Delphi: Efficient Asynchronous Approximate Agreement for Distributed Oracles

Akhil Bandarupalli*, Adithya Bhat†, Saurabh Bagchi*, Aniket Kate*‡, Chen-Da Liu-Zhang§¶, Michael K. Reiter**
*Purdue University {abandaru, sbagchi, aniket}@purdue.edu
†Visa Research haxolotl.research@gmail.com
§Lucerne University of Applied Sciences and Arts chen-da.liuzhang@hslu.ch
¶Duke University michael.reiter@duke.edu
‡Supra Research, §Web3 Foundation, **Chainlink Labs

Abstract—Agreement protocols are crucial in various emerging applications, spanning from distributed (blockchains) oracles to fault-tolerant cyber-physical systems. In scenarios where sensor/oracle nodes measure a common source, maintaining output within the convex range of correct inputs, known as convex validity, is imperative. Present asynchronous convex agreement protocols employ either randomization, incurring substantial computation overhead, or approximate agreement techniques, leading to high $O(n^3)$ communication for an $n$-node system.

This paper introduces Delphi, a deterministic protocol with $O(n^2)$ communication and minimal computation overhead. Delphi assumes that honest inputs are bounded, except with negligible probability, and integrates agreement primitives from literature with a novel weighted averaging technique. Experimental results highlight Delphi’s superior performance, showcasing a significantly lower latency compared to state-of-the-art protocols. Specifically, for an $n = 160$-node system, Delphi achieves an 8x and 3x improvement in latency within CPS and AWS environments, respectively.

I. INTRODUCTION

In the distributed oracle problem setting, a set of $n$ nodes each with an input coming from its sensor device, aim to closely agree on a common output while tolerating a minority of Byzantine faults. This problem is getting increased interest recently as it captures many scenarios including fault-tolerant distributed cyber-physical systems (CPS) [28], [38], [39], [44], [52], oracle networks [16], [20], or agreement on physical sensor inputs such as the average global temperature. Such systems are typically deployed in compute-starved environments supported by paltry infrastructure and asynchronous networks with unpredictable delays. In this context, to ensure that it accurately represents the real-world variable, the output of an ideal system is close to the range of honest inputs. The convex agreement problem [21] captures this requirement by ensuring that the output is in the convex hull (or min-max range) of honest inputs.

Current solutions to the convex agreement problem can be categorized into two different classes. In the first, protocols that use randomization using common coins [9]. Generally, such coins are computationally expensive and require complex cryptographic setup assumptions. As an example, the most efficient and popular implementation of a common coin [14] requires $O(n)$ bilinear pairing computations per coin [13], [18]. Each pairing requires 1000 times more time and energy than symmetric key primitives used in TLS. Furthermore, it also requires a threshold setup, which involves running an expensive Asynchronous Distributed Key Generation (ADKG) procedure amongst nodes [23], [36].

In the second, protocols use asynchronous approximate agreement (AAA) [24]. These AAA protocols are deterministic and, therefore, avoid the expensive computational cost of generating common coins. The state-of-the-art approximate agreement protocols [1] incur high communication costs of $O(n^2)$ bits per round and take $O(\log(\frac{1}{\epsilon}))$ rounds to achieve outputs that are $\epsilon$-close to each other when the range of honest inputs is $\delta$.

Towards addressing these inefficiencies, we make a key observation regarding input from honest parties for the oracle problems. In the CPS settings, the ambient noise causing the difference between inputs follows distributions like Normal and Lognormal [48], [49]. Even prices of stocks and cryptocurrencies are often modeled using distributions like Pareto and Lognormal, where samples are mostly concentrated within a small distance [20]. As a result, given that honest parties often measure inputs that are close to each other, we assume that honest inputs of size $\ell$ are sampled from a thin-tail distribution.

Building on the above observations and the corresponding assumption, we present a protocol that is efficient in both computation and communication. We introduce DELPHI, a signature-free deterministic protocol with $O((n^2 \log(\lambda \log n))$ communication, where $\lambda$ is a statistical security parameter. DELPHI’s outputs are $\epsilon$-close and are at most within a distance $\delta = O(\epsilon)$ from the convex range of honest inputs. See Table I for a comparison to prior works.

A. Solution Overview

Our starting point is a protocol BINAA that achieves Approximate Agreement with binary inputs, i.e. either 0 or 1. In this case, we observe that the parties output values within $\epsilon$ distance with $O(n^2 \log(\frac{1}{\epsilon}))$ communication.

Next, we divide the real input space into intervals of size $\rho$. Then, for each interval, each honest node $i$ participates in BINAA with input 1 if its input $v_i$ is within $\rho$-distance to the interval, or 0 otherwise. Note that if all honest inputs
For Lognormal and Pareto distributions, nodes output their corresponding multi-level weighted average scheme and design levels and $\rho$ so that the averages are close when the honest inputs are $\rho$ output of $B_{IN}$ (e.g. the midpoint value of the interval), and the in two testbeds: (a) A geo-distributed testbed on AWS for the best explained using loggamma and gamma distributions in the $ELPHI$ data over a prolonged period and configure $D$ analyze the nodes’ inputs in each application by collecting of surveillance Drones detecting and locating unauthorized $l$ values do $\rho$-close, ensuring that the weighted averages are also close.

This technique fails when honest nodes’ inputs are further than $\rho$ distance, in which case all weights are 0. We address this by running our technique over multiple levels, each with increasing values of $\rho_i$. The starting level $l = 0$ has $\rho_0 = \epsilon$, and higher levels $l$ have $\rho_l = 2^l \rho_0$. We then introduce a novel multi-level weighted average scheme and design levels and their corresponding $\rho_l$ values based on the input distribution so that the averages are close when the honest inputs are close. Finally, we ensure that levels with high $\rho_l$ values do not contribute to the average, while ensuring that our protocol incurs $O(n^2)$ bits of communication per round.

Data Analysis and Performance Evaluation. We evaluate $DELPHI$ in two applications: (a) A network of oracle nodes reporting the price of Bitcoin, and (b) A distributed system of surveillance Drones detecting and locating unauthorized vehicles in an area using an object detection program. We analyze the nodes’ inputs in each application by collecting data over a prolonged period and configure $DELPHI$ based on this analysis. We indeed observe that measurements can be best explained using loggamma and gamma distributions in the oracle network and object detection settings, both of which are indeed thin-tailed distributions. We also conduct experiments in two testbeds: (a) A geo-distributed testbed on AWS for the oracle network application and (b) A distributed CPS testbed comprised of Raspberry Pi devices for Drone-based object detection. For $n = 160$, $DELPHI$ takes $\frac{1}{3}$rd and $\frac{1}{4}$th the time taken by $FIN$ [27], the State-of-the-art Asynchronous Convex BA protocol, and $\frac{1}{3}$rd and $\frac{1}{4}$th the time taken by Abraham et al. [1], the best AAA protocol, in the AWS and CPS testbed, respectively. We also analyze the practical implications of the Validity relaxation of $DELPHI$.

### II. Preliminaries

#### A. System Model

We assume a system of $n$ nodes $\mathcal{P} := \{1, \ldots, n\}$ connected by pairwise authenticated channels and an asynchronous network. We consider an adaptive adversary $\mathcal{A}$, who can corrupt $t < \frac{n}{2}$ nodes at any time during the execution of the protocol. $\mathcal{A}$ also controls the network between honest nodes and can arbitrarily delay and reorder messages but cannot drop them. We consider a node honest if it is never faulty.

Nodes in the system measure a physical state variable $\tau$ and aim to agree on a value representative of $\tau$. A few examples of such state variables are the price of a cryptocurrency like Bitcoin, the physical location of objects like cars, and the temperature in a given area. Each node measures $\tau$ using an on-board data source, which provides the node with a value $v_i$, an estimate of $\tau$. Such data sources include the price feeds provided by currency exchanges, cameras indicating the position of an object, and sensors sensing temperature. These sources have been known to have a finite accuracy and hence measure $\tau$ with an accuracy error, conventionally modeled using a probability distribution. In line with this error, we assume node $i$’s input $v_i$ is independently sampled from a random variable $X$, with a probability distribution $p(x)$. Nodes and data sources do not know any details about $p$. We model the inputs $v_i$ as floating point numbers with a finite precision. We also assume $v_i \in [s, e]$, where $s$ and $e$ are very small and large finite numbers, respectively, defined at a system level.

#### B. Problem Definition

We define the Approximate Agreement for oracles.
**Definition II.1.** We denote the set of honest nodes’ input values by $V_h$. A protocol $\Pi_{AA}$ for $n$ nodes where node $i$ inputs $v_i$ and outputs $o_i$ solves Approximate Agreement with $\rho$-relaxed validity if it satisfies the following properties.

1) **Termination:** Each honest node must eventually produce an output $o_i$.
2) **$\rho$-relaxed Min-Max Validity:** Each honest node’s output must be within the $\rho$-relaxed interval formed by honest inputs. Let the interval formed by honest inputs be $[m, M]$, where $m = \min(V_h)$ and $M = \max(V_h)$. For every honest node $i \in P$: $m - \rho \leq o_i \leq M + \rho$.
3) **$\epsilon$-agreement:** The outputs of any pair of honest nodes $i$ and $j$ are within $\epsilon$ of each other, i.e., $|o_i - o_j| < \epsilon$. $\epsilon$ is called the agreement distance.

**C. Approximate Agreement for Binary Inputs**

Our starting building block BInAA is an Approximate Agreement protocol with convex validity (i.e., 0-relaxed validity) for binary inputs (every honest node has input 0 or 1), and communicating $O(n^2)$ bits per round, similar to the notion of binary Proxconsensus by Ghinea, Goyal and Liu-Zhang [31]. Our protocol builds upon a weaker variant of the Binary Value broadcast primitive defined in Mostefaoui, Moumen, and Raynal [43]. This primitive can be instantiated with (a straightforward adaptation of the) Crusader Agreement protocol by Abraham, Ben-David, and Yandamuri [2], with $O(n^2)$ bits per round, and terminating in three rounds.

**Definition II.2.** A protocol $\Pi_{BV}$ for $n$ nodes where node $i$ inputs a value $v_i$ and outputs a set of values $B_i$ achieves weak Binary Value broadcast if it satisfies the following properties.

- **Termination:** All honest nodes eventually terminate and output a non-empty set $B_i$, where $|B_i| \geq 1$.
- **Justification:** If the output set $B_i$ of an honest node contains value $v$, then $v$ must have been the input of at least one honest node.
- **Weak Uniformity:** The output sets of any pair of honest nodes $i$ and $j$ must have a non-empty intersection.

$B_i \cap B_j \neq \emptyset$

We describe the BInAA protocol in Algorithm 1 and give a brief intuition of its functioning here. BInAA proceeds in iterations, where in each iteration $r$, each node $i$ participates in an instance of a broadcast protocol $\Pi_{BV,r}$ with an input $z_i$ (in the initial iteration, $z_i = v_i \in \{0, 1\}$). Each iteration achieves the weak BV-broadcast primitive defined in Definition II.2.

We first show that the first iteration of the protocol satisfies all three properties specified in Definition II.2. First, the protocol satisfies Termination because honest inputs are binary, and at least one value $b$ would have been possessed by $t + 1$ honest nodes. This ensures that $b$ will be echoed by enough honest nodes and eventually enable honest nodes to terminate. Second, the protocol satisfies Justification because faulty nodes can produce at most $t$ ECHO1/ECHO2 messages on a value not possessed by honest nodes. Finally, the protocol satisfies weak uniformity because of two main reasons - (a) Each honest node sends at most one ECHO2 message, which implies at most one value $b$ can receive $n - t$ ECHO2s. This ensures that no pair of honest nodes satisfy condition 2 (Line 17) for two different values $b_1 \neq b_2$. (b) A node that terminates this iteration by satisfying the first condition (Line 16) must have at least one value in common with every other honest node. This is because only values 0, 1 can receive $n - t$ ECHO1s.

Second, at the end of each iteration, each node $i$ updates its value as the average of values in the output set $B_i$. Given the set of honest inputs is $\{v_0, v_1\}$, the set of honest output sets from $\Pi_{BV}$ is either $\{\{v_0\}\}$, $\{\{v_0\}\}$, $\{\{v_0, v_1\}\}$, $\{\{v_1\}\}$, $\{\{v_0, v_1\}\}$, or $\{\{v_1\}\}$. The range of honest inputs decreases by at least $\frac{1}{2}$ at each iteration. Further, the updated values in each iteration satisfy the binary input assumption. When run for $\log_\frac{1}{\epsilon}$ iterations, this protocol achieves approximate agreement with

**Algorithm 1** Binary Approximate Agreement (BInAA) protocol

1. **INPUT:** $P$, $b_i \in \{0, 1\}$, $\epsilon$

   ▷ **Parameter Setup**
   // $r_m$ is the number of rounds to run and $r$ is the current round
   2. $r_m \leftarrow \log_2(\frac{2}{\epsilon})$; $r = 1$
   // E1 and E2 stand for list of ECHO1 and ECHO2 messages received. $b_i$ is the output set for weak BV broadcast
   3. $E1_i[r] \leftarrow \emptyset$; $E2_i[r] \leftarrow \emptyset$; $B_i[r] \leftarrow \emptyset$ for $r \in \{1, 2, \ldots, r_m\}$

   ▷ **Begin protocol**
   4. **on** receiving $(\text{Start}, \text{ID}, i, r = 1)$
   5. $b_{i,1} \leftarrow b_i$
   6. $E1_i[r] \leftarrow E1_i[r] \cup \{(b_{i,1}, i)\}$
   7. **SendAll** $(\text{ECHO1}, b_i, r)$ // Send ECHO1 message
   8. **on** receiving $(\text{ECHO1}, b_{j,r}, r)$ from node $j$
   9. $E1_i[r] \leftarrow E1_i[r] \cup \{(b_{j,r}, j)\}$
   10. **on** receiving ECHO1s from $t + 1$ nodes for a value $b$ in current round $r$ i.e. there exists $b$ such that at least $t + 1$ $(b, k) : k \in P$ values are in $E1_i[r]$
   11. **SendAll** $(\text{ECHO1}, b, r)$ // Brachia amplify value $b$
   12. **on** receiving ECHO1s from $n - t$ nodes for a value $b$ and not previously sending an ECHO2 message in current round $r$
   13. $E2_i[r] \leftarrow E2_i[r] \cup \{(b, i)\}$
   14. **SendAll** $(\text{ECHO2}, b, r)$
   15. **wait until** one of the following conditions is true for round $r$:
   16. (1) Receiving ECHO1s from $n - t$ nodes for two values $b_1, b_2$
   i.e. $\exists b_1, b_2$ such that $n - t (b_1, k) : k \in P$ values and $n - t (b_2, k) : k \in P$ values are in the set $E1_i[r]$
   17. (2) Receiving ECHO2s from $n - t$ nodes for a value $b$ i.e.
   $\exists b$ such that $n - t (b, k) : k \in P$ values are in the set $E2_i[r]$
   18. **on** condition (1) being true
   19. $B_i[r] \leftarrow \{b_1, b_2\}$ // End here for Weak BV Broadcast
   20. $b_{i,r+1} \leftarrow \frac{b_1 + b_2}{2}$
   21. $r \leftarrow r + 1$ and **goto** Line 4 to begin new round
   22. **on** condition (2) being true
   23. $B_i[r] \leftarrow \{b\}$ // End here for Weak BV Broadcast
   24. $b_{i,r+1} \leftarrow b$
   25. $r \leftarrow r + 1$ and **goto** Line 4 to begin new round
   26. **output** the value $b_{i,r_m}$ after terminating $r_m$ rounds
restricting equivocation by faulty nodes, which requires using $n$ values. Achieving this at adjacent checkpoints in a level $l$.

- **Separator $p_l$** is the length of each interval or difference between adjacent checkpoints in a level $l$.
- **Checkpoint $\mu_k^l$ or CHECKPOINT$(k, y, l)$** is the $k$th multiple of the separator $p_l$: $\mu_k^l = k p_l$, where $k$ is an integer between $k \in \left[\frac{s}{\rho}, \frac{e}{\rho}\right]$. Level $l$ has checkpoints throughout the space of honest inputs $[s, e]$, where adjacent checkpoints are separated by $p_l = 2^l p_0$.
- **Weight $w_k^l$** is the weight of checkpoint $\mu_k^l$. Nodes achieve Approximate Agreement on these values using BinAA protocol.
- **Level LEVEL$(l, \rho)$ or $\delta[l]$** is the level object. It contains a list of intervals $C_{\rho}^l$ at level $l$ between endpoints $s$ and $e$. $\rho_l = 2^l \rho_0$ is the distance between adjacent checkpoints.

Figure 1: Description of symbols used in DELPHI.

III. DESIGN

In this section, we present the design of our protocol DELPHI. We first provide context, explain the limitations of prior solutions, and then discuss our approach.

A. Limitations of prior protocols

Prior Approximate Agreement protocols like Dolev et al. [24] and Abraham et al. [1] proceed in a round-based manner, where in each round, every node collects a set of values from other nodes, and updates its state using a Trimmed mean of this set. In these protocols, reaching Approximate Agreement requires each node to collect $2t + 1$ values in common with other honest nodes. This is for two major reasons. (a) Even with $t$ faulty values, each honest node must ensure that its updated state is within the range of honest inputs, and (b) Even after trimming $t$ greatest and smallest values in their sets, honest nodes must have enough values in common to reduce their range. Dolev et al. [24] achieve this condition using multicasts with $n = 5t + 1$, where a faulty node can make different honest nodes accept different values. Achieving this at $n = 3t + 1$ resilience requires restricting equivocation by faulty nodes, which requires using the Reliable Broadcast (RBC) primitive. RBC has a lower bound of $O(n^3)$ bits, which results in an overall $O(n^3)$ communication complexity. Therefore, we observe that this requirement of collecting $2t + 1$ values in common with other nodes, which is necessary for achieving strict convex-hull Validity, is the main reason for the $O(n^3)$ communication complexity of all prior Approximate Agreement protocols.

B. Our Approach

We overcome this $O(n^3)$ communication bottleneck by proposing DELPHI, an Approximate Agreement protocol that does not require RBC. DELPHI uses a novel approach based on checkpoints and Binary Approximate Agreement, which achieves communication efficiency by trading off a relaxation in the Validity condition, specified in Definition II.1.

Notation. We divide the space of possible honest inputs between $[s, e]$ among checkpoints separated by a system parameter $\rho$. A checkpoint $\mu_k = k \rho$ represents the space of values between $[k \rho - \frac{\rho}{\rho}, k \rho + \frac{\rho}{\rho}]$, where $k$ is the set of all integers in $(\frac{s}{\rho}, \frac{e}{\rho})$. Nodes run a BinAA protocol instance for all checkpoints in $[s, e]$ to approximately agree on a representative weight $w_k^l$ for each checkpoint. We denote the corresponding pseudocode in Algorithm 2. Each level, indexed by the number $l$, is characterized by the difference between two consecutive checkpoints $\rho_l$. Further, each level comprises checkpoints in $[s, e]$, where each checkpoint is a multiple of $\rho_l$.

For ease of understanding, we first describe a basic, single-level version of DELPHI, explain the challenges, and then explain the multi-level version that addresses these challenges.

1) Single level Approximate Agreement: We first describe a basic version of DELPHI at a single level with distance between checkpoints $\rho$. Nodes run BinAA for all checkpoints in a level. An honest node inputs $v = 1$ to checkpoint $k \rho$ only if $|v_i - k \rho| \leq \rho$, and inputs 0 otherwise (Line 11 in Algorithm 2). Nodes then send out messages for checkpoints and handle incoming messages. Each node then checks whether a round $r$ has terminated for a checkpoint. Upon terminating round $r$ of BinAA of all checkpoints in the level, nodes initiate round $r + 1$. The nodes terminate a BinAA instance after running for $r_{\text{max}}$ rounds (specified later). Upon terminating all BinAA instances, nodes calculate a weighted average of checkpoints $\alpha_i = \frac{\sum_k (k \rho) \times w_k}{\sum_k w_k}$ (Line 18 in Algorithm 2).

Intuition. We observe that when honest inputs are clustered within a distance $\delta \leq \rho$, at least one checkpoint will have weight $w_k = 1$. This is because the Validity property of BinAA ensures that if all honest nodes input the same value to any BinAA instance $B_k$, then all nodes must output that value. This property gives way to two key observations: (a) If all honest inputs $v_i$ are within $\rho$ distance, the weight $w_k = 1$ for at least one checkpoint $\mu_k$, and (b) No checkpoint $\mu_k$ : $|\mu_k - v_i| > \rho \forall i \in V_h$ will have a non-zero weight. The first observation guarantees approximate agreement of the weighted average, whereas the second observation enables us to show that the final output will strictly lie between the $\rho$-relaxed interval of honest inputs $[\min(V_h) - \rho, \max(V_h) + \rho]$.

Challenges. The described approach produces a valid result only when honest nodes’ inputs are within $\delta \leq \rho$ distance. When $\delta > \rho$, the weights of all checkpoints can be zero, resulting in a division by zero. Moreover, estimating the range $\delta$ in a distributed manner also costs a whopping $O(n^3)$ complexity in prior protocols [1]. We explain this challenge pictorially in Fig. 2.

We assume an upper bound $\Delta$ on the range $\delta$, and utilize it to ensure that nodes always output a well-defined weighted average. Prior works like Chakka et al. [20] also utilized this upperbound assumption to achieve BA efficiently, and practically justified this assumption [20]. Moreover, as many applications are modeled using thin-tailed probability distribu-
Figure 2: **DELPHI with one level:** The flags denote the checkpoints, and antennae denote honest inputs. Green checkpoints have weight $w_k = 1$ and drive agreement among nodes, and red checkpoints are outside the range of honest inputs that can have a non-zero weight and contribute to Validity relaxation. When $\delta_2 > \rho$, no green checkpoint exists, which results in agreement failure. A higher $2\rho$ enables nodes to reach agreement with range $\delta_2$, but also adds the weight of farther red checkpoints when the range is $\delta_1$, which affects Validity negatively.

Intuition. We give a pictorial description of this approach in Fig. 3. Since we know that $\delta \leq \Delta$, there must exist a level $\phi \leq l_M$ such that every level $\geq \phi$ must have at least one value $\mu^i_k$ with weight $w^i_k = 1$. In cases with relatively small $\delta$, the level $\phi$ is small. As the checkpoints at higher levels contribute a larger Validity relaxation to the output, we aim to minimize their contribution to the weighted average. We utilize the fact that the maximum weight of a checkpoint in a level $\max w_k^i = 1$ for any level $l \geq \phi$. We set the weight of a level $w^i_l = w_l |w_l - w_{l-1}|$, where $w_l$ is the maximum weight of a checkpoint in level $l$ (Line 21 in Algorithm 2). This operation is analogous to a differentiation. Any checkpoint at level $l > \phi$ will not contribute to the output.

### Algorithm 2 DELPHI protocol

1: **INPUT:** $v_i \in [s, e], \rho_0, \Delta, \epsilon$

    - **Setup**
      // Level and round parameters
      2: $l_M \leftarrow \log_2(\frac{\Delta}{\rho_0}); \epsilon' \leftarrow \frac{\epsilon}{4\Delta l_M}; r_M \leftarrow \log_2(\frac{1}{\epsilon'})$
      3: for $l$ in $\{0, 1, \ldots, l_M\}$ do
        4: Define $\rho_l = 2^l \rho_0$
        5: for $k$ in $\{i \in \mathbb{Z} : \frac{\Delta}{\rho_l} \leq i \leq \frac{2\Delta}{\rho_l}\}$ do
          // Initialize Binary AA instances for all checkpoints
          6: $B^i_k \leftarrow \text{BINAA}(\cdot)$
      
    - **BINAA phase**
      7: on receiving $\langle \text{Start}, ID.i \rangle$
      8: $M \leftarrow \{\}$
      9: for $l$ in $\{0, 1, \ldots, l_M\}$ do
        // Checkpoints separated by $\rho_l$
        10: $\mu^i_{k-1}, \mu^i_k \leftarrow$ Two closest checkpoints to input $v_i$ at lev $l$
        11: **Start** BINAAs instances $B^i_{k-1}$ and $B^i_k$ with input $b^i_{k-1} = 1, b^i_k = 1$ and all other instances with input $b_i = 0$.
      
    - **Bundle** messages of all BINAAs instances together and invoke SendAll on the bundled message
    
    - **Aggregation phase**
      13: on terminating $r_M$ rounds for all BINAAs instances at all levels:
        // 1. Aggregate weights of checkpoints at each level. Each level has a representative value $V_i$ and a weight $w_i$
      14: for $l$ in $\{0, 1, \ldots, l_M\}$ do
        15: if $\exists$ a BINA instance $B^i_k$ with output $> 0$ then
        16: $w^i_k \leftarrow B^i_k.\text{OUTPUT}()$
        17: $\mu^i_k \leftarrow k \rho_l$
        18: $(V_i, w_i) := \left(\frac{\sum |\frac{\Delta}{\rho_l} \leq k \leq \frac{2\Delta}{\rho_l}| w^i_k \mu^i_k}{\sum |\frac{\Delta}{\rho_l} \leq k \leq \frac{2\Delta}{\rho_l}| w^i_k} \right)\max_{1 \leq i \leq l_M} |w^i_k|$
      
      - **else**
        // As weighted average is undefined when all weights are 0, we handle this case by assigning a custom weight for level $l$
      20: $(V_i, w_i) := (v_i, \epsilon')$

    - **2. Cross-level Aggregation:** Aggregate weights across levels
      21: $w^i_l \leftarrow w^i_0$
      22: for $l$ in $\{1, 2, \ldots, l_M\}$ do
      23: $w^i_l \leftarrow w^i_l \times |w_l - w_{l-1}|$
      24: $o_i := \sum_{l=0}^{l_M} w^i_l$
    
    - **output** $o_i, \sum_{l=0}^{l_M} w^i_l$
separately sending messages for each BINA instance would require $(\frac{\rho_0^2}{\rho_1^2})n^2$ messages per round, which is infeasible in practice. We bundle messages from multiple BINA instances and treat them as a single BINA instance. For example, we consider a level $l$ with honest inputs between checkpoints $k_1, k_2$. All honest nodes will input 0 to checkpoints from $[s, k_1]$ and $[k_2, e]$. This implies all these checkpoints require only $O(n^2)$ communication per round. Further, for cases when $k_2 - k_1 = O(\epsilon)$, the total communication per level is $O(n^2)$ bits per round. We also perform cross-level optimization. If all honest inputs are within two consecutive checkpoints $\mu_k^\phi$ and $\mu_{k+1}^\phi$ at level $\phi$, the BINA data for levels $\phi$ to $l_M$ can be represented with constant bits.

IV. Analysis

Overview. We know that nodes approximately agree on the weight of each checkpoint, which implies each node’s weights are $\epsilon'$-close for all checkpoints at all levels. The weighted average function enables nodes to combine the weights and checkpoints by offering two key properties: (a) It forces the final output to be within checkpoints with non-zero weight outputs, and (b) As the weights are $\epsilon'$-close, the outputs from the function are $\epsilon'$-close, which guarantees approximate agreement with Validity. However, this approach also has a caveat: it becomes undefined when the sum of weights is zero. Hence, we first ensure that the denominator in our weighted average is always greater than $1$. Next, we utilize the properties offered by the weighted average scheme to prove approximate agreement and Validity of DELPHI.

A. Termination

We discuss DELPHI’s termination. We show that the weighted average in Line 24 in Algorithm 2 always has a non-zero sum of weights.

**Theorem IV.1.** Termination: Given that the BINA protocol provides Safety, Termination, and Validity, every honest node terminates DELPHI and outputs a finite, defined value.

**Proof.** We know that every honest node terminates all BINA instances corresponding to all checkpoints in levels $l = \{0, \ldots, l_M\}$. Consider the denominator in Line 24 in Algorithm 2. Every honest node starts BINA for all intervals in every level until $l_M$. From the Termination property of BINA, every honest node terminates $r_M$ rounds for all checkpoints until level $l_M$.

\[
S = w_0^2 + \sum_{l=1}^{l_M} w_l \times |w_l - w_{l-1}|
\]

\[
\implies S \geq \frac{1}{2}(2w_0^2 + \sum_{l=1}^{l_M} 2w_l^2 - 2w_l w_{l-1})
\]

\[
\implies S \geq \frac{1}{2}(w_0^2 + w_{l_M}^2 + \sum_{l=1}^{l_M} (w_l - w_{l-1})^2) \geq \frac{w_{l_M}^2}{2}
\]

As honest inputs have a maximum difference $\delta \leq \Delta$, there must exist level $l \leq l_M$ such that $w_l = 1$, which ensures that the sum of weights is $\geq \frac{1}{2}$.

B. Validity

The Validity offered by DELPHI depends on two factors: (i) The separator at level $0\rho_0$, and (ii) The rate of increase of interval length across levels. These factors decide the number of levels in DELPHI and therefore influence the Validity.

**Lemma IV.2.** Given that the BINA protocol satisfies Agreement, Termination, and Validity, an honest node $i$’s representative weight for level $l$ $w_{l,i}'$ in Line 21 in Algorithm 2 is at most $5\epsilon'$ away from $w_{l,j}' \forall l \in \{0, \ldots, l_M\} \forall j \in P$. Further, for all levels $l \geq \lceil \log(\frac{\Delta}{\rho_0}) \rceil$, the weight $w_{l,i}' = 0 \forall i \in P$.

\[
|w_{l,i}' - w_{l,j}'| \leq 5\epsilon', \forall l \in \{0, \ldots, \lceil \log(\frac{\Delta}{\rho_0}) \rceil\} \forall \{i, j\} \in P
\]

\[
w_{l,i}' = 0 \forall l > \lceil \log(\frac{\Delta}{\rho_0}) \rceil, \forall i \in P
\]

**Proof.** The weight $w_{l,i}$ in $w_{l,i}'$ is calculated as $w_{l,i} = \max\{w_{l,k} : B_{l,k}'\}$. From agreement of BINA, we know $|w_{l,i} - w_{l,j}| \leq \epsilon'\forall \{i, j\} \in V_k$, and $|w_{l,i} - w_{l-1,i} - |w_{l,i} - w_{l-1,j}| \leq 2\epsilon'$.

\[
w_{l,i}' - w_{l,j}' \leq (w_{l,j} + \epsilon) \times (|w_{l,j} - w_{l-1,j}| + 2\epsilon - w_{l,j}|w_{l-1,j} - w_{l-1,j}|)
\]

\[
w_{l,i}' - w_{l,j}' \leq \epsilon'(2w_{l,j} + |w_{l,j} - w_{l-1,j}|) + 2\epsilon^2
\]

\[
\implies w_{l,i}' - w_{l,j}' \leq 5\epsilon
\]

For level $\phi = \lceil \log(\frac{\Delta}{\rho_0}) \rceil$, the separation between checkpoints $\rho_1 \geq \delta$, implying $w_l = 1 \forall l \geq \phi, w_l = 0 \forall l \leq \phi$. This implies there exists at least one checkpoint $\mu_k^\phi$ that is within a $\rho_l$ distance from all honest inputs $v_i$. From the Validity of BINA, $\omega_k^\phi = 1\forall i \in P, \implies w_{\phi, i} = 1$. This statement is also true for all levels $> \phi$. Therefore, the weight $w_{l,i}' = 0 \forall l > \phi$. \qed
Theorem IV.3. Given that the BINA protocol satisfies Termination, and Validity, honest nodes’ output values \( \alpha_i \) from DELPHI are always within the following interval; i.e., \( \alpha_i \in [\min(V_h) - \max(\rho_l, \delta), \max(V_h) + \max(\rho_l, \delta)] \)

Proof. From Lemma IV.2, every level \( l > \phi \), where \( \phi = \lceil \log(\frac{4}{\delta}) \rceil \), has weight \( w_{l,i} \leq 0 \).

First, we examine all checkpoints \( \mu_k^l \) that can have a positive weight in Line 18 in Algorithm 2 for levels \( l < \phi \). We know that only checkpoints within a \( \rho_l \) distance from an honest input \( v_i \) can have a positive weight. Therefore, no \( \mu_k^l < \min(V_h) - \rho_l \) and \( \mu_k^l > \max(V_h) + \rho_l \) can have a weight \( \omega_k > 0 \). This implies that the level’s weight \( w_{l,i} \in (\min(V_h) - \rho_l, \max(V_h) + \rho_l) \forall l \leq \phi \). The farthest point which can have a positive weight in \( \alpha_i \) until level \( l < \phi \) is \( \min(V_h) - \delta \) and \( \max(V_h) + \delta \), because \( \rho_l < \delta \), \( \forall l \leq \phi \).

Second, we examine checkpoints \( \mu_k^l \) at \( l = \phi \), where \( \rho_\phi \in [\delta, 2\delta] \). There exists at least one \( \mu_k^l \) such that \( |\mu_k^l - v_i| \leq \rho_\phi, \forall v_i \in P \), which implies \( w_{k}^\phi = 1 \). The only other checkpoints that can have a non-zero weight are \( \mu_k^{l-1} \) and \( \mu_k^{l+1} \), which ensures that level \( \phi \)'s representative value \( w_\phi \) must be within \([\frac{\phi^{l-1} + \mu_k^{l-1} + \mu_k^{l+1}}{2}] \). As there must exist honest nodes \( i,j \) such that \( |\mu_k^{l-1} - v_i| < 2\delta \) and \( |\mu_k^{l+1} - v_j| < 2\delta \), the relaxation can be at most \( \delta \), which proves the theorem. □

C. Agreement

We prove that DELPHI achieves \( \epsilon \)-agreement.

Theorem IV.4. Given that the BINA protocol satisfies Agreement, Termination, and Validity, outputs of honest nodes \( o_i \), after terminating DELPHI must be within \( \epsilon \) distance of each other; i.e., \( |o_i - o_j| \leq \epsilon V \{i,j \} \in P \).

Proof. We consider the difference between the weighted averages of two honest nodes \( O = o_i - o_j \).

\[
O = \left( \sum_{l=0}^{l_m} w_{l,i} V_{l,j} \right) - \left( \sum_{l=0}^{l_m} w_{l,j} V_{l,i} \right)
\]

We assume \( w_{l,j} = w_{l,i} + c_l \). The equation then transforms to the following.

\[
O = \left( \sum_{l=0}^{l_m} w_{l,i} (V_{l,i} - V_{l,j}) + c_l (V_{avg,i} - V_{l,j}) \right)
\]

The term \( V_{avg,i} = \sum_{l=0}^{l_m} w_{l,i} V_{l,i} / \sum_{l=0}^{l_m} w_{l,i} \). From Theorem IV.1, we know that \( \sum_{l=0}^{l_m} w_{l,i} \geq \frac{1}{2} \). Therefore, we rewrite the equation as the sum of the following two terms and then examine each term individually.

\[
O \leq \sum_{l=0}^{l_m} \left( w_{l,i} |V_{l,i} - V_{l,j}| + c_l |V_{avg,i} - V_{l,j}| \right)
\]

We split the above sum into two different terms:

\[
O_1 = \sum_{l=0}^{l_m} \left( w_{l,i} |V_{l,i} - V_{l,j}| \right)
\]

We analyze level \( l \)'s contribution in \( O_1 \) denoted by \( O_{1,l,i} = w_{l,i} |V_{l,i} - V_{l,j}| \). Honest nodes can input 1 to at most \( \Delta \phi_l \) checkpoints at level \( l \), bounded by the total number of honest nodes \( n - t \). We denote the parameter \( |U|_{max} = \min(\frac{2\Delta}{\rho_0}, n - t) \), where the weighted average can have a non-zero weight for at most \( |U|_{max} \) checkpoints. We refer to these intervals using the terms \( \omega_{k,i} \) and \( \mu_k \), where \( k \in \{1, \ldots, n - t\} \). Additionally, \( w_{l,j} \leq w_{l,i} \) based on Fig. 3. We rewrite the above equation with these variables.

\[
O_{1,l,i} \leq w_{l,i} \left( \sum_{k=1}^{l_m} |w_{k,i} \mu_k| - \sum_{k=1}^{l_m} |w_{k,i} \mu_k| \right)
\]

We write \( \omega_{k,i} = \omega_{k,i} + c_k \).

\[
O_{1,l,i} \leq w_{l,i} \left( \sum_{k=1}^{l_m} |w_{k,i} \mu_k| \right)
\]

As \( w_{l,i} = \max(\omega_{k,i} : k \in \{1, \ldots, |U|_{max}\}) \), the term \( \sum_{k=1}^{l_m} |w_{k,i} \mu_k| \leq 1 + 2\epsilon V \{i,j \} \leq 2 \). Therefore, the term \( O_1 \) is upper bounded as follows: \( O_2 \leq 2\Delta M \epsilon |U|_{max} \).

Now, we examine the second term \( O_2 = \sum_{l=0}^{l_m} c_l |V_{avg,i} - V_{l,j}| \). From Lemma IV.2, we know that the level weights \( |V_{l,i} - V_{l,j}| \leq \epsilon \), meaning \( c_l \leq \epsilon V \forall l \in L \). Further, after a level \( l > \log_2(\frac{2\Delta}{\rho_0} + 1) \), we know \( w_{l,j} = 0 \) and \( c_l = 0 \). Finally, from the Validity of DELPHI in Theorem IV.3, \( w_{avg,i} \) always lies at a distance at most \( \delta \) from other honest inputs \( v_i \). Finally, the difference \( |V_{avg,i} - V_{l,j}| < 3\delta \). This gives \( O_2 \leq 6\Delta M \epsilon \).

Overall,

\[
O \leq 2\Delta M \epsilon |U|_{max} + 3 \leq 4\Delta M \epsilon |U|_{max}
\]

The parameters \( |U|_{max} = \min(\frac{2\Delta}{\rho_0}, n), \), \( l_M = \log(\frac{\Delta}{\rho_0}) \).

Therefore, DELPHI achieves Approximate Agreement when run for \( O(\frac{\Delta l_M}{\epsilon}) \) rounds. □

D. Complexity Analysis

We analyze the communication complexity of DELPHI. A level \( l \) has at most \( \frac{\delta}{\rho_l} \) checkpoints \( \mu_k^l \) to which honest nodes can input 1. Based on the optimization we described in Section III-C, level \( l \) requires \( |U|_{max} \min(\frac{2\Delta}{n}, n) \) bits of communication per round. Overall, DELPHI uses \( |U|_{max} \min(\frac{2\Delta}{\rho_0}, n|U|_{max}) \) bits of communication per
round. Combined over \( r_M \) rounds, this complexity totals to \( \mathcal{O}(n^2 \min(\frac{1}{\alpha}, n)\log(\frac{\Delta_{\text{min}}}{\alpha}) \)) bits.

**Setting parameters for DELPHI.** DELPHI requires setting \( \rho_0 \) and \( \Delta \) as global system parameters. For the scope of this paper, we statically set \( \rho_0 = \epsilon \), which ensures the minimum Validity relaxation is \( \epsilon \). We set the parameter \( \Delta \) using the assumption that honest nodes’ inputs come from a thin-tailed probability distribution. We theoretically analyze DELPHI’s communication complexity (in bits) under different distributions. We discuss the communication complexity under different \( \Delta, \epsilon, \) and \( \delta \) in Table II. Empirically, we collect and analyze data from the underlying applications and statistically infer the maximum possible difference between honest inputs. In the absence of such historical data from the application, \( \Delta \) can be set based on domain knowledge. For example, in a use case where nodes want to agree on the price of Bitcoin, we can set \( \Delta \) to be the maximum possible price observed so far - for example, \( \Delta = 1000000 \). This way of setting \( \Delta \) degrades the performance of DELPHI by increasing its round complexity.

**Analysis under probability distributions.** Many applications model real-world events using probability distributions. This analysis enables us to estimate the parameters in use and accordingly benchmark DELPHI’s performance. We consider two types of distributions: (a) Thin-tailed distributions like Normal, Gamma, and Lognormal, and (b) Fat-tailed distributions like Pareto. We use the extreme value theory and order statistics to conduct this analysis. We calculate \( \Delta \) based on a parameter \( \lambda \), where we ensure that the range \( \delta \leq \Delta \) with probability \( 1 - \text{neg}(\lambda) \).

Thin-tailed distributions are used in error modeling in practice. For instance, Normal, Lognormal, and Gamma distributions have been used in sensor noise modeling and modeling sizes of insurance claims, respectively [12], [29]. The extreme value theory suggests that the difference \( \delta \) between the maximum and minimum of \( n \) randomly drawn samples from both Normal and Gamma distributions follows a Gumbel distribution, with CDF of \( F(x) = e^{-e^{-x}} [29, \text{page } 155] \), and the mean of the distribution of \( \delta \) grows as \( \mathcal{O}(\log(n)) \) for Normal and Gamma distributions. We calculate \( \Delta \) using \( \lambda \) and \( n \) in the CDF. Using this estimation, we observe that \( \Delta = \mathcal{O}(\log(n)) \). With this \( \Delta \) value, DELPHI’s communication complexity is \( \mathcal{O}(n^2 \log(\frac{\lambda}{\epsilon}) \log \log(\log(n))) \).

In the case of honest samples coming from distributions like Pareto or Loggamma distributions with shape parameter \( \alpha \), which have relatively fatter tails, the range of honest inputs follows a Frechet distribution, with CDF \( F(x) = e^{-x^{-\alpha}} [29, \text{page } 153] \). The mean of this distribution grows as \( \mathcal{O}(n^{-\frac{1}{\alpha}}) \). In this case, we observe that \( \Delta = e^{n^{-\frac{1}{\alpha}}} \), which gives DELPHI’s communication to be \( \mathcal{O}(n^2 \log(\frac{\lambda}{\epsilon}) \log(\log(\log(n))) + \frac{\Delta}{\alpha} \log(n))) \) bits.

**V. APPLICATION**

Oracle networks play an important role in the web3 ecosystem. They provide trusted information about real-world events to blockchains, which utilize it to execute logic on smart contracts. Oracle networks empower multiple applications in the DeFi space, like provisioning loans and trading futures, by providing reliable market data about stocks and cryptocurrencies [37]. Oracle nodes also play an important role in decentralized insurance applications like crop insurance, providing metrics about critical variables like forecasted rainfall and potential drought [37]. In these cases, the network must provide values representative of the real-world event, even in the presence of malicious nodes.

An oracle network must provide an attested value within the range of honest nodes’ inputs to the blockchain, which verifies and includes it in the ledger. Chakka et al. [20] defined this as the Distributed Oracle Agreement (DORA) problem, where nodes must agree on a value within the range of honest inputs, assisted by the external blockchain modeled as an SMR channel. A strawman solution to solving DORA is for each honest oracle to broadcast a signature on its input \( v_i \), collect \( n - t \) other signed values, submit this list of \( n - t \) values to the SMR channel, wait for the first list in the SMR channel, and output the median of this list. The properties of SMR ensure that all honest oracles see the same first list and reach an agreement. However, this approach requires the SMR channel to verify \( \mathcal{O}(n) \) signatures and include a message of size \( \mathcal{O}(n \kappa) \) in the ledger, which makes it inefficient and expensive.

Prior approaches like Chainlink’s reporting protocol [16] and Chakka et al. [20] generate an aggregated BLS signature of size \( \mathcal{O}(n + \kappa) \) on a single value within the range of honest inputs\(^4\). In Chainlink, nodes use partially synchronous convex BA to reach an agreement and then create a signature on the output [16], Chakka et al. builds on the described strawman approach by adding a round to generate a succinct signature [20], and uses the SMR channel for agreement. However, both approaches are computationally expensive with \( \mathcal{O}(n^2) \) signature verifications per node per attested value (while achieving adaptive security). Refer to Table III for a detailed comparison between these approaches.

We solve the DORA problem in a computationally efficient manner using DELPHI. We introduce an additional step after

\[^4\] Message size can be reduced to \( \mathcal{O}(\kappa) \) with threshold BLS signatures, but require a threshold setup.
reaching Approximate Agreement, where nodes round off their input to the closest integer multiple of \( \epsilon \). Next, they broadcast a signature on their rounded values, wait for \( t + 1 \) signatures on a given value, and aggregate them to form a succinct signed message.

**Intuition.** After rounding, honest nodes’ outputs must be within at most two adjacent \( \epsilon \)-checkpoints, which implies at least \( t + 1 \) signatures must be broadcasted for at least one checkpoint. Further, no other value can receive \( \geq t + 1 \) signatures. Therefore, DELPHI solves the DORA problem without using any additional computation than what is required to send the output to the SMR channel. The rounding operation incurs a further \( \epsilon \) relaxation to the Validity.

VI. EVALUATION

We evaluate DELPHI in the context of two applications: (a) an oracle network reporting the price of a cryptocurrency, and (b) a group of surveillance drones agreeing on the location of an object. We analyze the input data in each application and accordingly configure DELPHI to maximize efficiency while ensuring security.

A. Oracle Network

In the first application, we consider an oracle network that reports the trading price of Bitcoin, a prominent cryptocurrency, in US Dollars. We configure the network to produce one price report every minute. Nodes in the Oracle network aim to agree on the current price of Bitcoin. Once a minute, each node measures the trading price of Bitcoin by querying one or a set of prominent cryptocurrency exchanges and computing the median of responses. Nodes then input this price, \( v_i \), into an instance of DELPHI protocol and report the output as Bitcoin’s trading price for that minute.

**Range analysis.** We configure DELPHI based on a comprehensive analysis of the difference between honest inputs or range \( \delta = \max(V_h) - \min(V_h) \), where \( V_h \) is the set of inputs of honest nodes. We collected price feeds of Bitcoin from 10 prominent exchanges: Binance, Coinbase, Crypto.com, Gate.io, Huobi, Mexc, Poloniex, Bybit, Kucoin, and Kraken, for a two-week timespan from July 7th to July 21st, 2023, at a granularity of one reading every minute. We then calculated the maximum difference between the prices reported by these exchanges each minute, giving us one value per minute for two weeks. This value represents the difference between honest nodes’ inputs or \( \delta \) because each node queries at least one exchange. We then plot a histogram of all these \( \delta \) values based on their frequency of occurrence in Fig. 4.

We observe that beyond \( \delta = 308 \), higher \( \delta \) values become increasingly improbable. The \( \delta \) values are below 1008 for 99.2% of the time and below 3008 for 100%. We also fit various probability distributions for this data and observed the Frechet distribution with \( \alpha = 4.11 \) and a scale factor of 29.3 to be the best fit for this data, implying the underlying distribution of inputs is a Loggamma distribution. We configure DELPHI’s parameters using these probability distribution parameters. We derive maximum range \( \Delta = 20000 \) such that the range \( \delta > \Delta \) with probability \( p = 1 - 10^{-10} \), which guarantees a statistical security of \( \lambda = 30 \) bits. In this context and under these distribution parameters, assuming we run one instance of DELPHI every minute, this assumption gets broken once every 2000 years. We set \( p_0 = \epsilon = 28 \).

B. Distributed CPS

In the second application, we consider a group of drones flying in an area, aiming to detect and localize objects. Such systems have gained increased prominence in light of recent advances in autonomous and semi-autonomous CPS [5], [8]. The drones are equipped with cameras to take photos of the area and are guided by the GPS navigation system. Each drone detects cars in the area using a deep learning-based object detection program like EfficientDet [45]. This program takes an image as input and outputs a rectangular bounding box of relative coordinates in the image containing the object. Each drone uses the center of this bounding box and its own geo-location provided by the GPS to estimate the geo-location of the target object. Given that the bounding box coordinates measured by drone \( i \) is \( L_{BB,i} \), its height is \( h \), and the drone’s 2D location from GPS is \( L_{GPS,i} \), the estimate of the target’s 2D location \( \hat{L}_{T,i} \) is calculated as \( \hat{L}_{T,i} = L_{BB,i} + L_{GPS,i} \). The drones use this estimated location as an input to DELPHI protocol to agree on the geo-location of each target in the area. As input \( L_{T,i} = (x, y) \) is a 2D vector, drones use two instances of DELPHI to agree on each coordinate, \( \hat{L}_{T,i} - \hat{x} \), individually.

**Range Analysis.** We analyze the coordinate-wise ranges \( \delta_x = |\max(L_{T,i} - x) - \min(L_{T,i} - x)| \) and \( \delta_y = |\max(L_{T,i} - y) - \min(L_{T,i} - y)| \) by first analyzing the distribution of the error distance \( d_i = |L_{T,i} - L_{GT}^i| \). \( d_i \) quantifies the accuracy of the estimate \( \hat{L}_{T,i} \) with the true location \( L_{GT}^i \) of the object. We then use extreme value theory on this distribution to estimate the ranges \( \delta_x, \delta_y \). We consider cars to be the objects that need to be localized. We assume the drones are deployed at height \( h = 45m \) or 150ft. The vectors \( L_{BB,i} \) and \( L_{GPS,i} \) both contribute to the parameter \( d_i \).
Table III: Comparison of relevant oracle reporting protocols

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Chainlink [16]</td>
<td>p sync</td>
<td>$O(\log n + \kappa n^2)$</td>
<td>×</td>
<td>$O(1)$</td>
<td>$O(1)$</td>
<td>4</td>
<td>[m, M]</td>
</tr>
<tr>
<td>DORA [20]*</td>
<td>async</td>
<td>$O(\log n + \kappa n^2)$</td>
<td>×</td>
<td>$O(1)$</td>
<td>$O(n)$</td>
<td>3</td>
<td>[m, M]</td>
</tr>
<tr>
<td>DELPHI*</td>
<td>async</td>
<td>$O(\log \frac{\log \frac{\log \frac{\log n}}{\log \lambda(n)}}{\log n})$</td>
<td>✓</td>
<td>0</td>
<td>0</td>
<td>$[m - \delta - \epsilon, M + \delta + \epsilon]$</td>
<td></td>
</tr>
</tbody>
</table>

$V_h$ is the set of all honest inputs, $l$ is the size of each input, $m = \min(V_h)$, $M = \max(V_h)$, $\delta = M - m$ is the range of honest inputs, and $\kappa$ is the cryptographic security parameter. $l < \kappa$ in practice. SMR Channel In all protocols, oracles produce an attested output and submit it to the blockchain for consumption by smart contracts [16, Section 5.4]. This step is common for all protocols, so we do not count its cost in this table. Adaptive Security Chainlink [16] and DORA [20] can be made adaptively secure at the expense of $O(n)$ factor increase in communication and computation complexities. Chainlink also incurs a $O(n)$ increase in round complexity.  

Agreement These protocols can produce multiple outputs with the specified Validity condition. DORA [20] can produce at least $O(n)$ possible outputs, whereas DELPHI can produce at most two possible outputs. The external blockchain orders them and consumes the first output.

Distribution fitting. Overall, the parameters $\delta_i, x, \delta_i, x$ have an expected value of 2m and do not exceed $d = 10.5m$ 99.99% of the time. However, analyzing the combined probability distribution is challenging because both $L_{GPS}$ and $L_{BB}$ are from different distributions. We perform asymptotic analysis by upper bounding the $L_{GPS}$ distribution using a Gamma distribution of the same scale as $L_{BB}$. This approximation allows us to combine $L_{GPS}$ and $L_{BB}$ into a Gamma distribution with scale $= 0.18$ and shape $= 30.77$. For large $n$, extreme value theory suggests that the range of $n$ independently drawn inputs from a Gamma distribution is a Gumbel distribution [29, page 155]. From the analysis in Section IV-D, a maximum range $\Delta = O(\lambda_{\text{mean}})$. In this context, we set $\rho_0 = \epsilon = 0.5m$ and $\Delta = 50m$.

C. Implementation and Testbed Details

We implement DELPHI in Rust with the tokio library as our asynchronous runtime [6]. We use Hash-based Message Authentication Codes (HMAC) with SHA256 Hash function and shared symmetric keys to implement authenticated channels. We evaluate both protocols in two testbeds - (a) A geo-distributed testbed on AWS for the oracle network application and (b) A distributed testbed on Raspberry Pi devices for the object detection application.

AWS testbed. We evaluate DELPHI on Amazon Web Services (AWS) with varying nodes $n = 16, 64, 112,$ and 160. We run both protocols on t2.micro nodes, each with 1 CPU core and 2GB RAM. We create a geo-distributed testbed of $n$ nodes to simulate execution over the internet. We distribute the nodes equally across 8 regions: N. Virginia, Ohio, N. California, Oregon, Canada, Ireland, Singapore, and Tokyo.

Embedded Device Testbed. We set up an embedded testbed consisting of 15 Raspberry Pi 4-B devices connected with the help of a network switch functioning as a router. These devices are used in most commercially available Drones and other CPS. Each device has 2 GB of RAM and 4 CPU cores. We emulate a large network of drones by running multiple processes of DELPHI on each device. We evaluate both protocols with varying nodes $n = 43, 85, 127, 169$.

2Available at https://github.com/akhilsb/delphi-rs

Figure 5: IoU histogram for Drone-based object detection: We plot the IoU values of detections output by the detection program and analyze their incidence by bucketing them into bins. These values follow a Gamma distribution, which has a thin tail.

Intersection-over-Union(IoU) analysis. We empirically estimate the error induced by the object detector in $L_{BB,i}$ by measuring and plotting the Intersection-over-Union(IoU) $I_i$ of the predictions. The IoU measures the overlap between the detector’s output bounding box and the ground truth bounding box of the object. A higher IoU $I_i$ implies a more accurate detection. We characterize this IoU distribution by first training a model of EfficientDet [45] on a training dataset consisting of over 100000 cars [5], [19], [25] and then testing it on a test dataset with 80000 cars. We then plot a histogram of all these IoU values based on their frequency of occurrence in Fig. 5. We observe that EfficientDet produces detections with IoU following a Gamma distribution with mean $I_{\text{mean},i} = 0.87$. Further, EfficientDet outputs a detection with $I_i < 0.6$ IoU only in 37% of the cases.

We translate the IoU $I_i$ into the error distance vector $\vec{d}_i$. The $x$ and $y$ coordinates of this vector are bounded by $d_i,x, d_i,y \leq (1 - I_i) \times l_{\text{diag}}$, where $l_{\text{diag}}$ is the length of the diagonal of the ground truth bounding box. We calculate $l_{\text{diag}}$ assuming by using the standard dimensions of 5m length and 2m width of a car. This formula gives us the distribution of the vector $\vec{d}_i = (5.3 \times (1 - I_i), 5.3 \times (1 - I_i))$, with a mean of 0.7m.

For $L_{GPS}$, the US FAA report on GPS accuracy documents the horizontal location error induced by GPS over 417 million samples [15]. The accuracy is within 5m 99.99% percent of the time, with an average accuracy of 1.3m.
Comparison to prior works. We compare DELPHI to FIN [27], the State-of-the-Art ACS protocol, and Abraham et al. [1], the current best approximate agreement protocol. We also implement FIN and Abraham et al. in Rust [6].

D. Results

We run the three protocols - DELPHI, Abraham et al., and FIN - in the described applications, each in their respective testbed. We configure DELPHI according to the calculated parameters for each application. We run the oracle network in the geo-distributed AWS testbed and the drone-based object detection in the CPS testbed. We measure the runtime and consumed network bandwidth of all three protocols. We evaluate DELPHI under two δ values: δ = 208, 1808 in the oracle network, and δ = 5m, 50m in the CPS testbed to measure the average case and worst case runtime of DELPHI. We plot our findings in Fig. 6.

Scalability Results. We plot and compare the runtimes of the protocols in the oracle network in Fig. 6a. DELPHI takes only 1/3rd and 1/4th the time taken by FIN and Abraham et al. at n = 160, which demonstrates its scalability. The reason for DELPHI’s higher runtime at small n is the high round complexity of DELPHI coupled with the round trip time of a geo-distributed network. However, DELPHI’s runtime scales much better mainly because of its computational efficiency and practical communication complexity. Even at a higher δ value, DELPHI considerably outperforms both prior works because of its low message complexity. We also plot the network bandwidth consumed by each protocol in Fig. 6b. DELPHI consumes an order of magnitude lower bandwidth in both cases of δ than FIN.

We also plot the runtime of the protocols in the drone-based object detection application in Fig. 6c. The devices in this testbed have lower computing power than the AWS testbed, increasing the weight of a protocol’s computational efficiency. At n = 169 processes, DELPHI’s runtime is 1/5th that of FIN and Abraham et al. in the average case and the worst case of δ = 5, 50m, respectively. Unlike in the AWS testbed, where the round complexity played an important role at low n, DELPHI outperforms FIN in this application at all n because of their computational efficiency. We also observe that DELPHI’s runtime is impacted significantly by the range δ, which is in contrast with DELPHI’s runtime in AWS.

Runtime analysis of DELPHI. We investigate the major factors affecting DELPHI’s runtime in both testbeds. We consider two parameters - (a) The agreement ratio \( \Delta \) and (b) The range ratio \( \rho \), which control the round complexity and per-round communication complexity. In Fig. 7, we plot a heatmap of the runtime of DELPHI with n = 64 and n = 85 processes in the AWS and CPS testbeds. We observe that the round complexity is the dominant driver of runtime in AWS. However, we observe the opposite in CPS, where per-round communication volume has a greater influence. The limited network bandwidth of the devices in CPS is the rate-limiting factor. Hence, the parameters \( \epsilon \) and \( \rho_0 \) are crucial for DELPHI’s performance in AWS.

Figure 7: Runtime patterns of DELPHI on AWS and CPS: We study the runtime (in seconds) of DELPHI with varying agreement ratio \( \Delta \), which controls the round complexity, and range ratio \( \rho \), which controls the per-round communication. On AWS, the round complexity substantially influences the runtime compared to per-round communication. Whereas in CPS, we see the opposite pattern where the per-round volume of communication has more influence over runtime.

Figure 6: Scalability results of DELPHI.
AWS and CPS, respectively.

E. Validity Relaxation

In the oracle network application, DELPHI’s output is $\delta$ distance away from the range of honest inputs. As we observe in Fig. 4, the mean $\delta$ is 25$. Taking the current price of Bitcoin as 40000$, DELPHI adds 0.05% error to the price value in expectation than FIN and Abraham et al. Concretely, DELPHI’s output is 25$ away from the average of honest inputs in expectation. For FIN and Abraham et al., it is 12.5$ away from the average in expectation. Further, this error is less than 0.5% for more than 99.2% of the time. In the Drone surveillance application, the mean difference along each coordinate is $\delta_{\text{mean}} \leq \ln(n) \times 0.18 = 0.92$m. Essentially, the output of DELPHI is at most 1.3m farther away from the target object than FIN and Abraham et al. in expectation. Concretely, DELPHI’s output is 2.6m away from the average of honest inputs in expectation. Whereas for FIN and Abraham et al., the output is 1.3m away from the average of honest inputs in expectation.

VII. RELATED WORK

Many agreement protocols were proposed with the weak Validity condition [3], [22], [43], which states that if all honest nodes have the same input $v$, then the output must be $v$. Binary BA protocols where nodes input only 0 or 1 offer this form of Validity. Prominent asynchronous binary BA protocols [3], [22], [43] have an $O(n^2)$ communication complexity (even sub-quadratic [11]) and require $O(1)$ common coins. However, such protocols are unsuitable for applications with multi-valent inputs, where honest nodes often input multiple values.

Validated BA protocols have been proposed, which have an external validity property, ensuring that the output satisfies an external predicate. Validated BA (VABA) [4] and Multi-valued Validated BA (MVBA) [17], [27], [40] protocols have been proposed with external Validity with a communication complexity of $O(\kappa n^2)$, and multi-valent common coins used for electing nodes as leaders. However, local Validity assessment cannot be done for real-world data like price feeds of a stock/cryptocurrency or the location of an object in an area.

Convex Validity enables nodes to agree on a value within the range of honest inputs. One way to achieve Convex Validity is to compute the Asynchronous Common Subset (ACS) [10], which enables agreement on a set of $n - t$ inputs. The median of the ACS is guaranteed to be within the range of honest inputs. Current ACS protocols can be classified as either being BKR-style [10] or MVBA-style [17]. BKR-style ACS such as HoneyBadgerBFT [42], Beat [26], and PACE [50] use RBC and $n$ binary BA instances. However, these protocols are computationally expensive, requiring $O(n)$ common coins to terminate. A concurrent work HashRand [7] overcomes this bottleneck by building common coins from computationally efficient Hash functions. However, the described ACS protocols still cost a high $O(\kappa n^3)$ bits.

The MVBA-style ACS uses RBC with MVBA to overcome the high runtime of BKR-style ACS. In exchange for running $n$ binary BAs, MVBA uses a more complex multi-valent common coin for proposal election (PE), implemented using $O(\log(n))$ binary coins [17]. Prior MVBA-style protocols like Dumbo2 [35] and SpeedingDumbo [34] trade computational complexity for communication efficiency and therefore use $O(n^2)$ signatures per ACS. FIN-ACS [27] is the most computationally efficient ACS protocol thus far, which uses RBC and a multi-valent coin for PE. FIN-ACS has an overall computational complexity of $O(\log(n))$ coins and uses $O(\kappa n^3)$ bits, which we argue is also very expensive in our setting. Information-theoretic approaches like WaterBear [51] do not use threshold signatures but require $O(exp(n))$ communication to terminate.

Approximate Agreement (AA) protocols have been proposed as an alternative to the compute-intensive exact agreement protocols. Dolev et al. [24] proposed the first asynchronous AA protocol with $n = 5t + 1$ resilience, with $O(n^3)$ communication per round and $O(\log_2(\frac{n}{t}))$ rounds. Abraham et al.’s [1] protocol improved this resilience to $n = 3t + 1$ but requires $O(n^3)$ communication per round. Followup works explored AA over a partially connected network [47], multi-dimensional AA [41], [46], and AA with asynchronous fallback [32], [33]. However, Abraham et al.’s per-round communication of $O(n^3)$ bits has not been improved so far. Binary AA is closely related to the notion of proxcensus [30], [31], a generalization of graded consensus.

VIII. CONCLUSION

The paper introduced Delphi, a deterministic asynchronous convex BA protocol designed for applications like distributed oracles in blockchains and fault-tolerant cyber-physical systems. Addressing the challenge of maintaining convex validity in sensor/oracle nodes measuring the same source, Delphi minimizes communication overhead to $O(n^2)$ while keeping computation costs low. Unlike existing protocols relying on randomization or approximate agreement techniques with high computational or communication costs, Delphi combines agreement primitives and a novel weighted averaging technique. Experimental results demonstrate Delphi’s superior performance, achieving an 8x and 3x improvement in latency over state-of-the-art protocols in CPS and AWS environments for a system with $n = 160$ nodes.

IX. ACKNOWLEDGMENTS

We thank the reviewers and our shepherd Darya Melnyk for the helpful feedback and suggestions to improve this draft significantly. We thank Manoj Patil and Saaransh Jakhar at Supra for collecting data on cryptocurrency price feeds. We also thank Azam Ikram and Mihir Patil for help with experiments on the CPS testbed. This work was supported in part by NIFA award number 2021-67021-34252, the National Science Foundation (NSF) under grant CNS1846316 and National Science Foundation Cyber Physical Systems (CPS). C. Liu-Zhang’s research was supported by Hasler Foundation Project #23090 and ETH Zurich Leading House RPG-072023-19.


