement in $\mathbb{Z}$. 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, which means that its total communication complexity is quadratic in n. Note that we do not require the parties to know M in advance for this.

Our work is inspired by [23], which achieves exact convex agreement in $\mathbb{Z}$ (on a single integer) with byzantine agreement iterations in a synchronous network. We instead achieve edge agreement in $\mathbb{Z}$ (on two adjacent integers) with iterations of graded consensus, which is a much simpler primitive than byzantine agreement, especially in asynchronous networks.

2 Model & Definitions

We consider an asynchronous network of n message-sending parties $P_1, P_2, \ldots, P_n$ which are fully connected via reliable and authenticated channels. An adversary corrupts up to $t < \frac{n}{3}$ parties, making them byzantine, and it takes control of these parties. The adversary adaptively chooses the parties it wants to corrupt during protocol execution, depending on the parties' internal states and all sent messages. 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.

We say that a party multicasts m when it sends m to every party. By corrupting a party that is multicasting a message, the adversary may deliver the message to only some parties.

Our protocols are live; i.e., they achieve liveness. That is, if the honest parties all acquire inputs and all run forever, then they all output. However, in Section 6, we explain a low-cost way to upgrade our live protocols into terminating versions that allow the parties to halt.

To define asynchronous round complexity, we imagine an external clock. If a protocol runs in R rounds, then it is live or terminating, and 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. Note that this definition matches the notion defined as "time complexity" in [4].

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 $(y_i, g_i) \in (\mathcal{M} \times \{1, \ldots, k\}) \cup \{(\bot, 0)\}$. 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 $(y, g) \neq (\bot, 0)$ 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).

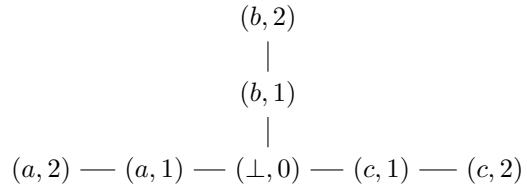

Figure 1 Observe that k-graded consensus with the input domain $\mathcal{M}$ is the same problem as edge agreement in the spider tree with the center $(\bot, 0)$ and the path $((m, 1), \ldots, (m, k))$ attached to it for each $m \in \mathcal{M}$ when the parties can only have the tree's leaves as edge agreement inputs, with each leaf input (m, k) in bijection with the k-graded consensus input m [4]. Note that this equivalence requires intrusion tolerance, which some works (e.g. [9]) do not include as a graded consensus property.

In this work, we use both binary 2-graded consensus (with $|\mathcal{M}| = 2$) and multivalued 2-graded consensus (with $|\mathcal{M}| > 2$). For the latter, there is a 9-round protocol in the literature that costs $\mathcal{O}(n^2)$ messages of size $\mathcal{O}(\log |\mathcal{M}|)$ [4]. This protocol is enough for our asymptotic

complexity results. However, it achieves a property called binding [4] that we do not need, and without this property, 6 rounds suffice. We show this in the appendix by constructing a family of multivalued 2^k-graded consensus protocols $\mathsf{GC}_{2^0}, \mathsf{GC}_{2^1}, \mathsf{GC}_{2^2}, \mathsf{GC}_{2^3}, \ldots$ which each take $3k+3$ rounds, with $\mathcal{O}(n^2)$ messages of size $\mathcal{O}(\log k + \log |\mathcal{M}|)$ per round. We obtain this family by constructing a 1-graded consensus protocol GC_1, and by repeatedly grade-doubling it.[1]

3 Overview & Contributions

Our first contribution is a new protocol for edge agreement in finite trees. Against byzantine faults, this problem was first studied by Nowak and Rybicki [29]. 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 arises from the party ID values that identify each witness technique reliable broadcast's sender party. Meanwhile, we achieve edge agreement in at most $6h(T) + 1$ rounds, where $h(T)$ is a property which we formally define in Section 4 as the maximum of the heights of T's centroid decompositions [30]. 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 such as paths etc.) worse than Nowak and Rybicki's. However, our rounds 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 Section 4, we present a parametrized recursive protocol $\mathsf{TC}(T)$ for edge agreement in a 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 one of its neighbors w_k. However, 2-graded consensus does not guarantee this, as 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 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 Section 5, we present a protocol for edge agreement in $\mathbb{Z}$ that on a high level works as follows (with some complications that we skip over for now due to our use of 2-graded consensus instead of byzantine agreement).

[1] Repeated grade-doubling is a standard method to achieve 2^k-graded consensus in $\mathcal{O}(k)$ rounds [18, 5, 25], though note that synchronous networks allow k^k-graded consensus in $\mathcal{O}(k)$ rounds [20].

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 simplify the comparisons, we assume for [14], [10], [1] and [5] that the inputs are multiples of ε.

Threshold	Bits Sent / Round	Round Complexity[a]	Relaxation[b]	Source
$t < \frac{n}{5}$	$\mathcal{O}(n^2 \log \frac{1}{\varepsilon})$	$\mathcal{O}(\log \frac{1}{\varepsilon})$	0	[14]
$t < \frac{n}{3}$	$\mathcal{O}(n^3 \log \frac{n}{\varepsilon})$	$\mathcal{O}(\log \frac{1}{\varepsilon})$	0	[10]
$t < \frac{n}{3}$	$\mathcal{O}(n^3 \log \frac{n}{\varepsilon})$[c]	$\mathcal{O}(\log \frac{S}{\varepsilon})$	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	[5]
$t < \frac{n}{3}$	$\mathcal{O}(n^2 \log\log \frac{M}{\varepsilon})$[d]	$\mathcal{O}(\log \frac{M}{\varepsilon})$	0	this work

a) The round complexities of [14] and [10] depend on the spread of all inputs, including byzantine ones. In the domain $[0,1]$, this spread is at most 1, which gives us the round complexity $\mathcal{O}(\log \frac{1}{\varepsilon})$.
b) The relaxation is how far an honest output is allowed to be from the honest input range $[v_{\mathsf{lo}}, v_{\mathsf{hi}}]$.
c) 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 [3].
d) The $\log\log \frac{M}{\varepsilon}$ factor here is for tags that distinguish messages sent in different protocol iterations.

1. First, the parties reach 2-graded consensus on whether they prefer to agree on the left path $(\dots, -1, 0)$ or the right path $\mathbb{N} = (0, 1, \dots)$, with each party preferring $\mathbb{N}$ iff its input is in $\mathbb{N}$. Below, we explain what the parties then do if they decide to agree in $\mathbb{N}$. Otherwise, they follow the same steps, but with mirrored (sign-flipped) inputs and outputs.
2. The parties run exponential search with the phases $k = 0, 1, \dots$; where in each phase k they reach 2-graded consensus on if they have inputs in the left path $(2^k - 1, \dots, 2^{k+1} - 1)$ or the right path $(2^{k+1}, \dots)$. If they decide on the left path, then they reach edge agreement in it using our protocol for edge agreement in finite trees. Otherwise, if they decide on the right path, then they increment the phase counter k and continue exponential search.
3. Instead of directly using exponential search for edge agreement in $\mathbb{N}$, we take inspiration from [6], and design a two-stage protocol that is asymptotically twice as round-efficient. Roughly speaking, the parties run the protocol we described above based on exponential search to approximately agree on some k such that the path $(2^k - 1, \dots, 2^{k+1} - 1)$ contains safe output values w.r.t. convex validity, and then they reach edge agreement in this path.

When the maximum honest input magnitude is M (when the honest input that is most distant from 0 is either M or $-M$), 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 Section 7, 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 factor 6 in the round complexity here is due to us using our 6-round 2-graded consensus protocol. If for any other r we used a r-round protocol instead, like the 2-round 2-graded consensus protocol in [4] that tolerates $t < \frac{n}{5}$ faults, then this factor would be r instead.

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 [5]. 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 in $\mathcal{O}(\log(\frac{S}{\varepsilon} \log \frac{S}{\varepsilon}) + \log(\lambda \log n))$ rounds with $\mathcal{O}(\ell n^2 \frac{S}{\varepsilon}(\log(\frac{S}{\varepsilon} \log \frac{S}{\varepsilon}) +$

$\log(\lambda \log n)))$ bits of communication, while relaxing validity by allowing outputs outside the range of the honest inputs by at most S. They use the parameter λ here to assume bounds on the honest inputs that hold except with a probability negligible in λ, and allow their protocol to fail if these bounds are violated. 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 $S \geq n \cdot \varepsilon$.

4 Edge Agreement in a Tree

In $\mathbb{R}^d$, a set $Z \subseteq \mathbb{R}^d$ is straight-line convex if for all $z_1, z_2 \in Z$ it contains the line segment (the shortest path in $\mathbb{R}^d$) that connects z_1 and z_2. This definition translates to convexity on a tree $T = (V, E)$, where a set $Z \subseteq V$ is convex if for all $z_1, z_2 \in Z$ the set Z contains all the vertices on the shortest path (the only path) between z_1 and z_2 [29, 11]. Hence, we can define convex hulls on T, where the convex hull $\langle Z \rangle$ of any vertex set $Z \subseteq V$ is the set that consists of the vertices in Z and every other vertex $v \in V$ that is on the path between some $z_1, z_2 \in Z$.

For edge agreement, the convex validity property is that when the honest parties have the set of inputs X, they obtain outputs in $\langle X \rangle$. To achieve this, we rely on the following fact:

▶ **Proposition 1.** *For every tree $T = (V, E)$, $Z \subseteq V$ and $Y \subseteq \langle Z \rangle$ it holds that $\langle Y \rangle \subseteq \langle Z \rangle$.*

Proof. Observe that for any tree $T = (V, E)$ and any $Z \subseteq V$, the graph $T[\langle Z \rangle]$ (the subgraph of T induced by $\langle Z \rangle$) is a connected union of paths from T, which makes $T[\langle Z \rangle]$ a tree itself. For any $Y \subseteq \langle Z \rangle$, the tree $T[\langle Z \rangle]$ contains every $y_1, y_2 \in Y$, and as $T[\langle Z \rangle]$ is a tree it also contains the path which connects y_1 and y_2. That is, $T[\langle Z \rangle]$ contains every vertex in $\langle Y \rangle$. ◀

Every finite tree T has a set of *centroid* vertices (either one vertex or two adjacent ones) such that if one deletes a centroid vertex σ from T, then every component tree of the resulting forest $T \setminus \{\sigma\}$ has at most half as many vertices as T [24, 30]. For any finite tree T, one can recursively define a centroid decomposition of T to be a rooted tree T' with the following properties (helpfully visualized in [30]):

- The root of T' is a centroid vertex σ of T.
- If in T the centroid σ has exactly d neighbors $w_1, \ldots, w_d$ for any $d \geq 0$, then in T' the root σ has exactly d child subtrees $W_1, \ldots, W_d$, such that for all $k \in \{1, \ldots, d\}$ the subtree W_k is a centroid decomposition of the tree component that contains w_k in the forest $T \setminus \{\sigma\}$.

Finally, let us define the *centroid decomposition height* $h(T)$ of a finite tree $T = (V, E)$ to be the maximum height of any centroid decomposition of T. The recursive definition of a centroid decomposition above allows one to prove by induction that $h(T) \leq \lceil \log_2 |V| \rceil$ (where $|V|$ is the number of vertices), and this bound is tight if T is a path and $|V|$ is a power of 2. However, there are trees that have low centroid decomposition heights despite having many vertices. For example, if T is a star, then $h(T) = 1$ no matter how many vertices T has, as removing a star's unique centroid (its center vertex) leaves behind a forest of isolated vertices.

Below, we present a recursive protocol $\mathsf{TC}(T)$ based on centroid decomposition for edge agreement in a finite tree T. If T has at most two vertices, then each party P_i just outputs its input vertex. Otherwise, the parties let σ be the minimum-index centroid of T, and run our 2-graded consensus protocol GC_2 to either directly output σ, or to reduce edge agreement in T to edge agreement in a component tree T' of $T \setminus \{\sigma\}$, handled with a recursive $\mathsf{TC}(T')$ instance. The recursion depth is at most $h(T)$ since each recursive call represents a step from a vertex to its child in a centroid decomposition of T whose height is upper bounded by $h(T)$.

Protocol $\mathsf{TC}(T)$

Code for a party P_i with the input (v_i)

1: **if** T has 1 or 2 vertices **then**
2: output v_i and do not run the rest of the protocol
3: let σ be the minimum-index centroid vertex of T, of degree $d \geq 2$
4: let $w_1, \ldots, w_d$ be the neighbors of σ, sorted by vertex index
5: let H_j be the tree component of $T \setminus \{\sigma\}$ that contains w_j for each $j \in \{1, \ldots, d\}$
6: run an instance of GC_2 with the other parties where your input is 0 if $v_i = \sigma$, and otherwise your input is the unique index j such that $v_i \in H_j$
7: wait until you output some (k, g) from GC_2
8: **if** $k = 0$ **then**
9: output σ
10: **else if** $g \geq 1$ **then**
11: let $v_i^{\mathsf{next}} \leftarrow v_i$ if $g = 2 \wedge v_i \in H_k$, and let $v_i^{\mathsf{next}} \leftarrow w_k$ otherwise
12: let $T_i^{\mathsf{next}} \leftarrow H_k$
13: if $g = 1$, then multicast $\langle \texttt{KVAL}, k \rangle$
14: **else**
15: output σ
16: multicast `CENTER`
17: wait until you have received the message $\langle \texttt{KVAL}, k \rangle$ for some k from $t+1$ parties, and let $(v_i^{\mathsf{next}}, T_i^{\mathsf{next}}) \leftarrow (w_k, H_k)$ when this happens
18: run an instance of $\mathsf{TC}(T_i^{\mathsf{next}})$ with the other parties where your input is v_i^{next}
19: if $g \geq 1$, then when you output y from the $\mathsf{TC}(T_i^{\mathsf{next}})$, output y from $\mathsf{TC}(T)$
20: **upon** receiving `CENTER` from $t+1$ parties **do**
21: output σ if you haven't output before and ignore future output commands

▷ *Note that P_i should run Line 21 as soon as it receives `CENTER` from $t+1$ parties, even if it is waiting for something else, e.g. waiting to output from GC_2 on Line 7*

The idea behind $\mathsf{TC}(T)$ when T has 3 or more vertices is that either there is a component H_k of $T \setminus \{\sigma\}$ that contains every honest input vertex v_i, or there is no such component.

- In the former case where there is such a component, every honest party P_i runs GC_2 with the input k, outputs $(k, 2)$ from it, lets $(v_i^{\mathsf{next}}, T_i^{\mathsf{next}}) = (v_i, H_k)$, and obtains its final output from a recursive $\mathsf{TC}(H_k)$ instance which it runs with the input $v_i^{\mathsf{next}} = v_i$. Thus, edge agreement in T is reduced to edge agreement in H_k, which the parties reach via $\mathsf{TC}(H_k)$.
- In the latter case where there is no such component, there are some honest inputs v_i and v_j such that either $\sigma \subseteq \{v_i, v_j\}$ or v_i and v_j are in different components of $T \setminus \{\sigma\}$. So, σ is on the path which connects v_i and v_j in T, which makes it a safe output vertex w.r.t. convex validity. Moreover, if some honest party P_i outputs (k, g) from GC_2 for some $k \notin \{\bot, 0\}$ and either $g = 1$ or P_i's $\mathsf{TC}(T)$ input v_i is not in H_k, then some but not all of the honest parties have inputs in H_k, which means that w_k is a safe output vertex w.r.t. convex validity as it is incident to the edge that connects H_k with the rest of the tree. With these in mind, we assign each GC_2 output a behavior such that no matter which two adjacent GC_2 outputs the parties settle on, they behave in a compatible manner that leads to edge agreement.

▶ **Theorem 2.** *For any finite tree T, suppose the honest parties run $\mathsf{TC}(T)$ with input vertices in T. Then, they reach edge agreement in T based on their input vertices in at most $6h(T)$ rounds if they have a common input vertex, and in at most $6h(T) + 1$ rounds otherwise.*

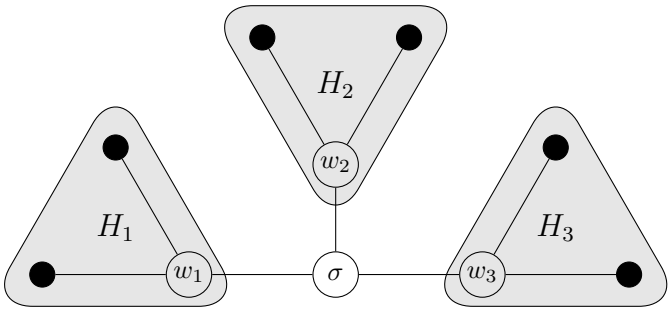

Figure 2 Let T be the tree depicted above with the unique centroid σ and the subtrees H_1, H_2, H_3 (the components of $T \setminus \{\sigma\}$). If the parties run $\mathsf{TC}(T)$, then each party with the input σ runs GC_2 with the input 0, and each party P_i with an input $v_i \neq \sigma$ lets its GC_2 input be the index $k \in \{1, 2, 3\}$ such that $v_i \in H_k$. If for some $k \in \{1, 2, 3\}$ every input v_i is in H_k, then the parties all run GC_2 with the input k and thus output $(k, 2)$ from it. Otherwise, σ is in the convex hull of the input vertices v_i.

Proof. Below, we show for any finite tree T that if $\mathsf{TC}(H)$ works well (in accordance with the theorem) for every tree H such that $h(H) < h(T)$, then $\mathsf{TC}(T)$ also works well. So, by strong induction on $h(T)$, $\mathsf{TC}(T)$ works well for every finite tree T, no matter what $h(T) \geq 0$ is.

In the base case where T has at most 2 vertices, the parties reach edge agreement in 0 rounds by outputting their inputs. This is what happens when $h(T) = 0$, as $h(T) = 0$ iff T is a vertex. For the rest of the proof, suppose T has at least 3 vertices, which implies $h(T) \geq 1$.

Let σ be the minimum-index centroid vertex of T with the neighbors $w_1, \ldots, w_d$ sorted by vertex index, and let $H_1, \ldots, H_d$ be the corresponding tree components of $T \setminus \{\sigma\}$ such that H_j contains w_j for all j. Observe that $h(T)$ is greater than $h(H_j)$ for all j, as T has some centroid decomposition rooted at σ, with d child subtrees $W_1, \ldots, W_d$ attached to σ where each subtree W_j is a centroid decomposition of H_j of height $h(H_j)$. So, our inductive assumption tells us that $\mathsf{TC}(H_j)$ works well for all $j \in \{1, \ldots, d\}$.

First, let us consider the simpler scenario, which is when there exists some $k \in \{1, \ldots d\}$ such that the subtree H_k contains every honest input vertex. In this scenario, the honest parties all run GC_2 with the input k, and thus all output $(k, 2)$ from it. Then, each honest party P_i sets $(v_i^{\mathsf{next}}, T_i^{\mathsf{next}}) = (v_i, H_k)$, runs a common instance of $\mathsf{TC}(T_i^{\mathsf{next}}) = \mathsf{TC}(H_k)$ with the other parties where its input is $v_i^{\mathsf{next}} = v_i$, and obtains its final output from this recursive $\mathsf{TC}(H_k)$ instance. Therefore, edge agreement in T follows from edge agreement in H_k, which the parties reach via $\mathsf{TC}(H_k)$. The round complexity of $\mathsf{TC}(T)$ here is at most that of GC_2 and $\mathsf{TC}(H_k)$ added together; which is always at most $6 + 6h(H_k) + 1 \leq 6h(T) + 1$, and is at most $6 + 6h(H_k) \leq 6h(T)$ if the honest parties run $\mathsf{TC}(T)$ (and thus $\mathsf{TC}(H_k)$) with a common input. Note that the honest parties do not output σ on Line 21 in this scenario since none of them obtain the grade 0 from GC_2, which means that none of them multicast `CENTER` and thus that none of them receive the message `CENTER` from $t + 1$ parties.

Now, let us consider the more complicated scenario where none of the subtrees $H_1, \ldots, H_d$ contains every honest input vertex. Then, there exist some distinct honest inputs v_i and v_j such that either $\sigma \in \{v_i, v_j\}$, or v_i and v_j are in different components of $T \setminus \{\sigma\}$. In either case, σ is on the path that connects v_i and v_j in T, which means that σ is in the convex hull of the honest inputs. In addition, if for some $k \in \{1, \ldots, d\}$ an honest party P_i outputs (k, g) from GC_2 and either $g = 1$ or P_i's input v_i is not in H_k, then some but not all of the honest parties have inputs in H_k (the "some" part by GC_2's intrusion tolerance and the "not all" part by either P_i's grade being below 2 or by P_i's input not being in H_k), and this places w_k

inside the convex hull of the honest inputs since w_k is on every path in T that connects the honest inputs in H_k with the honest inputs outside H_k. With these in mind, let us consider all the ways a $\mathsf{TC}(T)$ execution can go, depending on the honest parties' GC_2 outputs.

- It could happen that the honest parties all output $(k, 2)$ or $(k, 1)$ from GC_2 for some $k \in \{1, \ldots, d\}$, with at least one outputting $(k, 2)$. This situation is similar to the one we considered previously, except for the fact that some honest parties P_i (those with inputs outside H_k and those with the GC_2 grade 1) set $v_i^{\mathsf{next}} = w_k$ instead of setting $v_i^{\mathsf{next}} = v_i$. By Proposition 1, them doing this does not impact convex validity, as the convex hull $\langle V \rangle$ of the honest $\mathsf{TC}(T)$ input vertices $V = \bigcup_{P_i \text{ is honest}} \{v_i\}$ is a superset of the convex hull of $\{w_k\} \cup V \subseteq \langle V \rangle$. So, edge agreement in T follows from edge agreement in H_k (in at most $6 + 6h(H_k) + 1 \leq 6h(T) + 1$ rounds), because after running GC_2, the honest parties all run $\mathsf{TC}(H_k)$ with safe inputs in H_k and all obtain their final outputs from $\mathsf{TC}(H_k)$. Again in this case, the parties do not output σ on Line 21 since none of them multicast `CENTER`.
- It could happen that the honest parties all obtain GC_2 outputs in $\{(\perp, 0), (0, 1), (0, 2)\}$. Then, every honest party outputs σ, either directly on Line 9 or Line 15 after outputting from GC_2, or even earlier by receiving $t + 1$ `CENTER` messages. In either case, the honest parties reach exact agreement on the safe vertex σ, in at most $6 \leq 6h(T) + 1$ rounds.
- Finally, it could happen that the honest parties all output $(k, 1)$ or $(\perp, 0)$ from GC_2 for some $k \in \{1, \ldots, d\}$, with at least one outputting $(k, 1)$. Then, every honest party P_i that runs $\mathsf{TC}(T_i^{\mathsf{next}})$ does so with the tree T_i^{next} set to H_k and with the input vertex $v_i^{\mathsf{next}} = w_k$. For the honest parties that output $(k, 1)$ from GC_2, this follows from Line 11 and Line 12. Meanwhile, for an honest party that outputs $(\perp, 0)$ and thus sets $(v_i^{\mathsf{next}}, T_i^{\mathsf{next}}) = (w_{k'}, H_{k'})$ on Line 17 upon receiving the message $(\texttt{KVAL}, k')$ from $t + 1$ parties; this follows from k' being equal to k due to every honest `KVAL` message being on k and this making k the only value on which a party can receive $t+1$ `KVAL` messages. From all of these, we conclude that the honest parties can only output the safe vertex σ (on Line 15 or 21), or output σ's safe neighbor w_k after outputting w_k from $\mathsf{TC}(H_k)$, which the honest parties can only run with the input w_k. It remains to show liveness. Observe that since $n - t \geq 2t + 1$, either $t + 1$ honest parties output $(\perp, 0)$ from GC_2, or $t + 1$ honest parties output $(k, 1)$ from GC_2.
  - If the former happens, then $t+1$ honest parties multicast `CENTER` after outputting $(\perp, 0)$ from GC_2. Hence, after one round following GC_2 (i.e. $7 \leq 6h(T) + 1$ rounds after $\mathsf{TC}(T)$ begins), every honest party becomes able to output σ on Line 21.
  - If the latter happens, then $t+1$ honest parties multicast $\langle \texttt{KVAL}, k \rangle$ after outputting $(k, 1)$ from GC_2. So, after one round following GC_2 (i.e. $7 \leq 6h(T) + 1$ rounds after $\mathsf{TC}(T)$ begins), the honest parties all learn k and start running $\mathsf{TC}(H_k)$ with the common input w_k. Every honest party outputs w_k from $\mathsf{TC}(H_k)$ once $7 + 6h(H_k) \leq 6h(T) + 1$ rounds have passed, and thus outputs σ or w_k from $\mathsf{TC}(T)$ in at most $6h(T) + 1$ rounds. ◀

Complexity of $\mathsf{TC}(T)$. The round complexity of $\mathsf{TC}(T)$ is at most $6h(T) + 1$, by Theorem 2. If T's maximum degree is Δ, then in each of $\mathsf{TC}(T)$'s at most $h(T)$ recursive iterations the GC_2 instance is run with inputs in $\{0, 1, \ldots, \Delta\}$ and the `KVAL` messages carry values in $\{1, \ldots, \Delta\}$. This means that each iteration costs $\mathcal{O}(n^2)$ messages, each of size at most $\mathcal{O}(\log \Delta + \log(h(T)))$, where the $\log(h(T))$ term is due to the iteration ID tags which distinguish different iterations' messages from each other. So, $\mathsf{TC}(T)$'s total message complexity is $\mathcal{O}(n^2 h(T))$, and its total communication complexity is $\mathcal{O}(n^2 h(T)(\log \Delta + \log(h(T))))$ bits. Finally, for when we need edge agreement in paths, note that $h(T) = q$ if T is a path of length 2^q for any $q \geq 0$.

An Alternative Way. A recent work has reduced edge agreement in a finite tree $T = (V, E)$ to two instances of edge agreement in paths of length $\mathcal{O}(|V|)$ [19]. If one uses this reduction with our TC protocol serving as the path edge agreement protocol, then edge agreement in a finite tree $T = (V, E)$ costs $\mathcal{O}(\log |V|)$ rounds, $\mathcal{O}(n^2 \log |V|)$ messages and $\mathcal{O}(n^2 \log |V| \log\log |V|)$ bits of communication. The drawback of using the reduction here is that it can cost $\Theta(\log |V|)$ rounds even if $h(T) = o(\log |V|)$. For example, the spider tree $T_{k,\mathcal{M}}$ for multivalued k-graded consensus with the input domain $\mathcal{M}$ (the tree defined in Figure 1's caption) has $|\mathcal{M}|k + 1$ vertices, which means that the reduction allows edge agreement in $T_{k,\mathcal{M}}$ in $\Theta(\log |\mathcal{M}| + \log k)$ rounds, while $\mathsf{TC}(T_{k,\mathcal{M}})$ only takes $\mathcal{O}(\log k)$ rounds because $h(T_{k,\mathcal{M}}) = \mathcal{O}(\log k)$.

5 Edge Agreement in Infinite Paths

In this section, we achieve edge agreement in the infinite paths $\mathbb{N}$ and $\mathbb{Z}$. First, we achieve edge agreement in $\mathbb{N}$ with a protocol based on exponential search where we again use GC_2 to repeatedly shrink the agreement domain. Then, we build upon this protocol so that the parties need roughly half as many rounds to reach edge agreement when they have very large inputs. Finally, we upgrade edge agreement in $\mathbb{N}$ to edge agreement $\mathbb{Z}$ with one initial GC_2 iteration.

5.1 Edge Agreement in $\mathbb{N}$

We begin with a sequence of protocols $\mathsf{Exp}_0, \mathsf{Exp}_1, \dots$, with each protocol Exp_j allowing the parties to reach edge agreement in the infinite path $(2^j - 1, \dots)$. In Exp_j, we use GC_2 to reduce edge agreement in $(2^j - 1, \dots)$ to edge agreement in either the left path $(2^j - 1, \dots, 2^{j+1} - 1)$, which the parties can reach via TC, or the right path $(2^{j+1} - 1, \dots)$, which the parties can recursively[2] reach via Exp_{j+1}. At some point, this recursion ends: When the parties run Exp_0 with inputs in $\{0, \dots, M\}$ for some $M \geq 0$, there is eventually some $q \leq \lfloor \log_2(\max(M, 1)) \rfloor$ such that in Exp_q (which recursively appears in Exp_0) the parties do not prefer the right path $(2^{q+1} - 1, \dots)$ as they have inputs below 2^{q+1}, and consequently they reach edge agreement in the finite left path $(2^q - 1, \dots, 2^{q+1} - 1)$.

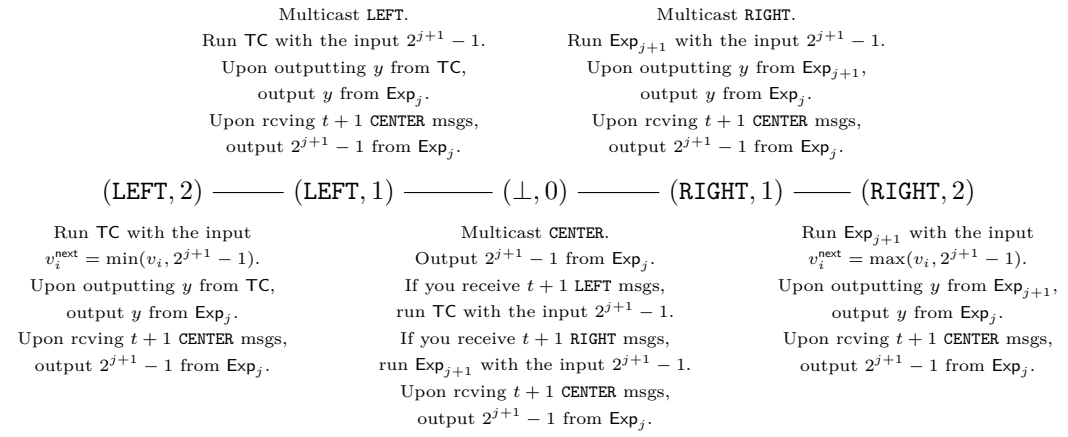

Figure 3 A depiction of how each party behaves in Exp_j, depending on the party's GC_2 output. The crucial observation is that the parties always obtain GC_2 outputs that are adjacent in the figure. Note that a party can output $2^{j+1} - 1$ by receiving $t+1$ CENTER messages before it outputs from GC_2.

[2] Note that one could recast the recursive protocol sequence $\mathsf{Exp}_0, \mathsf{Exp}_1, \dots$ as a single iterative protocol.

Protocol Exp_j

Code for a party P_i with the input v_i

1: join a common instance of GC_2 where your input is `LEFT` if $v_i < 2^{j+1}$ and `RIGHT` otherwise, and wait until you output some (k, g) from GC_2
2: $\mathsf{side} \leftarrow k$
3: **if** $g = 1$ **then**
4: $\quad v_i^{\mathsf{next}} \leftarrow 2^{j+1} - 1$
5: $\quad$ multicast k
6: **else if** $(k, g) = (\texttt{LEFT}, 2)$ **then**
7: $\quad v_i^{\mathsf{next}} \leftarrow \min(v_i, 2^{j+1} - 1)$
8: **else if** $(k, g) = (\texttt{RIGHT}, 2)$ **then**
9: $\quad v_i^{\mathsf{next}} \leftarrow \max(v_i, 2^{j+1} - 1)$
10: **else if** $(k, g) = (\perp, 0)$ **then**
11: $\quad v_i^{\mathsf{next}} \leftarrow 2^{j+1} - 1$
12: $\quad$ output v_i^{next} from Exp_j
13: $\quad$ multicast `CENTER`
14: $\quad$ wait until you have received some $k \in \{\texttt{LEFT}, \texttt{RIGHT}\}$ from $t + 1$ parties, and let $\mathsf{side} \leftarrow k$ when this happens
15: **if** $\mathsf{side} = \texttt{LEFT}$ **then**
16: $\quad$ run an instance of $\mathsf{TC}((2^j - 1, \ldots, 2^{j+1} - 1))$ with the other parties for edge agreement in the path $(2^j - 1, \ldots, 2^{j+1} - 1)$, where your input is v_i^{next}
17: $\quad$ if $g \geq 1$, then when you output y from TC, output y from Exp_j
18: **else if** $\mathsf{side} = \texttt{RIGHT}$ **then**
19: $\quad$ run an instance of Exp_{j+1} with the other parties for edge agreement in the path $(2^{j+1} - 1, \ldots)$, where your input is v_i^{next}
20: $\quad$ if $g \geq 1$, then when you output y from Exp_{j+1}, output y from Exp_j
21: **upon** receiving `CENTER` from $t + 1$ parties **do**
22: $\quad$ output $2^{j+1}-1$ if you haven't output before and ignore future output commands

▷ *Note that P_i should run Line 22 as soon as it receives `CENTER` from $t+1$ parties, even if it is waiting for something else, e.g. waiting to output from GC_2 on Line 1*

▶ **Theorem 3.** *For any positive integer q and any $0 \leq j < q$, suppose the honest parties run Exp_j with inputs in $\{2^j - 1, \ldots, 2^q - 1\}$. Then, they reach edge agreement in the path $\mathbb{N}$ based on their inputs in at most $6(j + 1)$ rounds if they have the common input $2^j - 1$ and in at most $12q - 6j - 5$ rounds otherwise.*

We prove Theorem 3 in the appendix, with a proof similar to the one of Theorem 2. Note that the value q in the theorem is not something that the parties need to know to run Exp_j.

Complexity of Exp_0. Consider an Exp_0 execution with the maximum honest input M. Let $q = \lfloor \log_2(\max(M, 1)) \rfloor + 1$, i.e., let q be the least positive integer such that every honest input is below 2^q. There is a minimum $k \in \{0, \ldots, q-1\}$ such that in Exp_k (recursively a subprotocol of Exp_0 if $k > 0$), the parties do not set $\mathsf{side} = \texttt{RIGHT}$ and therefore do not run Exp_{k+1}. So, the Exp_0 execution consists of the GC_2 instances and the `LEFT`/`CENTER`/`RIGHT` multicasts of $\mathsf{Exp}_0, \mathsf{Exp}_1, \ldots, \mathsf{Exp}_k$, plus the $\mathsf{TC}((2^k - 1, \ldots, 2^{k+1} - 1))$ instance of Exp_k. These total up to $\mathcal{O}(n^2 q) = \mathcal{O}(n^2 \log M)$ messages, each of size $\mathcal{O}(\log q) = \mathcal{O}(\log \log M)$ due to the message tags since for all $j \geq 0$ one can assign $\mathcal{O}(\log j)$-bit tags to Exp_j and to its GC_2 and TC subprotocols. Meanwhile, the round complexity is at most $12q - 5 = 12 \log_2 M + \mathcal{O}(1)$, by Theorem 3.

5.2 Nearly Halving the Round Complexity

The protocol Exp_0 is based on exponential search, which is a search algorithm to find a target value $x \in \mathbb{N}$ with approximately $2\log_2 x$ checks for different choices of v whether $v \leq x$ or not. Each such check roughly corresponds to a 6-round GC_2 instance in Exp_0, and thus Exp_0 takes $12\log_2 M + \mathcal{O}(1)$ rounds by the simple fact that $2 \cdot 6 = 12$. However, it is possible to almost halve the number of checks. The strategy for this, from [6], is to accelerate exponential search by using it to find $q = \lfloor \log_2(x+1) \rfloor$ instead of x, and then, knowing that $2^q - 1 \leq x \leq 2^{q+1} - 2$, to run binary search with the lower and upper bounds $2^q - 1$ and $2^{q+1} - 2$ to find x.

Below, we use an analogous strategy in our protocol $\mathsf{2\text{-}Step}(\mathsf{Agr}_\mathbb{N})$ for edge agreement in $\mathbb{N}$, where we let $\mathsf{Agr}_\mathbb{N}$ be any protocol for edge agreement in $\mathbb{N}$, and accelerate $\mathsf{Agr}_\mathbb{N}$ by using it in a two-stage manner. In $\mathsf{2\text{-}Step}(\mathsf{Agr}_\mathbb{N})$, the parties roughly speaking use $\mathsf{Agr}_\mathbb{N}$ to agree on some k such that the path $(2^k - 1, \ldots, 2^{k+1} - 1)$ intersects the convex hull of the honest inputs, with each party P_i with the input v_i wanting k to be $\lfloor \log_2(v_i + 1) \rfloor$. Then, each party with an input outside this path adopts a new safe value in the path (the path endpoint closest to its value), and finally the parties run TC to reach edge agreement in the path. Naturally, we face the issue that $\mathsf{Agr}_\mathbb{N}$ does not guarantee agreement on a single k, which means that the parties might output different (though adjacent) k values from it and therefore behave differently after outputting from $\mathsf{Agr}_\mathbb{N}$. Luckily, by making each party P_i let its $\mathsf{Agr}_\mathbb{N}$ input be what it wants $5k$ to be $(5\lfloor \log_2(v_i + 1) \rfloor)$ rather than what it wants k to be, we can ensure that the discrepancy between how each party behaves after outputting from $\mathsf{Agr}_\mathbb{N}$ is sufficiently small.

Protocol $\mathsf{2\text{-}Step}(\mathsf{Agr}_\mathbb{N})$

Code for a party P_i with the input v_i

join a common instance of $\mathsf{Agr}_\mathbb{N}$ where your input is $5\lfloor \log_2(v_i + 1) \rfloor$
upon outputting $5k_i + r_i$ from $\mathsf{Agr}_\mathbb{N}$ for some $r_i \in \{0, \ldots 4\}$ **do**
  if $r_i = 0$ **then**
    $v_i^{\mathsf{next}} \leftarrow \min(\max(v_i, 2^{k_i} - 1), 2^{k_i+1} - 1)$
  else
    $v_i^{\mathsf{next}} \leftarrow 2^{k_i+1} - 1$
  if $0 \leq r_i \leq 2$ **then**
    run an instance of $\mathsf{TC}((2^{k_i} - 1, \ldots, 2^{k_i+1} - 1))$ with the other parties for edge agreement in $(2^{k_i} - 1, \ldots, 2^{k_i+1} - 1)$, where your input is v_i^{next}
    if $r_i \leq 1$, then when you output y from TC, output y from $\mathsf{2\text{-}Step}(\mathsf{Agr}_\mathbb{N})$
  if $2 \leq r_i \leq 3$ **then**
    output v_i^{next}
  if $3 \leq r_i \leq 4$ **then**
    run an instance of $\mathsf{TC}((2^{k_i+1} - 1, \ldots, 2^{k_i+2} - 1))$ with the other parties for edge agreement in $(2^{k_i+1} - 1, \ldots, 2^{k_i+2} - 1)$, where your input is v_i^{next}
    if $r_i = 4$, then when you output y from TC, output y from $\mathsf{2\text{-}Step}(\mathsf{Agr}_\mathbb{N})$

▶ **Theorem 4.** *Suppose $\mathsf{Agr}_\mathbb{N}$ is a live protocol for edge agreement in $\mathbb{N}$ which for some function f takes at most $f(M)$ rounds when the honest parties run it with inputs in $\{0, \ldots, M\}$. Then, $\mathsf{2\text{-}Step}(\mathsf{Agr}_\mathbb{N})$ is a live protocol for edge agreement in $\mathbb{N}$ that takes at most $f(5\lfloor \log_2(M+1) \rfloor) + 6\lfloor \log_2(M+1) \rfloor + 1$ rounds when the honest parties run it with inputs in $\{0, \ldots, M\}$.*

We prove Theorem 4 in the appendix, with a proof similar to the one of Theorem 2. Though $\mathsf{2\text{-}Step}(\mathsf{Agr}_\mathbb{N})$ does not explicitly use graded consensus, it depends on the same core idea as TC and Exp: The parties reach graded consensus on how to behave, and thus they behave well

together. We do not need an actual graded consensus protocol in 2-Step($\mathsf{Agr_N}$) to implement this idea as the "grades" in 2-Step($\mathsf{Agr_N}$) are provided directly by $\mathsf{Agr_N}$, in the form of the r_i value which a party P_i obtains as the remainder when it divides its $\mathsf{Agr_N}$ output by 5.

With 2-Step one can obtain a sequence of protocols $\Pi_1 = \mathsf{Exp}_0, \Pi_2 = \text{2-Step}(\Pi_1), \Pi_3 = \text{2-Step}(\Pi_2), \ldots$ for edge agreement in $\mathbb{N}$, with each protocol Π_k corresponding to the algorithm B_k in [6]. Letting $f_k(M)$ be Π_k's round complexity depending on the maximum honest input M, we have $f_1(M) = 12\log_2 M + \mathcal{O}(1)$ and $f_{k+1}(M) = f_k(5\lfloor\log_2(M+1)\rfloor) + 6\lfloor\log_2(M+1)\rfloor + 1$ for all $k \geq 1$. One can show by induction that $f_k(M) = 6(L^{(k)}(M) + \sum_{j=1}^{k} L^{(j)}(M)) + \mathcal{O}(k)$ for all $k \geq 1$, where $L^{(1)}(v) = \lfloor\log_2(v+1)\rfloor$ and $L^{(k)}(v) = L(L^{(k-1)}(v))$ for all $k \geq 2$.

The protocol $\Pi_2 = \text{2-Step}(\mathsf{Exp}_0)$ above notably takes $\frac{1}{2} + \mathcal{O}(\frac{\log\log M}{\log M}) = \frac{1}{2} + o(1)$ times as many rounds as $\Pi_1 = \mathsf{Exp}_0$, and only $1+o(1)$ times as many rounds as the protocols $\Pi_3, \Pi_4, \ldots$ beyond it. Therefore, in the rest of the paper we use 2-Step(Exp_0) for edge agreement in $\mathbb{N}$.

Complexity of 2-Step(Exp_0). Consider a 2-Step(Exp_0) execution with the maximum honest input M, and let $q = \lfloor\log_2(M+1)\rfloor$. The initial Exp_0 in 2-Step(Exp_0) costs $12\log_2 q + \mathcal{O}(1)$ rounds, $\mathcal{O}(n^2 \log q)$ messages and $\mathcal{O}(n^2 \log q \log\log q)$ bits of communication in total. Then, in 2-Step(Exp_0)'s remaining $6q+1$ rounds, each party P_i runs an instance of TC. Due to GC_2 and thus TC being fully symmetric protocols with balanced communication, each party P_i (with an Exp_0 output z_i that is upper bounded by $5q$) sends $\mathcal{O}(nz_i) = \mathcal{O}(nq)$ messages and $\mathcal{O}(nz_i \log z_i) = \mathcal{O}(nq \log q)$ bits to the others in TC. So, we conclude that 2-Step(Exp_0) takes $12\log_2 q + 6q + \mathcal{O}(1) = 6\log_2 M + 12\log_2\log_2 M + \mathcal{O}(1)$ rounds, with $\mathcal{O}(n^2 q) = \mathcal{O}(n^2 \log M)$ messages and $\mathcal{O}(n^2 q \log q) = \mathcal{O}(n^2 \log M \log\log M)$ bits of communication in total.

5.3 Edge Agreement in $\mathbb{Z}$

Protocol $\mathsf{Agr_Z}(\mathsf{Agr_N})$

Code for a party P_i with the input v_i

1: join a common instance of GC_2 and a common instance of $\mathsf{Agr_N}$
2: let your GC_2 input be 1 if $v_i \geq 0$, and let it be -1 otherwise
3: wait until you output some (k, g) from GC_2
4: let your $\mathsf{Agr_N}$ input v_i^{next} be $\max(0, k \cdot v_i)$ if $g = 2$, and 0 otherwise
5: **if** $(k, g) = (\perp, 0)$ **then**
6: output 0 from $\mathsf{Agr_Z}(\mathsf{Agr_N})$
7: **else**
8: when you output y from $\mathsf{Agr_N}$, output $k \cdot y$ from $\mathsf{Agr_Z}(\mathsf{Agr_N})$

As before, let $\mathsf{Agr_N}$ be a protocol for edge agreement in $\mathbb{N}$. Observe that the parties can reach edge agreement in $(\ldots, -1, 0)$ by mirroring their inputs (multiplying them by -1), running $\mathsf{Agr_N}$ with their mirrored inputs, and mirroring their $\mathsf{Agr_N}$ outputs. This allows the protocol $\mathsf{Agr_Z}(\mathsf{Agr_N})$ above for edge agreement in $\mathbb{Z}$, where the parties run GC_2 with the input -1 if they have negative $\mathsf{Agr_Z}$ inputs and the input 1 otherwise, and afterwards use their GC_2 outputs to determine how they should run $\mathsf{Agr_N}$. Note that [23] upgrades agreement in $\mathbb{N}$ to agreement in $\mathbb{Z}$ similarly, though with byzantine agreement instead of 2-graded consensus.

▶ **Theorem 5.** *If $\mathsf{Agr_N}$ is a live protocol for edge agreement in $\mathbb{N}$, then $\mathsf{Agr_Z}(\mathsf{Agr_N})$ is a live protocol for edge agreement in $\mathbb{Z}$.*

We prove Theorem 5 in the appendix, with a proof similar to the one of Theorem 2.

Complexity. When the honest parties run $\mathsf{Agr}_\mathsf{Z}(\mathsf{Agr}_\mathsf{N})$ with inputs in $\{-M, \ldots, M\}$ for some $M \geq 0$, they run GC_2, and then run Agr_N with inputs upper bounded by M. So, the Agr_Z outer shell adds an overhead of 6 rounds, $\mathcal{O}(n^2)$ messages and $\mathcal{O}(n^2)$ bits of communication to Agr_N. If $\mathsf{Agr}_\mathsf{N} = \text{2-}\mathsf{Step}(\mathsf{Exp}_0)$, then $\mathsf{Agr}_\mathsf{Z}(\mathsf{Agr}_\mathsf{N})$'s complexity is $6 \log_2 M + \mathcal{O}(\log \log M)$ rounds, $\mathcal{O}(n^2 \log M)$ messages and $\mathcal{O}(n^2 \log M \log \log M)$ bits of communication.

6 Termination

Our edge agreement and graded consensus protocols do not terminate. They require every honest party to run forever, and lose liveness if any honest party ever halts. Fortunately, there is a simple 3-round quadratic-complexity protocol Term which can address this shortcoming. We present it in the appendix. It is the same termination procedure as the one on page 18 in [28], with only cosmetic changes. It relies on the fact that edge agreement implies agreement on at most two values, i.e., implies that that for some y and y' everybody outputs y or y'. Note that unlike our other protocols, Term does not require every honest party to acquire an input.

▶ **Theorem 6.** *In any* Term *execution where the maximum honest message delay is* Δ *and the honest parties only acquire inputs in some set* $\{y, y'\}$*, if by some time* T *either every honest party acquires an input in* $\{y, y'\}$ *or some honest party terminates* Term*, then by the time* $T + 3\Delta$ *every honest party terminates* Term *with an output that is an honest party's input.*

We prove Theorem 6 in the appendix, with a proof similar to one of Theorem 6 in [28].

For any edge agreement protocol Π, there is an edge agreement protocol $\mathsf{Term} \circ \Pi$ in which the parties reach edge agreement with Π and terminate with Term. In it, the parties run Π with their inputs, run Term by using their Π outputs as their Term inputs and terminate the whole protocol upon terminating Term, letting their Term outputs be their final outputs. The composed protocol $\mathsf{Term} \circ \Pi$ inherits Π's agreement and validity properties since each honest Term output is some honest party's Π output. Moreover, if Π achieves liveness in k rounds, then $\mathsf{Term} \circ \Pi$ terminates in $k + 3$ rounds. This is because after k rounds, either every honest has output from Π and thus acquired a Term input, or some honest party has terminated Term. Either way, the honest parties all terminate Term within $k + 3$ rounds of every honest party acquiring a Π input. Note that some honest parties might terminate Term before k rounds have passed and stop running Π before everybody outputs from Π. This might cause Π to lose its liveness, meaning that some honest parties might never output from Π and thus never acquire Term inputs. This is not an issue because if some honest party terminates Term, then every honest party terminates Term, even if not every honest party acquires a Term input.

In Term, each honest party multicasts one constant-size `READY` message and at most one $\langle \texttt{ECHO}, v \rangle$ message for each honest input v. So, the overhead of terminating an edge agreement protocol Π with Term is 3 rounds and $\mathcal{O}(n^2)$ messages which each carry honest Π outputs.

Termination for edge agreement in $\mathbb{Z}$. If the parties terminate with Term after reaching edge agreement in $\mathbb{Z}$ with inputs of magnitude at most M, then each honest Term input is an integer in $\{-M, \ldots, M\}$, with an $\mathcal{O}(\log M)$-bit representation. So, for any live protocol Agr_Z for edge agreement in $\mathbb{Z}$, $\mathsf{Term} \circ \mathsf{Agr}_\mathsf{Z}$ is a terminating protocol for edge agreement in $\mathbb{Z}$ which costs 3 more rounds, $\mathcal{O}(n^2)$ more messages and $\mathcal{O}(n^2 \log M)$ more bits of communication than Agr_Z.

Termination for edge agreement in trees. If the parties run Term to terminate after they reach edge agreement in a tree $T = (V, E)$, then each honest Term input is a vertex in V, with an $\mathcal{O}(\log |V|)$-bit representation. So, $\mathsf{Term} \circ \mathsf{TC}(T)$ is a terminating version of $\mathsf{TC}(T)$ that costs

3 rounds, $\mathcal{O}(n^2)$ messages and $\mathcal{O}(n^2 \log |V|)$ bits of communication more than $\mathsf{TC}(T)$. Note that Term here does not asymptotically cost more communication than $\mathsf{TC}(T)$ because we have $h(T) = \Omega(\log_\Delta |V|)$, where Δ is T's maximum degree. To see why, let $h_\Delta(k)$ for any $k \geq 1$ be the minimum value $h(T)$ can have for any tree T of maximum degree at most $\Delta \geq 2$ with at least k vertices, and observe that $h_\Delta(k) \geq 1 + h_\Delta(\lceil \frac{k-1}{\Delta} \rceil)$ for all $k \geq 2$ as removing a centroid of degree at most Δ from a tree with at least $k \geq 2$ vertices creates a forest whose largest component has at least $\lceil \frac{k-1}{\Delta} \rceil$ vertices. By induction, this implies $h_\Delta(\Delta^k) \geq k$ for all $k \geq 0$.

7 Extension to Real Numbers

Termination is harder for approximate agreement in $\mathbb{R}$. As approximate agreement in $\mathbb{R}$ does not entail agreement on two values, Term alone cannot directly provide termination. So, to achieve approximate agreement in $\mathbb{R}$ with termination, we reduce it to edge agreement in $\mathbb{Z}$.

▶ **Theorem 7.** *For all $\varepsilon > 0$, ε-agreement in $\mathbb{R}$ with the maximum honest input magnitude M can be reduced to edge agreement in $\mathbb{Z}$ with the maximum honest input magnitude $\lceil \frac{2M}{\varepsilon} - \frac{1}{2} \rceil$.*

Proof. We reduce 2-agreement in $\mathbb{R}$ with the maximum honest input magnitude M to edge agreement in $\mathbb{Z}$ with the maximum honest input magnitude $\lceil M - \frac{1}{2} \rceil$. This reduction implies the theorem since the parties can reach ε-agreement in $\mathbb{R}$ by multiplying their inputs by $\frac{2}{\varepsilon}$, reaching 2-agreement, and dividing their outputs by $\frac{2}{\varepsilon}$.

Each party P_i rounds its 2-agreement input v_i to the nearest integer v'_i, rounding towards 0 in the case of ties. Then, the parties reach edge agreement in $\mathbb{Z}$ with their rounded integers of magnitude at most $\lceil M - \frac{1}{2} \rceil$, and each party P_i obtains some edge agreement output y'_i. For all honest parties P_i and P_j it holds that $|y'_i - y'_j| \leq 1$ and that $v'_a \leq y'_i \leq v'_b$, where a and b are respectively the indices of the honest parties with the minimum and maximum edge agreement inputs. Observe that this implies $v_a - \frac{1}{2} \leq v'_a \leq y'_i \leq v'_b \leq v_b + \frac{1}{2}$ for all honest P_i.

Afterwards, each party P_i uses its 2-agreement input v_i to convert y'_i into its 2-agreement output y_i by letting $y_i = \min(y'_i + \frac{1}{2}, v_i)$ if $y'_i \leq v_i$, and $y_i = \max(y'_i - \frac{1}{2}, v_i)$ otherwise. That is, P_i obtains y_i by moving from y'_i towards its input v_i by a distance of at most $\frac{1}{2}$. For all honest P_i and P_j it holds that $\min(v_a, v_i) \leq y_i \leq \max(v_b, v_i)$, which implies validity, and that $|y_i - y_j| \leq |y_i - y'_i| + |y'_i - y'_j| + |y'_j - y_j| \leq \frac{1}{2} + 1 + \frac{1}{2} = 2$, which implies 2-agreement. ◀

As we can achieve edge agreement in $\mathbb{Z}$ in $6 \log_2 M + \mathcal{O}(\log \log M)$ rounds with $\mathcal{O}(n^2 \log M)$ messages and $\mathcal{O}(n^2 \log M \log \log M)$ bits of communication, the reduction 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. Note that M here is (as always in this paper) not a parameter which must be known in advance, but a dynamic value that depends on the honest parties' inputs.

Our reduction allows the parties to reach ε-agreement in $\mathbb{R}$ by only sending each other discrete messages. So, it allows approximate agreement without any value rounding errors. In contrast, protocols where the parties send values in $\mathbb{R}$ [14, 1, 21, 5] must in order to avoid rounding errors assume that the inputs in $\mathbb{R}$ fit ℓ-bit strings, with ℓ affecting complexity.

Future Work. It remains open to design a protocol for ε-agreement in $\mathbb{R}$ that tolerates $t < \frac{n}{3}$ faults with quadratic communication in $\mathcal{O}(\log \frac{S}{\varepsilon})$ rounds, where S is the honest input spread. Our protocol takes $\mathcal{O}(\log \frac{M}{\varepsilon})$ rounds, where possibly $M \gg S$, while [1] takes $\mathcal{O}(\log \frac{S}{\varepsilon})$ rounds but requires cubic communication. In fact, this task is open even for the synchronous setting. The classical synchronous protocol in [14] takes as many rounds as the adversary wants since its complexity scales with the the spread of *all* inputs, including byzantine ones. Solving this issue remains open, especially in the asynchronous setting where a new approach seems necessary.

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 Nicolas Alhaddad, Sourav Das, Sisi Duan, Ling Ren, Mayank Varia, Zhuolun Xiang, and Haibin Zhang. Balanced byzantine reliable broadcast with near-optimal communication and improved computation. In *Proceedings of the 2022 ACM Symposium on Principles of Distributed Computing*, PODC '22, pages 399–417, New York, NY, USA, 2022. Association for Computing Machinery. doi:10.1145/3519270.3538475.

4 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.

5 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.

6 Jon Louis Bentley and Andrew Chi-Chih Yao. An almost optimal algorithm for unbounded searching. *Information Processing Letters*, 5(3):82–87, 1976. doi:10.1016/0020-0190(76)90071-5.

7 Erica Blum, Jonathan Katz, and Julian Loss. Synchronous consensus with optimal asynchronous fallback guarantees. In *Theory of Cryptography — TCC 2019*, Cham, Switzerland, 2019. Springer International Publishing. doi:10.1007/978-3-030-36030-6_6.

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

9 Jing Chen and Silvio Micali. Algorand: A secure and efficient distributed ledger. *Theor. Comput. Sci.*, 777(C):155–183, July 2019. doi:10.1016/j.tcs.2019.02.001.

10 B. A. Coan. A compiler that increases the fault tolerance of asynchronous protocols. *IEEE Trans. Comput.*, 37(12):1541–1553, December 1988. doi:10.1109/12.9732.

11 Andrei Constantinescu, Diana Ghinea, Roger Wattenhofer, and Floris Westermann. Convex Consensus with Asynchronous Fallback. In *38th International Symposium on Distributed Computing (DISC 2024)*, volume 319 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 15:1–15:23, Dagstuhl, Germany, 2024. Schloss Dagstuhl – Leibniz-Zentrum für Informatik. doi:10.4230/LIPIcs.DISC.2024.15.

12 Giovanni Deligios and Mose Mizrahi Erbes. Closing the efficiency gap between synchronous and network-agnostic consensus. In *Advances in Cryptology – EUROCRYPT 2024*, pages 432–461, Cham, 2024. Springer Nature Switzerland. doi:10.1007/978-3-031-58740-5_15.

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

14 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.

15 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.

16 Maya Dotan, Gilad Stern, and Aviv Zohar. Validated byzantine asynchronous multidimensional approximate agreement, 2022. doi:10.48550/arXiv.2211.02126.

17 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.

18 Matthias Fitzi, Chen-Da Liu-Zhang, and Julian Loss. A new way to achieve round-efficient byzantine agreement. In *Proceedings of the 2021 ACM Symposium on Principles of Distributed Computing*, PODC '21, pages 355–362, New York, NY, USA, 2021. Association for Computing Machinery. doi:10.1145/3465084.3467907.

19 Marc Fuchs, Diana Ghinea, and Zahra Parsaeian. Brief announcement: Towards round-optimal approximate agreement on trees. In *Proceedings of the ACM Symposium on Principles of Distributed Computing*, PODC '25, pages 54–57, New York, NY, USA, 2025. Association for Computing Machinery. doi:10.1145/3732772.3733555.

20 Diana Ghinea, Vipul Goyal, and Chen-Da Liu-Zhang. Round-optimal byzantine agreement. In *Advances in Cryptology – EUROCRYPT 2022*, pages 96–119, Cham, 2022. Springer International Publishing. doi:10.1007/978-3-031-06944-4_4.

21 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.

22 Diana Ghinea, Chen-Da Liu-Zhang, and Roger Wattenhofer. Multidimensional approximate agreement with asynchronous fallback. In *Proceedings of the 35th ACM Symposium on Parallelism in Algorithms and Architectures*, SPAA '23, pages 141–151, New York, NY, USA, 2023. Association for Computing Machinery. doi:10.1145/3558481.3591105.

23 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.

24 Camille Jordan. Sur les assemblages de lignes. *Journal für die reine und angewandte Mathematik*, 70:185–190, 1869. URL: http://eudml.org/doc/148084.

25 Simon Holmgaard Kamp. A new way to achieve round-efficient asynchronous byzantine agreement. Cryptology ePrint Archive, Paper 2025/143, 2025. URL: https://eprint.iacr.org/2025/143.

26 Hammurabi Mendes, Maurice Herlihy, Nitin Vaidya, and Vijay K. Garg. Multidimensional agreement in byzantine systems. *Distrib. Comput.*, 28(6):423–441, December 2015. doi:10.1007/s00446-014-0240-5.

27 Mose Mizrahi Erbes and Roger Wattenhofer. Brief Announcement: Asynchronous Approximate Agreement with Quadratic Communication. In Dariusz R. Kowalski, editor, *39th International Symposium on Distributed Computing (DISC 2025)*, volume 356 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 61:1–61:7, Dagstuhl, Germany, 2025. Schloss Dagstuhl – Leibniz-Zentrum für Informatik. doi:10.4230/LIPIcs.DISC.2025.61.

28 Mose Mizrahi Erbes and Roger Wattenhofer. Quit-Resistant Reliable Broadcast and Efficient Terminating Gather. In *28th International Conference on Principles of Distributed Systems (OPODIS 2024)*, volume 324 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 15:1–15:22, Dagstuhl, Germany, 2025. Schloss Dagstuhl – Leibniz-Zentrum für Informatik. doi:10.4230/LIPIcs.OPODIS.2024.15.

29 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.

30 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.

A Graded Consensus Protocols

In this section, we design a family of 2^k-graded consensus protocols $\mathsf{GC}_{2^0}, \mathsf{GC}_{2^1}, \mathsf{GC}_{2^2}, \ldots$ that support inputs in $\{0,1\}^\ell$ for any public length parameter $\ell \geq 1$. The complexity of each protocol GC_{2^k} is $3k+3$ rounds, $\mathcal{O}(kn^2)$ messages and $\mathcal{O}(kn^2(\ell+\log k))$ bits of communication. To obtain these protocols we begin with the 3-round GC_1, and then for all $k \geq 1$ obtain GC_{2^k} by grade-doubling $\mathsf{GC}_{2^{k-1}}$ in 3 rounds. While we only use GC_2 in this paper, note that grades above 2 can also be useful, as 2^k-graded consensus forms the "expand" part of the "expand-and-extract" paradigm of achieving byzantine agreement with an $\mathcal{O}(2^{-k})$ error probability [18].

As 2^k-graded consensus is a special case of edge agreement with the centroid decomposition height $k+1$ (see Figure 1), our edge agreement protocol TC is also a 2^k-graded consensus protocol, but one that takes up to $6k+7$ rounds. The more restricted definition of 2^k-graded consensus allow us to achieve it in $3k+3$ rounds instead. This is the round complexity of the binary 2^k-graded consensus protocol in [5], and it is for all $k \geq 0$ lower than the round complexity of any previous multivalued 2^k-graded consensus protocol which we know of that tolerates $t < \frac{n}{3}$ faults with perfect security. Note that 2^k-graded consensus can be achieved in less rounds than $3k+3$ if $t < \frac{n}{5}$ ($k+1$ rounds [14, 4]) or if the parties can use a cryptographic setup ($k+2$ rounds, [25]). If in our edge agreement protocols we replaced GC_2 with a faster r-round 2-graded consensus protocol, then our edge agreement protocols would become $\frac{6}{r}$ times faster, excluding some constant round complexity terms that would not change.

A.1 1-Graded Consensus

Below is our live 3-round multivalued 1-graded consensus protocol GC_1.

Protocol GC_1

Code for a party P_i with the input v_i

$V_i^1, V_i^2, \ldots, V_i^\ell \leftarrow \varnothing, \varnothing, \ldots, \varnothing$
$W_i^1, W_i^2, \ldots, W_i^\ell \leftarrow \varnothing, \varnothing, \ldots, \varnothing$
multicast $\langle \texttt{ECHO}, v_i \rangle$
upon receiving echoes ($\texttt{ECHO}$ messages) on $\perp$ or on strings other than v_i from $t+1$ parties **do**
    multicast $\langle \texttt{ECHO}, \perp \rangle$
    output $(\perp, 0)$ if you haven't output before, and keep running
for any bit $b \in \{0,1\}$ and any index $k \in \{1, \ldots, \ell\}$, when you have received echoes on $\perp$ or on strings with the k^{th} bit b from $t+1$ parties **do**
    $V_i^k \leftarrow V_i^k \cup \{b\}$
    if $V_i^k = \{0,1\}$ **then**
        output $(\perp, 0)$ if you haven't output before, and keep running
for any bit $b \in \{0,1\}$ and any index $k \in \{1, \ldots, \ell\}$, when you have received echoes on $\perp$ or on strings with the k^{th} bit b from $2t+1$ parties **do**
    $W_i^k \leftarrow W_i^k \cup \{b\}$
    if $W_i^1, W_i^2, \ldots, W_i^\ell = \{c_1\}, \{c_2\}, \ldots, \{c_\ell\}$ for some $c_1, \ldots, c_\ell \in \{0,1\}^\ell$ **then**
        multicast $\langle \texttt{PROP}, c_1 \| c_2 \| \ldots \| c_\ell \rangle$ ▷ *Here, $\|$ stands for string concatenation.*
upon receiving ($\texttt{PROP}$ messages) on some string v from $n-t$ parties **do**
    let $y = (v, 1)$ if $v = v_i$, and let $y = (\perp, 0)$ otherwise
    output y if you haven't output before, and keep running

This protocol is based on the 4-round protocol AWC in [12]. Our addition is the list of sets $V_i^1, \ldots, V_i^\ell$, which AWC does not have. While according to AWC a party P_i would output $(\perp, 0)$ upon observing that $W_i^k = \{0, 1\}$ for some k, in GC_1 a party P_i outputs $(\perp, 0)$ upon observing that $V_i^k = \{0, 1\}$. As the parties fill their V sets more readily than their W sets (due to Line 7 having a weaker condition than Line 11), the V sets let us shave off one round.

▶ **Theorem 8.** GC_1 *is a secure* 3*-round* 1*-graded consensus protocol with liveness.*

Proof. The simplest property is intrusion tolerance. It follows from the fact that if an honest party outputs $(v, 1)$ for any v, then v is the party's input, which makes it an honest input.

Agreement is also simple. Suppose there are two honest parties P and P' who respectively output $(v, 1)$ and $(v', 1)$. Then, P has received $n - t$ proposals on v, while P' has received $n - t$ proposals on v'. Hence, there are $n - 2t \geq t + 1$ parties, or at least one honest party, who have sent P a proposal on v and P' a proposal on v'. This implies that $v = v'$ because an honest party can only propose one value. The reason why this is true is that an honest party P_i only proposes a value when it adds a bit to one of its W sets $W_i^1, \ldots, W_i^\ell$ and after doing so observes that $|W_i^1| = \cdots = |W_i^\ell| = 1$, which can happen only once.

For validity, suppose the honest parties have a common input $v^* = b_1^* \| \ldots \| b_\ell^* \in \{0, 1\}^\ell$. In this case, they echo v^* (multicast $\langle \texttt{ECHO}, v^* \rangle$), and they do not echo $\perp$ since they do not receive $t + 1$ echoes on values other than v^*. For all $k \in \{1, \ldots, \ell\}$, since no honest party echoes $\perp$ or a string with the k^{th} bit $1 - b_k^*$, the bit b_k^* is the only bit that any honest party P_i can add to V_i^k or W_i^k. Consequently, an honest party can neither output $(\perp, 0)$ on Line 10 nor propose anything other than v^*. This means that an honest party can only output after receiving $n - t$ proposals on v^*, which leads to the party outputting v^* since it has the input v^*.

Finally, for liveness in 3 rounds, suppose that by some time T every honest party acquires its input and thus echoes it. We show via case analysis that every honest party outputs by the time $T + 3\Delta$, where Δ is the maximum honest message delay that the adversary causes.

1. The easier case is if there is no input that $t + 1$ honest parties have. Then, by the time $T + \Delta$, every honest party receives $n - 2t \geq t + 1$ echo messages on inputs that are not its own, and therefore becomes able to output $\perp$.
2. Suppose instead that some $v^* = b_1^* \| \ldots \| b_\ell^* \in \{0, 1\}^\ell$ is the input of $t + 1$ honest parties. By the time $T + \Delta$, all the honest parties with inputs other than v^* receive $t + 1$ echoes on v^* and thus echo $\perp$. Then, by the time $T + 2\Delta$, for each $k \in \{1, \ldots, \ell\}$ each honest party P_i receives $n - t \geq 2t + 1$ echoes that are on $\perp$ or v^* (which has the k^{th} bit b_i^*), and thus adds b_i^* to W_i^k. So, by the time $T + 2\Delta$, either every honest party P_i observes that $W_i^k = \{b_k^*\}$ for all k, or some honest party P_i obtains $W_i^k = \{0, 1\}$. In the former case, the honest parties all propose v^*, which by the time $T + 3\Delta$ leads to them all receiving $n - t$ proposals on v^* and thus becoming able to output $(v^*, 1)$ or $(\perp, 0)$. In the latter case, the honest party P_i with $W_i^k = \{0, 1\}$ has by the time $T + 2\Delta$ for both bits $b \in \{0, 1\}$ received echo messages on $\perp$ or on strings with the k^{th} bit b from $n - t$ parties ($t + 1$ or more of which are honest). Hence, by the time $T + 3\Delta$, for both bits $b \in \{0, 1\}$ every honest party P_j receives from $t + 1$ honest parties echoes on $\perp$ or on strings with the k^{th} bit b, which means that P_j obtains $V_j^k = \{0, 1\}$ and becomes able to output $(\perp, 0)$. ◀

Complexity. The round complexity is 3, as proven above. The message complexity is $\mathcal{O}(n^2)$ as each party multicasts at most 3 messages (an echo on $\perp$, an echo on its input and a proposal), and the communication complexity is $\mathcal{O}(\ell n^2)$ bits. Note that the communication is balanced evenly between the parties: Each party sends $\mathcal{O}(n)$ messages and $\mathcal{O}(\ell n)$ bits to the others.

A.2 Grade Doubling

We achieve grade-doubling with a *proposal* protocol we call Prop. In proposal, each party P_i acquires an input $v_i \in \mathcal{M}$ in an input domain $\mathcal{M}$, and outputs a set $Y_i \subseteq M$ of size 1 or 2. To work properly, a proposal protocol requires there to exist a set S of size 2 (which the parties might not know in advance) such that the honest parties only acquire inputs in S. If there exists such a set S, a proposal protocol achieves liveness with the following safety properties:

- ***agreement:*** If Y_i and Y_j are honest output sets, then $Y_i \cap Y_j \neq \varnothing$.
- ***validity:*** For every honest output set Y_i, every $y \in Y_i$ is an honest party's input.

With binary inputs, proposal is equivalent to binary 1-graded consensus. Suppose the parties run a proposal protocol with inputs in $\{0,1\}$. If they run it with a common input bit b^* then they all output $\{b^*\}$ from it, and otherwise for some bit b^* they each output $\{b^*\}$ or $\{0,1\}$ from it. These output guarantees are exactly those of binary 1-graded consensus when we map the proposal outputs $\{0\},\{0,1\},\{1\}$ to the 1-graded consensus outputs $(0,1),(\perp,0),(1,1)$. However, when there are more than two possible inputs, a binary 1-graded consensus protocol cannot be used, whereas a proposal protocol still works with output guarantees comparable to those of 1-graded consensus as long as the honest parties happen to run it with at most two inputs a and b, even if the parties with the input a do not know what b is and vice versa.

Our 3-round proposal protocol Prop is based on the 4-round protocol AProp in [12], which is in turn based on the protocol $\Pi^{t_s}_{\mathsf{prop}}$ in [7]. Again, we shave off one round. While according to AProp's design a party P_i would need to receive $2t+1$ echoes on a value v to add v to its set V_i, below in Prop the party P_i adds v to V_i after receiving $t+1$ echoes on v. This change makes it easier for P_i to add values to V_i, obtain $|V_i| = 2$ and output V_i.

Note that using proposal to double grades is not a new invention. The BinAA protocol in [5] essentially uses proposal (therein called weak binary value broadcast) for this purpose.

Protocol Prop

Code for a party P_i with the input v_i

1: multicast $\langle \texttt{ECHO}, v_i \rangle$ if you haven't done so before
2: **for** any v, when you have received echoes on v from $t+1$ parties **do**
3: multicast $\langle \texttt{ECHO}, v \rangle$ if you haven't done so before
4: $V_i \leftarrow V_i \cup \{v\}$
5: **if** $|V_i| = 2$ **then**
6: output V_i if you haven't output before, and keep running
7: **when** there is a first v such that you've received echoes on v from $2t+1$ parties **do**
8: multicast $\langle \texttt{PROP}, v \rangle$
9: **upon** receiving proposals on any v from $n-t$ parties **do**
10: output $\{v\}$ if you haven't output before, and keep running

▶ **Theorem 9.** Prop *is a secure 3-round proposal protocol with liveness.*

Proof. Below, we assume that there exists a set $S = \{a, b\}$ such that the honest parties only acquire inputs in S. Without this, the agreement and liveness properties would be lost.

If an honest party echoes a value v, then v must be some honest party's input. Otherwise, the first honest party that echoes v would have to have received $t+1$ echoes on v, at least one being from an honest party who contradictorily echoed v earlier. For any value v that is not any honest party's input, the fact that no honest party echoes v means that the honest parties cannot output sets that contain v (i.e., that we have validity), due to the following:

1. An honest party P_i cannot add v to its set V_i, as P_i would need to receive $t+1$ echoes on v to do this. Therefore, if P_i outputs V_i when $|V_i| = 2$, then $v \notin V_i$.
2. An honest party P_i cannot propose v, as P_i would need to receive $2t+1$ echoes on v to do this. This means that an honest party P_j cannot output $\{v\}$, as P_j would have to receive $n - t \geq t + 1$ proposals on v to do this.

As for agreement, observe that by the validity property the honest parties can only output non-empty subsets of $S = \{a, b\}$, as their inputs are all in S. So, we only need to show that if some honest parties P and P' respectively output $\{v\}$ and $\{v'\}$ for some $v, v' \in S$, then $v = v'$. Again, this follows from a traditional quorum intersection argument. If we have such parties P and P', then P has received $n-t$ proposals on v, while P' has received $n-t$ proposals on v'. So, there are $n - 2t \geq t + 1$ parties, or at least one honest party, who have sent P a proposal on v and P' a proposal on v'. This implies $v = v'$ as an honest party can only propose once.

It remains to show liveness in 3 rounds. As we did for GC_1, let us suppose that by some time T every honest party has acquired an input in $\{a, b\} = S$ and thus echoed it. Let us show that every honest party outputs by the time $T + 3\Delta$, where Δ is the maximum honest message delay that the adversary causes. Since there exists some $v \in S$ that is at least $\lceil \frac{n-t}{2} \rceil \geq t + 1$ honest parties' input, we know that by the time $T + \Delta$ every honest party receives $t + 1$ echoes on v and thus echoes v even if v is not its input. Then, by the time $T + 2\Delta$, every honest party receives $n - t \geq 2t + 1$ echoes on v. Finally, this is followed by one of the two cases below, in both of which the honest parties all output by the time $T + 3\Delta$.

1. It could be the case that no honest party proposes any $v' \neq v$ by the time $T + 2\Delta$. Then, by the time $T + 2\Delta$ every honest party proposes v after receiving $2t + 1$ echoes on v. Consequently, by the time $T + 3\Delta$, every honest party receives $n - t$ proposals on v, which allows the party to output v if it did not output earlier.
2. It could be the case that some honest party P_i proposes some $v' \neq v$ by the time $T + 2\Delta$. Then, at the time $T + 2\Delta$ the party P_i must have received $2t + 1$ echoes on v', with at least $t + 1$ of these echoes being from honest parties. Since both v and v' get echoed by at least $t + 1$ honest parties by the time $T + 2\Delta$, we conclude that by the time $T + 3\Delta$ every honest party P_j gains the ability to output $V_j = \{v, v'\} = S$ after it receives $t + 1$ echoes on both v and v' and thus adds both v and v' to V_j. ◀

Complexity. The round complexity is 3, as proven above. The message complexity is $\mathcal{O}(n^2)$ as each party multicasts at most 3 messages (two echoes and one proposal), with each honest party echo/proposal carrying an honest party's input. Again, the communication is balanced.

With Prop, it is simple to grade-double $\mathsf{GC}_{2^{k-1}}$ into GC_{2^k}. All we need to do is sequentially compose $\mathsf{GC}_{2^{k-1}}$ and Prop, and interpret Prop outputs as GC_{2^k} outputs.

Protocol GC_{2^k} when $k \geq 1$

Code for a party P_i with the input v_i

1: join a common instance of $\mathsf{GC}_{2^{k-1}}$ with the input v_i
2: join a common instance of Prop
3: **upon** outputting (y, g) from $\mathsf{GC}_{2^{k-1}}$ **do**
4: let your Prop input be (y, g)
5: **upon** outputting Y from Prop **do**
6: if $Y = \{(y, j)\}$ for some y and some $j \geq 0$, then output $(y, 2j)$
7: if $Y = \{(y, j), (y', j+1)\}$ for some y, y' and some $j \geq 0$, then output $(y', 2j+1)$

▶ **Theorem 10.** *For all $k \geq 0$, GC_{2^k} is a secure 2^k-graded consensus protocol with liveness.*

Proof. Naturally, this theorem follows by induction on k. The base case $k = 0$ is the protocol GC_1, which we have already proven secure. Below, we consider $k \geq 1$, where we obtain GC_{2^k} by sequentially composing $\mathsf{GC}_{2^{k-1}}$ and Prop.

The honest parties begin by running $\mathsf{GC}_{2^{k-1}}$ with their inputs. So, for some honest input v^* and some grade $g \in \{0, \ldots, 2^{k-1} - 1\}$ they each output either (v', g) or $(v^*, g+1)$ from $\mathsf{GC}_{2^{k-1}}$, where $v' = \perp$ if $g = 0$ and $v' = v^*$ if $g \geq 1$. Afterwards, they run Prop with inputs in $\{(v', g), (v^*, g+1)\}$, and therefore either they all output $\{(v', g)\}$ or $\{(v', g), (v^*, g+1)\}$ from Prop and thus all output $(v', 2g)$ or $(v^*, 2g{+}1)$ from GC_{2^k}, or they all output $\{(v', g), (v^*, g{+}1)\}$ or $\{(v^*, g+1)\}$ from Prop and thus all output $(v^*, 2g+1)$ or $(v^*, 2g+2)$ from GC_{2^k}. Either way, GC_{2^k} achieves liveness, intrusion tolerance and agreement.

If the honest parties run GC_{2^k} with a common input v^*, then they all run $\mathsf{GC}_{2^{k-1}}$ with the input v^*, output $(v^*, 2^{k-1})$ from it, run Prop with the input $(v^*, 2^{k-1})$, output $\{(v^*, 2^{k-1})\}$ from it, and finally output $(v^*, 2^k)$ from GC_{2^k}. Hence, GC_{2^k} achieves validity as well. ◀

Complexity. As GC_{2^k} consists of one 3-round GC_1 instance followed by k sequential 3-round Prop iterations, GC_{2^k}'s round complexity is $3k{+}3$ and its message complexity is $\mathcal{O}(kn^2)$. When the parties run GC_{2^k} with ℓ-bit inputs, they run GC_1 with inputs in $\{0,1\}^\ell$, and so GC_1 costs $\mathcal{O}(\ell n^2)$ bits of communication. Then, for each $j \in \{1, \ldots, k\}$ they run the j^{th} Prop iteration with inputs that are 2^{j-1}-graded consensus outputs in $(\{0,1\}^\ell \times \{1, \ldots, 2^{j-1}\}) \cup \{(\perp, 0)\}$, of size $\mathcal{O}(\ell + \log j) = \mathcal{O}(\ell + \log k)$, and so the j^{th} Prop iteration costs $\mathcal{O}(n^2(\ell + \log k))$ bits of communication. Overall, we see that GC_{2^k}'s communication complexity is $\mathcal{O}(kn^2(\ell + \log k))$ bits, with the $\log k$ term here also covering the Prop iterations' ID tags. Note that the communication in GC_{2^k} is balanced, as both GC_1 and Prop have balanced communication.

B The Termination Protocol Term

Protocol Term

Code for a party P_i

1: $y_i \leftarrow \perp$ ▷ *Here, $\perp$ is a placeholder value than cannot be a valid* Term *input.*
2: **for** any v, if you acquire the input v or receive $\langle \texttt{ECHO}, v \rangle$ from $t+1$ parties **do**
3: if $y_i = \perp$, then $y_i \leftarrow v$
4: multicast $\langle \texttt{ECHO}, v \rangle$
5: **when** you have received `READY` from $t+1$ parties or there exists some v such that you have received $\langle \texttt{ECHO}, v \rangle$ from $2t+1$ parties **do**
6: multicast `READY`
7: **when** you have received `READY` from $2t+1$ parties, you have multicast `READY` and you have set $y_i \neq \perp$ **do**
8: output y_i and terminate

▶ **Theorem 6.** *In any* Term *execution where the maximum honest message delay is Δ and the honest parties only acquire inputs in some set $\{y, y'\}$, if by some time T either every honest party acquires an input in $\{y, y'\}$ or some honest party terminates* Term*, then by the time $T + 3\Delta$ every honest party terminates* Term *with an output that is an honest party's input.*

Proof. Suppose an honest party P_i outputs some v from Term. Then, P_i has received $t+1$ echoes on v, at least one of which is honest. An honest party only echoes a value if the value is its input or it receives $t+1$ echoes on the value (at least one of which is honest), which means that the first honest party who echoed v did so because its input is v. So, we see that the honest parties can only output honest party inputs from Term.

As for termination in 3 rounds, let Δ be the maximum honest message delay the adversary causes. We first prove a fact we will use later: If every honest party multicasts `READY` by some time T, then every honest party terminates by the time $T+\Delta$. To see why, suppose every honest party multicasts `READY` by some time T. At the time T, the first honest party who has multicast `READY` has done so because it has received $2t+1$ echoes on some v, at least $t+1$ of which are honest. So, by the time $T+\Delta$ every honest party P_i receives $t+1$ honest echoes on v, which allows P_i to set $y_i \leftarrow v$ if $y_i = \bot$. Moreover, since every honest party multicasts `READY` by the time T, every honest party P_i receives $n-t \geq 2t+1$ honest `READY` messages by the time $T+\Delta$, which allows P_i to terminate Term by the time $T+\Delta$ with an output $y_i \neq \bot$.

Now, let us show that if an honest party P_i terminates by some time T, then every honest party terminates by the the time $T+2\Delta$. This is because if P_i terminates by the time T, it does so after receiving `READY` from $2t+1$ parties, at least $t+1$ of which are honest. Since at least $t+1$ honest parties multicast `READY` by the time T, every honest party receives $t+1$ honest `READY` messages and therefore multicasts `READY` by the time $T+\Delta$, and consequently every honest party terminates by the time $T+2\Delta$.

Finally, suppose that by some time T every honest party acquires an input. We want to show that every honest party terminates by the time $T+3\Delta$. If some honest party terminates by the time $T+\Delta$, then this follows from what we have proven above; so, suppose no honest party terminates by the time $T+\Delta$. By the time T, every honest party echoes its input in $\{y, y'\}$, and so there exists some $y^* \in \{y, y'\}$ that at least $\lceil \frac{n-t}{2} \rceil \geq t+1$ honest parties echo. By the time $T+\Delta$, every honest party receives $t+1$ echoes on y^* before terminating, and thus echoes y^* even if it has not acquired the input y^*. By the time $T+2\Delta$, every honest party receives $n-t \geq 2t+1$ echoes on y^*, and thus multicasts `READY`. Finally, by the time $T+3\Delta$, every honest party terminates Term. ◀

C Skipped Proofs

▶ **Theorem 3.** *For any positive integer q and any $0 \leq j < q$, suppose the honest parties run Exp_j with inputs in $\{2^j-1, \ldots, 2^q-1\}$. Then, they reach edge agreement in the path $\mathbb{N}$ based on their inputs in at most $6(j+1)$ rounds if they have the common input 2^j-1 and in at most $12q-6j-5$ rounds otherwise.*

Proof. To prove the theorem, we pick any arbitrary $q \geq 1$, and by induction prove for each $j \in \{q-1, q-2, \ldots, 0\}$ (in this order) that Exp_j is a protocol for edge agreement in the path $(2^j-1, \ldots, 2^q-1)$ that takes at most $6(j+1)$ rounds if run with the common input 2^j-1 and at most $12q-6j-5$ rounds otherwise.

Note that for any $j \geq 0$, if no honest party obtains the grade 0 from GC_2 in Exp_j, then no honest party multicasts `CENTER`, and thus no honest party outputs $2^{j+1}-1$ on Line 22 upon receiving `CENTER` from $t+1$ parties. We use this fact in the rest of the proof below.

The induction's base case is $q = j+1 \geq 1$, where each honest party P_i runs Exp_j with an input $v_i \in \{2^j-1, \ldots, 2^{j+1}-1\}$. In this base case, the honest parties all input `LEFT` to GC_2, and thus all output $(\texttt{LEFT}, 2)$ from it. Then, each honest party P_i sets $\mathsf{side} = \texttt{LEFT}$, sets $v_i^{\mathsf{next}} = \min(v_i, 2^{j+1}-1) = v_i$, runs $\mathsf{TC}((2^j-1, \ldots, 2^{j+1}-1))$ with the other parties with the input $v_i^{\mathsf{next}} = v_i$, and finally lets its TC output be its final output. So, Exp_j's security follows

from TC's security, and Exp_j takes at most $6 + 6j + 1 = 12q - 6j - 5$ rounds: 6 rounds due to GC_2 and $6j + 1$ rounds due to TC (since TC is run on a path of length 2^j). However, if the honest parties have a common input, then TC only takes $6j$ rounds, and thus Exp_j takes $6(j + 1)$ rounds instead of $6(j + 1) + 1$.

Now, let us consider the cases that might arise when the honest parties run Exp_j with inputs in $\{2^j - 1, \ldots, 2^q - 1\}$ for any $j \in \{q - 2, q - 3, \ldots, 0\}$, with the inductive assumption that Exp_{j+1} provides edge agreement in $(2^{j+1} - 1, \ldots, 2^q - 1)$ in at most $6(j + 2)$ rounds when run with the common input $2^{j+1} - 1$ and at most $12q - 6(j + 1) - 5$ rounds otherwise.

1. It could be the case that every honest input is less than 2^{j+1}. This case is identical to the base case above: The honest parties reach edge agreement in at most $6(j + 1)$ rounds if they have a common input and in at most $6(j + 1) + 1 < 12q - 6j - 5$ rounds otherwise.
2. It could be the case that every honest input is at least 2^{j+1}. Then, the honest parties all run GC_2 with the input $\texttt{RIGHT}$, and thus all output $(\texttt{RIGHT}, 2)$ from it. Then, each honest party P_i sets $\mathsf{side} = \texttt{RIGHT}$, sets $v_i^{\mathsf{next}} = \max(v_i, 2^{j+1} - 1) = v_i$, runs Exp_{j+1} with the other parties for edge agreement in the path $(2^{j+1} - 1, \ldots, 2^q - 1)$ with the input $v_i^{\mathsf{next}} = v_i$, and finally lets its Exp_{j+1} output be its final output. Hence, Exp_j's security follows from Exp_{j+1}'s security, and Exp_j takes at most $12q - 6j - 5$ rounds: 6 rounds due to the initial GC_2, and $\max(6(j + 2), 12q - 6(j + 1) - 5) = 12q - 6(j + 1) - 5$ rounds due to Exp_{j+1}. The last equality here follows from the fact that $12q \geq 12j + 24$.
3. The most challenging case is when some honest inputs are below 2^{j+1}, while others or not. In this case, the parties run GC_2 with different inputs, and so for some $k \in \{\texttt{LEFT}, \texttt{RIGHT}\}$ they either all output $(k, 2)$ or $(k, 1)$ from GC_2, or all output $(k, 1)$ or $(\perp, 0)$ from GC_2. Below, we say that the corresponding protocol for the graded consensus value $k = \texttt{LEFT}$ is $\mathsf{TC}((2^j - 1, \ldots, 2^{j+1} - 1))$, while the corresponding protocol for $k = \texttt{RIGHT}$ is Exp_{j+1}.
   - **(a)** The case where the parties output $(k, 2)$ or $(k, 1)$ from GC_2 for some $k \in \{\texttt{LEFT}, \texttt{RIGHT}\}$ is like the two cases we considered above where they all output $(k, 2)$ from GC_2, except for that some honest parties run the corresponding protocol with the input $v_i^{\mathsf{next}} = v_i$ while some others run it with the input $v_i^{\mathsf{next}} = 2^{j+1} - 1$. As some honest inputs are below $2^{j+1} - 1$ while others or not, $2^{j+1} - 1$ is in the convex hull of the honest inputs, which means that some honest parties switching their inputs to it does not impact convex validity. So, the honest parties reach edge agreement in at most $12q - 6j - 5$ rounds by running GC_2 in 6 rounds and then running the corresponding protocol in at most $\max(6j + 1, \max(6(j + 2), 12q - 6(j + 1) - 5)) = 12q - 6(j + 1) - 5$ rounds.
   - **(b)** If the honest parties all output $(k, 1)$ or $(k, \perp)$ from GC_2 for some $k \in \{\texttt{LEFT}, \texttt{RIGHT}\}$, then every honest party P_i sets $v_i^{\mathsf{next}} = 2^{j+1} - 1$, which ensures that the honest parties can only run TC or Exp_{j+1} with the input $2^{j+1} - 1$ and thus only output $2^{j+1} - 1$ from them. Hence, an honest party that outputs from Exp_j can only output the valid value $2^{j+1} - 1$, no matter if it outputs on Line 12, 17, 20 or 22. What remains to show is liveness. For this, observe that since $n - t \geq 2t + 1$, there are either $t + 1$ honest parties that output $(\perp, 0)$ from GC_2, or $t + 1$ honest parties that output $(k, 1)$ from GC_2.
     - **(i)** In the former case, the $t + 1$ honest parties that output $(\perp, 0)$ from GC_2 multicast $\texttt{CENTER}$, and so, after one round (i.e. $7 < 12q - 6j - 5$ rounds after Exp_j begins) every honest party becomes able to output $2^{j+1} - 1$ on Line 22.
     - **(ii)** In the latter case, the $t + 1$ honest parties that output $(k, 1)$ from GC_2 multicast $k \in \{\texttt{LEFT}, \texttt{RIGHT}\}$ before running the corresponding protocol, and this allows the honest parties that output $(\perp, 0)$ from GC_2 to also set $\mathsf{side} = k$ and run the corresponding protocol, albeit with one additional round of delay compared to the parties that output $(k, 1)$ from GC_2. Therefore, every honest party that does

not output $2^{j+1}-1$ on Line 22 eventually outputs $2^{j+1}-1$ by outputting this from the corresponding protocol. The corresponding protocol (which the honest parties run with the common input $2^{j+1}-1$) is either TC, which takes at most $6j$ rounds, or Exp_{j+1}, which takes at most $6(j+2)$ rounds. So, every honest party outputs in at most $6+1+6(j+2) = 12(j+2)-6j-5 \leq 12q-6j-5$ rounds. Here, the first 6 rounds are due to GC_2, the following 7^{th} round is due to the honest parties that output $(\perp, 0)$ from GC_2 needing one extra round to learn k, and the final $6(j+2) \leq 12q-6j-12$ rounds are due to the corresponding protocol. ◀

▶ **Theorem 4.** *Suppose* $\mathsf{Agr}_{\mathbb{N}}$ *is a live protocol for edge agreement in* $\mathbb{N}$ *which for some function* f *takes at most* $f(M)$ *rounds when the honest parties run it with inputs in* $\{0,\ldots,M\}$*. Then,* $\mathsf{2\text{-}Step}(\mathsf{Agr}_{\mathbb{N}})$ *is a live protocol for edge agreement in* $\mathbb{N}$ *that takes at most* $f(5\lfloor\log_2(M+1)\rfloor)+6\lfloor\log_2(M+1)\rfloor+1$ *rounds when the honest parties run it with inputs in* $\{0,\ldots,M\}$*.*

Proof. Let $V = \{\lfloor\log_2(v_i+1)\rfloor : P_i \text{ is honest}\}$, let $q_1 = \min V$, and let $q_2 = \max V$. Observe that $v_i \in \{2^{q_1}-1,\ldots,2^{q_2+1}-1\}$ for every honest party P_i and that $q_2 = \lfloor\log_2(M+1)\rfloor$ if the maximum honest input is M. The honest parties run $\mathsf{Agr}_{\mathbb{N}}$ with inputs in $\{5q_1,\ldots,5q_2\}$, and so they output adjacent integers in $\{5q_1,\ldots,5q_2\}$ from it, in at most $f(5q_2)$ rounds.

If $q_1 = q_2$, then the honest parties all run $\mathsf{Agr}_{\mathbb{N}}$ with the common input $5q_2$, and thus they all output $5q_2$ from it. Consequently, each honest party P_i sets $(k_i, r_i) = (q_2, 0)$ and obtains its final output from a common instance of $\mathsf{TC}((2^{q_2}-1,\ldots,2^{q_2+1}-1))$ which it runs with the input $v_i^{\mathsf{next}} = \min(\max(v_i, 2^{q_2}-1), 2^{q_2+1}-1) = v_i \in \{2^{q_2}-1,\ldots,2^{q_2+1}-1\}$. Edge agreement thus follows from TC, in $6q_2+1$ more rounds.

For the rest of the proof, we assume $q_1 < q_2$. Observe that the boundary values q_1 and q_2 arise from some honest inputs v_i, v_j being such that $\lfloor\log_2(v_i+1)\rfloor = q_1$ and $\lfloor\log_2(v_j+1)\rfloor = q_2$. These equalities respectively imply $v_i < 2^{q_1+1}-1$ and $v_j \geq 2^{q_2}-1$, which means that every integer in $\{2^{q_1+1}-1,\ldots,2^{q_2}-1\}$ is in the convex hull of the honest parties' inputs.

When an honest party P_i outputs $5k_i+r_i$ from $\mathsf{Agr}_{\mathbb{N}}$, it lets $v_i^{\mathsf{next}} = \min(\max(v_i, 2^{k_i}-1), 2^{k_i+1}-1)$ if $r_i = 0$, and lets $v_i^{\mathsf{next}} = 2^{k_i+1}-1$ otherwise. Let us show that in every possible case P_i chooses a valid v_i^{next}; that is, P_i sets v_i^{next} to a value which is inside the convex hull of the honest inputs. If $r_i = 0$, then $q_1 \leq k_i \leq q_2$, and there are three cases we must consider: If $v_i < 2^{k_i}-1$ then $v_i^{\mathsf{next}} = 2^{k_i}-1$ is valid since $2^{k_i}-1$ is between the valid values v_i and $2^{q_2}-1$, if $v_i > 2^{k_i+1}-1$ then $v_i^{\mathsf{next}} = 2^{k_i+1}-1$ is valid since $2^{k_i+1}-1$ is between the valid values $2^{q_1+1}-1$ and v_i, and finally if $2^{k_i}-1 \leq v_i \leq 2^{k_i+1}-1$ then $v_i^{\mathsf{next}} = v_i$ is valid as well. Meanwhile, if $r_i > 0$, then by $5q_1 \leq 5k_i+r_i \leq 5q_2$ we have $q_1 \leq k_i \leq q_2-1$, which makes $v_i^{\mathsf{next}} = 2^{k_i+1}-1 \in \{2^{q_1+1}-1,\ldots,2^{q_2}-1\}$ valid.

Convex validity follows from the fact that every honest party P_i either directly outputs the valid value $2^{k_i+1}-1$ or obtains its output from a TC instance for which the honest parties can only acquire valid inputs. Note that the honest parties do not run TC with out-of-range inputs: The value $2^{k_i+1}-1$ which a party P_i inputs to $\mathsf{TC}((2^{k_i}-1,\ldots,2^{k_i+1}-1))$ if $r_i \in \{1,2\}$ or $\mathsf{TC}((2^{k_i+1}-1,\ldots,2^{k_i+2}-1))$ if $r_i \in \{3,4\}$ is in range for both $\mathsf{TC}((2^{k_i}-1,\ldots,2^{k_i+1}-1))$ and $\mathsf{TC}((2^{k_i+1}-1,\ldots,2^{k_i+1}-1))$, and the value $\min(\max(v_i, 2^{k_i}-1), 2^{k_i+1}-1)$ which a party P_i inputs to $\mathsf{TC}((2^{k_i}-1,\ldots,2^{k_i+1}-1))$ if $r_i = 0$ is in-range as well.

The initial $\mathsf{Agr}_{\mathbb{N}}$ guarantees that there exists some $v = 5k+r$ where $q_1 \leq k \leq q_2-1$ and $0 \leq r \leq 4$ such that every honest party outputs either v or $v+1$ from it. Based on this, let us prove the edge agreement and liveness properties of $\mathsf{2\text{-}Step}(\mathsf{Agr}_{\mathbb{N}})$ via case analysis on r.

1. If $r = 0$, then the honest parties all run $\mathsf{TC}((2^k-1,\ldots,2^{k+1}-1))$ with in-range valid inputs, and obtain their final outputs from it. Edge agreement in $\mathbb{N}$ thus follows from $\mathsf{TC}((2^k-1,\ldots,2^{k+1}-1))$, in $6k+1$ more rounds.

2. If $r = 1$, then the honest parties all run $\mathsf{TC}((2^k - 1, \ldots, 2^{k+1} - 1))$ with the common input $2^{k+1} - 1$, and thus all output $2^{k+1} - 1$ from it in $6k$ rounds. The honest parties P_i that have $r_i = 1$ output $2^{k+1} - 1$ from $\mathsf{2\text{-}Step}(\mathsf{Agr_N})$ after they output $2^{k+1} - 1$ from TC, while those that have $r_i = 2$ output $2^{k+1} - 1$ directly after they output from $\mathsf{Agr_N}$.
3. If $r = 2$, then every honest party P_i has $r_i \in \{2, 3\}$, which leads to it directly outputting $2^{k+1} - 1$ after it outputs from $\mathsf{Agr_N}$. The fact that the honest parties P_i with $r_i = 2$ run $\mathsf{TC}((2^k - 1, \ldots, 2^{k+1} - 1))$ while those with $r_i = 3$ run $\mathsf{TC}((2^{k+1} - 1, \ldots, 2^{k+2} - 1))$ is not an issue as none of the honest parties care about their TC outputs.
4. If $r = 3$, then the honest parties run $\mathsf{TC}((2^{k+1} - 1, \ldots, 2^{k+2} - 1))$ with the common input $2^{k+1} - 1$, and thus all output $2^{k+1} - 1$ from it in $6(k + 1)$ rounds. The honest parties P_i that have $r_i = 4$ output $2^{k+1} - 1$ from $\mathsf{2\text{-}Step}(\mathsf{Agr_N})$ after they output $2^{k+1} - 1$ from TC, while those that have $r_i = 3$ output $2^{k+1} - 1$ directly after they output from $\mathsf{Agr_N}$.
5. If $r = 4$, then every honest party P_i (both if $(k_i, r_i) = (k, r)$ and if $(k_i, r_i) = (k + 1, 0)$) runs $\mathsf{TC}((2^{k+1} - 1, \ldots, 2^{k+2} - 1))$ with an in-range valid input, and obtains its final output from it. Edge agreement thus follows from $\mathsf{TC}((2^{k+1} - 1, \ldots, 2^{k+2} - 1))$, in $6(k + 1) + 1$ more rounds.

Note that in all the cases above, $\mathsf{2\text{-}Step}(\mathsf{Agr_N})$ takes at most $6q_2 + 1$ rounds after $\mathsf{Agr_N}$. Hence, $\mathsf{2\text{-}Step}(\mathsf{Agr_N})$ takes at most $f(5q_2) + 6q_2 + 1$ rounds. This is the round complexity stated in the theorem since $q_2 = \lceil \log_2(M + 1) \rceil$ by definition. ◀

▶ **Theorem 5.** *If $\mathsf{Agr_N}$ is a live protocol for edge agreement in $\mathbb{N}$, then $\mathsf{Agr_Z}(\mathsf{Agr_N})$ is a live protocol for edge agreement in $\mathbb{Z}$.*

Proof. If the honest parties all have inputs in $\mathbb{N}$, then every honest party P_i runs GC_2 with the input 1, outputs $(1, 2)$ from GC_2, lets $v_i^{\mathsf{next}} = \max(0, 1 \cdot v_i) = v_i$, runs $\mathsf{Agr_N}$ with the input $v_i^{\mathsf{next}} = v_i$, and finally outputs $1 \cdot y = y$ from $\mathsf{Agr_Z}(\mathsf{Agr_N})$ when it outputs y from $\mathsf{Agr_N}$. Thus, $\mathsf{Agr_Z}(\mathsf{Agr_N})$' security follows from $\mathsf{Agr_N}$'s security.

Similarly, if the honest parties all have negative inputs, then every honest party P_i runs GC_2 with the input -1, outputs $(-1, 2)$ from GC_2, lets $v_i^{\mathsf{next}} = \max(0, -1 \cdot v_i) = -v_i$, runs $\mathsf{Agr_N}$ with the input $v_i^{\mathsf{next}} = -v_i$, and finally outputs $-1 \cdot y = -y$ from $\mathsf{Agr_Z}(\mathsf{Agr_N})$ when it outputs y from $\mathsf{Agr_N}$. Again, $\mathsf{Agr_Z}(\mathsf{Agr_N})$'s security follows from $\mathsf{Agr_N}$'s security. This is because by symmetry $\mathsf{Agr_N}$ behaves as an edge agreement protocol for $(\ldots, -1, 0)$ when the parties mirror (sign-flip) their inputs for it and outputs from it.

Finally, there is the case where some honest parties have negative inputs while others do not. In this case, the honest parties run GC_2 with different inputs, and so for some $k \in \{1, -1\}$ they either all output $(k, 2)$ or $(k, 1)$ from GC_2, or all output $(k, 1)$ or $(\perp, 0)$ from GC_2. The case where they all output $(k, 2)$ or $(k, 1)$ from GC_2 is similar to the cases above, though with some honest parties P_i running $\mathsf{Agr_N}$ with the input $v_i^{\mathsf{next}} = 0$ rather than $v_i^{\mathsf{next}} = kv_i$. As 0 is in the convex hull of the honest inputs, this does not impact validity, and the honest parties safely reach edge agreement via $\mathsf{Agr_N}$, with input/output mirroring if $k = -1$ and without mirroring if $k = 1$. Meanwhile, if the honest parties all output $(k, 1)$ or $(\perp, 0)$ from GC_2, then they all run $\mathsf{Agr_N}$ with the input 0. So, the honest parties with the GC_2 output $(k, 1)$ output $k \cdot 0 = 0$ from $\mathsf{Agr_Z}(\mathsf{Agr_N})$ after outputting 0 from $\mathsf{Agr_N}$, while those with the GC_2 output $(\perp, 0)$ output 0 from $\mathsf{Agr_Z}(\mathsf{Agr_N})$ directly without caring about their $\mathsf{Agr_N}$ outputs. ◀