COOL Is Optimal in Error-Free Asynchronous Byzantine Agreement

1

Jinyuan Chen

Abstract

COOL (Chen'21) is an error-free, information-theoretically secure Byzantine agreement (BA) protocol proven to achieve BA consensus in the synchronous setting for an ℓ -bit message, with a total communication complexity of $O(\max\{n\ell, nt\log q\})$ bits, four communication rounds in the worst case, and a single invocation of a binary BA, under the optimal resilience assumption $n \geq 3t+1$ in a network of n nodes, where up to t nodes may behave dishonestly. Here, q denotes the alphabet size of the error correction code used in the protocol.

In this work, we present an adaptive variant of COOL, called OciorACOOL, which achieves error-free, information-theoretically secure BA consensus in the asynchronous setting with total $O(\max\{n\ell, nt\log q\})$ communication bits, O(1) rounds, and a single invocation of an asynchronous binary BA protocol, still under the optimal resilience assumption $n \geq 3t+1$. Moreover, OciorACOOL retains the same low-complexity, traditional (n,k) error-correction encoding and decoding as COOL, with k=t/3.

I. INTRODUCTION

Byzantine agreement (BA) has been extensively studied for over forty years [1]. In a BA problem, n consensus nodes aim to agree on a common ℓ -bit message, where up to t of the nodes may be dishonest. BA is widely considered as a fundamental building block of Byzantine fault-tolerant distributed systems and cryptographic protocols [1]–[18].

In the study of multi-valued, error-free BA, significant efforts have been made to improve communication complexity, round complexity, and resilience [3]–[5], [12]–[15]. In this direction, Chen proposed the COOL (or OciorCOOL) protocol, which achieves multi-valued, error-free, information-theoretically secure BA consensus in the synchronous setting, with a communication complexity of $O(\max\{n\ell, nt\log q\})$ bits and with four communication rounds in the worst case and a single invocation of a binary BA, under the optimal resilience condition $n \ge 3t+1$ [3]–[5]. Here, q denotes the alphabet size of the error correction code used in the protocol.

The COOL protocol introduced two key primitives: *Unique Agreement* (UA) and *Honest-Majority Distributed Multicast* (HMDM).

Unique Agreement: UA is a variant of BA. In UA, each node i inputs an initial value w_i and seeks to produce an output of the form (w, s, v), where $s \in \{0, 1\}$ is a success indicator and $v \in \{0, 1\}$ is a vote. A UA protocol guarantees three properties. One property of UA is that if two honest nodes output (w', 1, *) and (w'', 1, *), respectively, then w' = w'' (Unique Agreement). Additionally, if an honest node outputs (*, *, 1), then at least t + 1 honest nodes eventually output (w, 1, *) for the same w (Majority Unique Agreement). Furthermore, if all honest nodes start with the same input value w, then all honest nodes eventually output (w, 1, 1) (Validity).

Honest-Majority Distributed Multicast: In the HMDM problem, at least t+1 honest nodes act as senders, each multicasting a message to all n nodes. The HMDM property ensures that if all honest senders input the same message w, then every honest node eventually outputs w.

In this work, we present an adaptive variant of COOL, called OciorACOOL, which achieves BA consensus in the asynchronous setting with total communication $O(\max\{n\ell, nt\log q\})$ bits, O(1) rounds, and a single invocation of an asynchronous binary BA protocol, under the optimal resilience assumption $n \geq 3t+1$. Moreover, OciorACOOL retains the same low-complexity, traditional (n,k) error-correction encoding and decoding as COOL, with k=t/3.

As shown in Fig. 1, the COOL protocol is composed of three main components: COOL-UA, a binary BA (BBA), and COOL-HMDM. OciorACOOL extends COOL to the asynchronous setting. As shown in

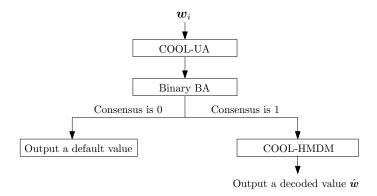


Fig. 1. A block diagram of the COOL protocol, which consists of the COOL-UA, binary BA, and COOL-HMDM algorithms.

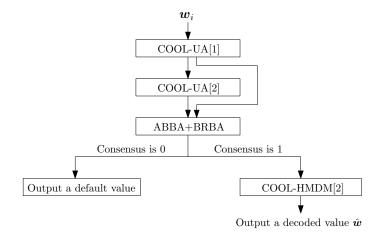


Fig. 2. A block diagram of the OciorACOOL protocol, which consists of the COOL-UA[1], COOL-UA[2], asynchronous binary BA (ABBA), binary reliable Byzantine agreement (BRBA), and COOL-HMDM[2] algorithms. If an invoked ABBA algorithm already guarantees the *Totality* property, then BRBA is not required in OciorACOOL.

Fig. 2, the OciorACOOL protocol consists of COOL-UA[1], COOL-UA[2], an asynchronous binary BA (ABBA), a binary reliable Byzantine agreement (BRBA), and COOL-HMDM[2]. Some existing ABBA protocols require each honest node to have an input value before producing an output. In such cases, BRBA is employed to ensure the *Totality* property: if one honest node outputs a value, then every honest node eventually outputs a value. If the invoked ABBA protocol already guarantees the *Totality* property, then BRBA is not required in OciorACOOL.

In the direction of error-free asynchronous BA (ABA), Li and Chen proposed an ABA protocol built on COOL that achieves communication complexity $O(\max\{n\ell, nt\log q\})$ and expected O(1) rounds, but with a weaker resilience requirement $n \geq 5t+1$ [9]. Chen later proposed a multi-valued validated Byzantine agreement protocol with expected communication complexity $O(n\ell\log n + n^2\log q)$ bits, and expected $O(\log n)$ rounds under the optimal resilience $n \geq 3t+1$ [6]. Another ABA protocol by Chen achieves expected communication complexity $O(n\ell + n^3\log n)$ bits and expected round complexity O(1), under optimal resilience $n \geq 3t+1$ [7]. Erbes and Wattenhofer proposed an ABA protocol with communication complexity $O\left(\frac{n\ell}{\min\{1,\epsilon^2\}} + n^2\max\{1,\log\frac{1}{\epsilon}\}\right)$ bits and near-optimal resilience $t < \frac{n}{3+\epsilon}$ for some $\epsilon > 0$ [19].

Recently, Abraham and Asharov [20] built upon COOL by incorporating list decoding [21], [22] for an (n,k') error correction code with k'=t/7, achieving asynchronous BA consensus with a communication complexity of $O(\max\{n\ell,n^2\log n\})$ bits and a single invocation of a binary BA protocol. However, compared to traditional unique decoding of error correction codes, list decoding is less mature in terms of practical implementations. Therefore, off-the-shelf error correction code decoders cannot be directly used for the protocol in [20].

In contrast, OciorACOOL achieves error-free ABA without relying on list decoding. It retains the same low-complexity, traditional (n,k) error-correction encoding and decoding as COOL, with k=t/3. Compared to the protocol in [20], OciorACOOL achieves a smaller communication cost, providing at least a 57% reduction in total communication.

A. Primitives

Information-Theoretic (IT) Protocol. A protocol that guarantees all required properties without relying on any cryptographic assumptions—such as digital signatures or hash functions—except for the common coin or binary BA assumptions, is said to be *information-theoretically secure*.

Error-Free Protocol. Under the common coin or binary BA assumptions, a protocol that guarantees all required properties in *every* execution is said to be *error-free*.

Error Correction Code (ECC). An (n,k) error correction code consists of an encoding function ECCEnc: $\mathbb{F}_q^k \to \mathbb{F}_q^n$ and a decoding function ECCDec: $\mathbb{F}_q^{n^\diamond} \to \mathbb{F}_q^k$, where \mathbb{F}_q denotes the alphabet of each symbol, and $n^\diamond \leq n$ represents the number of symbols available for decoding. Specifically, the encoding $[y_1, y_2, \ldots, y_n] \leftarrow \text{ECCEnc}(n, k, \boldsymbol{w})$ outputs n encoded symbols, where $y_j(\boldsymbol{w})$ denoted the j-th symbol encoded from the message \boldsymbol{w} . An (n,k) Reed-Solomon (RS) code can correct up to t Byzantine (arbitrary) errors in n^\diamond observed symbols, provided that $2t + k \leq n^\diamond$ and $n^\diamond \leq n$. The RS code operates over a finite field \mathbb{F}_q , subject to the constraint $n \leq q - 1$ (cf. [23]), where q denotes the alphabet size. Since RS codes are limited by the constraint $n \leq q - 1$, alternative error correction codes with constant alphabet size, such as expander codes [24], can be used instead.

Online Error Correction (OEC). Online error correction is particularly useful for decoding messages in asynchronous settings [25]. A node may not be able to decode the message from n^{\diamond} symbol observations, when the actual number of Byzantine errors among these observations, denoted by t^{\diamond} , satisfies $2t^{\diamond}+k>n^{\diamond}$. In such cases, the node waits for an additional symbol before attempting to decode again. This procedure repeats until the message is successfully reconstructed. In the worst case, OEC may perform up to t such trials before decoding the message.

Definition 1 (**Byzantine Agreement** (BA)). *In a* BA *protocol, consensus nodes aim to reach an agreement on a common value. It guarantees three properties:*

- **Termination:** If all honest nodes have received their inputs, then every honest node eventually produces an output and terminates.
- Consistency: If any honest node outputs a value w, then all honest nodes eventually output the same value w.
- Validity: If all honest nodes start with the same input value w, then every honest node eventually outputs w.

Definition 2 (**Reliable Byzantine Agreement** (RBA)). RBA *is a variant of both the Byzantine agreement and reliable broadcast problems. An* RBA *protocol guarantees the following three properties:*

- Consistency: If any two honest nodes output w' and w'', respectively, then w' = w''.
- **Validity:** *Same as in* BA.
- Totality: If one honest node outputs a value, then every honest node eventually outputs a value.

The proposed OciorACOOL uses a binary RBA (BRBA), which is described in Lines 27-33 of Algorithm 1. In Algorithm 4, we provide an optimal, error-free, multi-valued RBA protocol, denoted as OciorRBA, which is an updated version of Algorithm 4 in [6]. The OciorRBA protocol achieves error-free multi-valued RBA consensus with a total communication complexity of $O(\max\{n\ell, n^2 \log q\})$ bits, requiring at most *five* asynchronous communication rounds in the worst case (*four* rounds in the good case, when all honest nodes have the same input message), under the optimal resilience condition $n \ge 3t + 1$. The proof of OciorRBA is similar to that of the OciorRBC protocol (see the proof of OciorRBC in [5]).

Definition 3 (Reliable Broadcast (RBC)). In the reliable broadcast problem, a designated leader disseminates an input message to n nodes. A protocol implementing RBC must satisfy the following properties:

- Consistency: Same as in RBA.
- Validity: If the leader is honest and broadcasts a value w, every honest node eventually outputs w.
- Totality: Same as in RBA.

In Algorithm 5, we present an optimal, error-free, multi-valued, balanced RBC protocol, denoted as OciorRBC, which is an updated version of Algorithm 3 in [5]. The OciorRBC protocol achieves error-free multi-valued RBC consensus with a total communication complexity of $O(\max\{n\ell, n^2 \log q\})$ bits (a per-node communication complexity of $O(\max\{\ell, n \log q\})$ bits), requiring at most six asynchronous communication rounds in the worst case (or five rounds in the good case, without balanced communication), under the optimal resilience condition $n \geq 3t+1$ (see the proof of OciorRBC in [5]).

Definition 4 (**Honest-Majority Distributed Multicast** (HMDM) [3]–[5]). *In the* distributed multicast (DM) *problem, a subset of nodes acts as senders that multicast messages to all n nodes, where up to t nodes may be dishonest. Each sender node has its own input message. A protocol is called a DM protocol if it satisfies the following property:*

• Validity: If all honest senders input the same message w, then every honest node eventually outputs w.

The DM problem is referred to as an honest-majority distributed multicast (HMDM) if at least t+1 of the senders are honest. The HMDM abstraction was used as a building block in the COOL protocol [3]–[5].

Definition 5 (Strongly-Honest-Majority Distributed Multicast (SHMDM) [6]). A distributed multicast problem is called a strongly-honest-majority distributed multicast if the set of senders is fixed and denoted by $S \subseteq [n]$, where $|S| \ge 3t + 1$. A SHMDM protocol satisfies the following property:

• Validity: If all honest senders in S input the same message w, then every honest node eventually outputs w.

Definition 6 (Unique Agreement (UA) [3]–[5]). In a UA protocol, each node begins with an initial input value and aims to produce an output of the form (w, s, v), where $s \in \{0, 1\}$ denotes a success indicator and $v \in \{0, 1\}$ represents a vote. A UA protocol must satisfy the following properties:

- Unique Agreement: If two honest nodes output (w', 1, *) and (w'', 1, *), respectively, then w' = w''.
- Majority Unique Agreement: If an honest node outputs (*,*,1), then at least t+1 honest nodes eventually output $(\boldsymbol{w},1,*)$ for the same \boldsymbol{w} .
- Validity: If all honest nodes begin with the same input value w, then all honest nodes eventually output (w, 1, 1).

Notations: In an asynchronous setting, when measuring communication rounds, we use asynchronous rounds, where each round need not be synchronized across the distributed nodes. Let := denote "is defined as." Let \bot denote a default or empty value. Let $[b]:=\{1,2,\ldots,b\}$ and $[a,b]:=\{a,a+1,a+2,\ldots,b\}$ for integers a and b. We use f(x)=O(g(x)) to denote that $\limsup_{x\to\infty}\frac{|f(x)|}{g(x)}<\infty$. We use $f(x)=\Omega(g(x))$ to denote that $\limsup_{x\to\infty}\frac{|f(x)|}{(\log x)^ag(x)}<\infty$ for some constant $a\ge 0$.

II. OciorACOOL

The proposed OciorACOOL is an error-free asynchronous Byzantine agreement protocol. OciorACOOL does not rely on any cryptographic assumptions such as signatures or hashing, except for a single invocation of an asynchronous binary BA protocol. OciorACOOL is an adaptive variant of COOL.

Algorithm 1 OciorACOOL protocol with identifier ID. Code is shown for Node i for $i \in [n]$.

```
1: Initially set k \leftarrow t/3; \tilde{\boldsymbol{w}}_i \leftarrow \bot; \boldsymbol{w}^{(i)} \leftarrow \bot; I_{\text{oec}} = 0; I_{\text{oecfinal}} \leftarrow 0; \bar{\mathbb{Y}}_{\text{oec}} \leftarrow \{\}; \mathbb{Y}_{\text{oec}} \leftarrow \{\}; \mathbb{M} \leftarrow \{\}; I_3 \leftarrow 0; \check{\boldsymbol{y}}_i \leftarrow \bot
2: wait until the delivery of \bar{\mathbf{s}}_i^{[1]}, \bar{\mathbf{s}}_i^{[2]}, \bar{\mathbb{S}}_1^{[1]}, \bar{\mathbb{S}}_2^{[1]}, \bar{\mathbb{S}}_2^{[1]}
         3: upon receiving a non-empty message input w_i do:
                pass w_i into COOL-UA[1] as an input value
         5: upon receiving (NewSYMBOL, ID, \check{y}_j) from Node j for the first time, and j \notin \bar{\mathbb{Y}}_{oec} do:
  7:
                if |\bar{\mathbb{Y}}_{\text{oec}}| \geq k + t and I_{\text{oec}} = 0 then
                                                                                                                                                                                                    // online error correcting (OEC)
  8:
                        \hat{\boldsymbol{w}} \leftarrow \text{ECCDec}(n, k, \bar{\mathbb{Y}}_{\text{oec}})
  9:
                        [y_1, y_2, \cdots, y_n] \leftarrow \text{ECCEnc}(n, k, \hat{\boldsymbol{w}})
                        if at least k+t symbols in [y_1,y_2,\cdots,y_n] match with those in \bar{\mathbb{Y}}_{oec} then set \tilde{\boldsymbol{w}}_i\leftarrow\hat{\boldsymbol{w}} and I_{oec}\leftarrow 1
 10:
11: upon COOL-UA[1] having delivered (SYMBOL, 1, (*, \bar{y}_i^{(j)})) and \bar{\mathbb{S}}_1^{[1]} such that j \in \bar{\mathbb{S}}_1^{[1]}, and j \notin \bar{\mathbb{Y}}_{oec} do:
                 \bar{\mathbb{Y}}_{\text{oec}}[j] \leftarrow \bar{y}_{i}^{(j)}
12:
                run the OEC steps as in Lines 7-10
13:
        upon delivery of (SYMBOL, 1, (\bar{y}_i^{(j)}, *)) from COOL-UA[1] do:
14:
                if \bar{y}_i^{(j)} \in \mathbb{M} then \mathbb{M}[\bar{y}_i^{(j)}] \leftarrow \mathbb{M}[\bar{y}_i^{(j)}] \cup \{j\} else \mathbb{M}[\bar{y}_i^{(j)}] \leftarrow \{j\}
16: upon (|\mathbb{M}[y^*] \cup \bar{\mathbb{S}}_0^{[2]}| \ge n - t) \wedge (|\mathbb{M}[y^*]| \ge n - 2t) \wedge (\bar{\mathbf{s}}_i^{[1]} \ne 1) \wedge (\check{y}_i = \bot) for some y^* do: \#\bar{\mathbf{s}}_i^{[1]}, \bar{\mathbb{S}}_0^{[2]} updated from COOL-UA[1]
                set \check{y}_i \leftarrow y^*; and send (NewSYMBOL, ID, \check{y}_i) to all nodes
         18: upon delivery of \bar{s}_i^{[2]} = 1 from COOL-UA[1], and COOL-UA[2] has no input message yet do:
19:
20:
                pass \tilde{w}_i into COOL-UA[2] as an input message
21: upon \tilde{w}_i \neq \bot, and COOL-UA[2] has no input message yet do:
                pass \tilde{\boldsymbol{w}}_i into COOL-UA[2] as an input message
         // ************ ABBA ***********
23: upon delivery of [\tilde{w}^{(i)}, \tilde{\mathbf{s}}_i^{[2]}, \tilde{\mathbf{v}}_i] from COOL-UA[2], and ABBA has no input yet do:
                pass \tilde{\mathbf{v}}_i into ABBA as an input message
                                                                                                                                               // an asynchronous binary BA (ABBA) protocol
25: upon delivery of [\bar{\boldsymbol{w}}^{(i)}, \bar{s}_i^{[2]}, \bar{v}_i = 0] or \bar{s}_i^{[2]} = 0 from COOL-UA[1], and ABBA has no input yet do:
                pass 0 into ABBA as an input message
         27: upon ABBA outputting v^* do:
                send (READY, ID, v^*) to all nodes
28:
29: upon receiving t + 1 (READY, ID, v) messages from different nodes for the same v, and (READY, ID, *) not yet sent do:
                send (READY, ID, v) to all nodes
30:
31: upon receiving 2t + 1 (READY, ID, v) messages from different nodes for the same v do:
32:
                set v^{\diamond} \leftarrow v
                if \mathbf{v}^{\diamond} = 0 then output \mathbf{w}^{(i)} = \bot and terminate else set I_3 \leftarrow 1
33:
         34: upon I_3 = 1 do:
                                                                                                                                                              // only after executing Line 33
                if COOL-UA[2] has delivered [\tilde{\boldsymbol{w}}^{(i)}, \tilde{\mathbf{s}}_i^{[2]}, \tilde{\mathbf{v}}_i] with \tilde{\mathbf{s}}_i^{[2]} = 1 then
                        output \tilde{\boldsymbol{w}}^{(i)} and terminate
36:
37:
                        wait until COOL-UA[2] delivering at least t+1 (SYMBOL, 2, (\tilde{y}_i^{(j)}, *)), \forall j \in \tilde{\mathbb{S}}_1^{[2]}, for the same \tilde{y}_i^{(j)} = y^*, for some y^*
38:
                                                                                                                                                                             // update coded symbol based on majority rule
39:
                        send (CORRECT, ID, \tilde{y}_i^{(i)}) to all nodes
40:
                        wait until I_{\text{oecfinal}} = 1
41:
                        output w^{(i)} and terminate
42:
43: upon receiving (CORRECT, ID, \tilde{y}_i^{(j)}) from Node j for the first time, j \notin \mathbb{Y}_{oec}, and I_{oecfinal} = 0 do:
                 \mathbb{Y}_{\text{oec}}[j] \leftarrow \tilde{y}_i^{(j)}
44:
45:
                if |\mathbb{Y}_{\text{oec}}| \geq k + t then
                                                                                                                                                                                                               // online error correcting
                        \hat{\boldsymbol{w}} \leftarrow \text{ECCDec}(n, k, \mathbb{Y}_{\text{oec}})
46:
47:
                        [y_1, y_2, \cdots, y_n] \leftarrow \text{ECCEnc}(n, k, \hat{\boldsymbol{w}})
                        if at least k+t symbols in [y_1,y_2,\cdots,y_n] match with those in \mathbb{Y}_{oec} then set \boldsymbol{w}^{(i)}\leftarrow\hat{\boldsymbol{w}} and I_{oecfinal}\leftarrow 1
48:
         \textbf{upon} \; \mathsf{COOL\text{-}UA}[2] \; \mathsf{having} \; \mathsf{delivered} \; (\mathsf{SYMBOL}, 2, (*, \tilde{y}_j^{(j)})) \; \mathsf{and} \; \tilde{\mathbb{S}}_1^{[2]} \; \mathsf{such} \; \mathsf{that} \; j \in \tilde{\mathbb{S}}_1^{[2]}, \; \mathsf{and} \; j \notin \mathbb{Y}_{\mathrm{oec}}, \; \mathsf{and} \; I_{\mathrm{oecfinal}} = 0 \; \mathbf{do}:
49:
                 \mathbb{Y}_{\text{oec}}[j] \leftarrow \tilde{y}_i^{(j)}
50:
                run the OEC steps as in Lines 45-48
```

51:

Algorithm 2 COOL-UA protocol in asynchronous setting, with identifier ID. Code is shown for Node i.

 $/\!/$ ** This COOL-UA protocol continuously updates and delivers the values $\mathbf{s}_{i}^{[1]}, \mathbf{s}_{i}^{[2]}, \mathbf{S}_{1}^{[1]}, \mathbf{S}_{0}^{[1]}, \mathbf{S}_{0}^{[2]}, \mathbf{v}_{i}, \boldsymbol{w}^{(i)}, \{(y_{i}^{(j)}, y_{j}^{(j)})\}_{j}$ to the main protocol that invokes it. It terminates when the main protocol terminates. ** 1: Initially set $k \leftarrow t/3; \mathbb{U}_0 \leftarrow \{\}; \mathbb{U}_1 \leftarrow \{\}; \mathbb{S}_0^{[1]} \leftarrow \{\}; \mathbb{S}_1^{[2]} \leftarrow \{\}; \mathbb{S}_1^{[2]} \leftarrow \{\}; \mathbf{w}^{(i)} \leftarrow \bot; I_{\text{ecc}} \leftarrow 0; \mathbf{s}_i^{[1]} \leftarrow \bot; \mathbf{s}_i^{[2]} \leftarrow \{\}; \mathbf{w}^{(i)} \leftarrow \bot; I_{\text{ecc}} \leftarrow 0; \mathbf{s}_i^{[1]} \leftarrow \bot; \mathbf{s}_i^{[2]} \leftarrow \{\}; \mathbf{w}^{(i)} \leftarrow \bot; I_{\text{ecc}} \leftarrow 0; \mathbf{s}_i^{[1]} \leftarrow \bot; \mathbf{s}_i^{[2]} \leftarrow \{\}; \mathbf{w}^{(i)} \leftarrow \bot; I_{\text{ecc}} \leftarrow 0; \mathbf{s}_i^{[1]} \leftarrow \bot; \mathbf{s}_i^{[2]} \leftarrow \{\}; \mathbf{w}^{(i)} \leftarrow \bot; I_{\text{ecc}} \leftarrow 0; \mathbf{s}_i^{[1]} \leftarrow \bot; \mathbf{s}_i^{[2]} \leftarrow \{\}; \mathbf{w}^{(i)} \leftarrow \bot; I_{\text{ecc}} \leftarrow 0; \mathbf{s}_i^{[1]} \leftarrow \bot; \mathbf{s}_i^{[2]} \leftarrow \{\}; \mathbf{w}^{(i)} \leftarrow \bot; I_{\text{ecc}} \leftarrow 0; \mathbf{s}_i^{[1]} \leftarrow \bot; \mathbf{s}_i^{[2]} \leftarrow \{\}; \mathbf{w}^{(i)} \leftarrow \bot; I_{\text{ecc}} \leftarrow 0; \mathbf{s}_i^{[1]} \leftarrow \bot; \mathbf{s}_i^{[2]} \leftarrow \{\}; \mathbf{w}^{(i)} \leftarrow \bot; I_{\text{ecc}} \leftarrow 0; \mathbf{s}_i^{[1]} \leftarrow \bot; \mathbf{s}_i^{[2]} \leftarrow \{\}; \mathbf{w}^{(i)} \leftarrow \bot; \mathbf{s}^{(i)} \leftarrow \bot; \mathbf{$ $\begin{array}{l} \bot; \mathbf{v}_i \leftarrow \bot \quad \text{$/\!/$\bot is a default value or an empty value} \\ \textbf{2: deliver} \ \mathbf{s}_i^{[1]}, \ \mathbf{s}_i^{[2]}, \ \mathbb{S}_1^{[1]}, \ \mathbb{S}_0^{[1]}, \ \mathbb{S}_1^{[2]}, \ \mathbb{S}_0^{[2]} \end{array}$ Phase 1 3: **upon** receiving a non-empty message input w_i do: $\begin{aligned} & \boldsymbol{w}^{(i)} \leftarrow \boldsymbol{w}_i; \ [y_1^{(i)}, y_2^{(i)}, \cdots, y_n^{(i)}] \leftarrow \text{ECCEnc}(n, k, \boldsymbol{w}_i) \\ & \text{send} \ (\text{SYMBOL}, \text{ID}, (y_j^{(i)}, y_i^{(i)})) \ \text{to Node} \ j, \ \forall j \in [n] \end{aligned}$ 5: 6: 7: **upon** receiving (SYMBOL, ID, $(y_i^{(j)}, y_i^{(j)})$) from Node j for the first time **do**: $\begin{array}{l} \text{wait until } I_{\text{ecc}} = 1 \\ \text{if } (y_i^{(j)}, y_j^{(j)}) = (y_i^{(i)}, y_j^{(i)}) \underset{\text{for } (j), \dots}{\text{then }} \mathbb{U}_1 \leftarrow \mathbb{U}_1 \cup \{j\} & \text{else } \mathbb{U}_0 \leftarrow \mathbb{U}_0 \cup \{j\} \end{array}$ 8: 9: **deliver** (SYMBOL, ID, $(y_i^{(j)}, y_i^{(j)})$) 10: 11: **upon** $(|\mathbb{U}_1| \ge n - t) \land (\mathbf{s}_i^{[1]} = \bot)$ **do**: set $\mathbf{s}_i^{[1]} \leftarrow 1$; **send** (SI1, ID, $\mathbf{s}_i^{[1]}$) to all nodes; and **deliver** $\mathbf{s}_i^{[1]}$ 12: 13: **upon** $(|\mathbb{U}_0| \ge t+1) \wedge (\mathbf{s}_i^{[1]} = \bot)$ **do**: set $\mathbf{s}_i^{[1]} \leftarrow 0$; send (SI1, ID, $\mathbf{s}_i^{[1]}$) to all nodes; and deliver $\mathbf{s}_i^{[1]}$ 14: 15: **upon** receiving (SI1, ID, $s_i^{[1]}$) from Node j for the first time **do**: $\begin{array}{l} \text{if } \mathbf{s}_{j}^{[1]} = 1 \text{ then } \mathbb{S}_{1}^{[1]} \leftarrow \mathbb{S}_{1}^{[1]} \cup \{j\} \text{ else } \mathbb{S}_{0}^{[1]} \leftarrow \mathbb{S}_{0}^{[1]} \cup \{j\} \\ \text{deliver } \mathbb{S}_{1}^{[1]} \text{ and } \mathbb{S}_{0}^{[1]} \end{array}$ 16: 17: 18: **upon** $((\mathbf{s}_i^{[1]} = 0) \lor (|\mathbb{S}_0^{[1]} \cup \mathbb{U}_0| \ge t + 1)) \land (\mathbf{s}_i^{[2]} = \bot)$ **do**: 19: set $\mathbf{s}_i^{[2]} \leftarrow 0$; **send** (SI2, ID, $\mathbf{s}_i^{[2)}$) to all nodes; and **deliver** $\mathbf{s}_i^{[2]}$ 20: **upon** $(\mathbf{s}_i^{[1]} = 1) \wedge (|\mathbb{S}_1^{[1]} \cap \mathbb{U}_1| \geq n - t) \wedge (\mathbf{s}_i^{[2]} = \bot)$ **do**: set $\mathbf{s}_i^{[2]} \leftarrow 1$; **send** (SI2, ID, $\mathbf{s}_i^{[2]}$) to all nodes; and **deliver** $\mathbf{s}_i^{[2]}$ 21: 22: **upon** receiving (SI2, ID, $s_i^{[2]}$) from Node j for the first time **do**: $\begin{array}{l} \text{if } \mathbf{s}_{j}^{[2]} = 1 \text{ then } \mathbb{S}_{1}^{[2]} \leftarrow \mathbb{S}_{1}^{[2]} \cup \{j\} \text{ else } \mathbb{S}_{0}^{[2]} \leftarrow \mathbb{S}_{0}^{[2]} \cup \{j\} \\ \text{deliver } \mathbb{S}_{1}^{[2]} \text{ and } \mathbb{S}_{0}^{[2]} \end{array}$ 23: 24: 25: **upon** $|\mathbb{S}_1^{[2]}| \ge n - t$ **do**:

A. Revisiting COOL for the Synchronous Setting

 $\mathbf{v}_i \leftarrow 1$; and **deliver** $[\boldsymbol{w}^{(i)}, \mathbf{s}_i^{[2]}, \mathbf{v}_i]$

 $\mathbf{v}_i \leftarrow 0$; and **deliver** $[\boldsymbol{w}^{(i)}, \mathbf{s}_i^{[2]}, \mathbf{v}_i]$

27: **upon** $|\mathbb{S}_0^{[2]}| \ge t + 1$ **do**:

26:

28:

Before presenting the proposed OciorACOOL protocol for the asynchronous setting, we first revisit the original COOL (or OciorCOOL) protocol designed for the synchronous setting. Note that the original COOL protocol consists of the following components: the COOL-UA, binary BA, and the COOL-HMDM algorithms, as shown in Fig. 1.

- 1) COOL-UA: COOL-UA is a UA algorithm in which each node i inputs an initial value w_i and aims to produce an output (w, s, v), where w is the updated message, $s \in \{0, 1\}$ is a success indicator, and $v \in \{0, 1\}$ is a vote. COOL-UA guarantees three properties: Unique Agreement, Majority Unique Agreement, and Validity (see Definition 6).
- 2) Binary BA: After obtaining outputs from COOL-UA, each node i passes its vote v to the binary BA consensus as input. The binary BA consensus ensures that all honest nodes reach the same decision on whether to terminate at the end of the binary BA or proceed to the COOL-HMDM phase.
- 3) COOL-HMDM: By the Consistency and Validity properties of the binary BA consensus, if any honest node enters the COOL-HMDM phase, it is guaranteed that at least one honest node has voted v=1 in the

binary BA consensus. In this case, by the Unique Agreement and Majority Unique Agreement properties of UA, at least t+1 honest nodes must have output $(\boldsymbol{w},1,*)$ from COOL-UA for the same \boldsymbol{w} . This condition satisfies the HMDM requirement in distributed multicast, i.e., at least t+1 senders are honest.

HMDM guarantees the *Validity* property: if all honest senders (with at least t+1 honest senders) input the same message w, then every honest node eventually outputs w in HMDM. COOL-HMDM is an HMDM algorithm that ensures that the encoded symbols from honest nodes can be calibrated using a majority rule, so that the calibrated symbols are encoded from the same w. Consequently, all honest nodes eventually decode and output the same final message.

B. The Challenge of COOL in the Asynchronous Setting

It can be verified that the COOL protocol satisfies the *Consistency* and *Validity* properties in the asynchronous setting. However, the main challenge lies in ensuring *termination* for the COOL protocol under asynchrony. To illustrate this challenge, we consider the following example.

Let w_i denote the initial message of node i, and let \mathcal{F} denote the index set of all dishonest nodes. We define the index groups of honest nodes as

$$\mathcal{A}_l := \{ i \in [n] \setminus \mathcal{F} \mid \boldsymbol{w}_i = \bar{\boldsymbol{w}}_l \},$$

for $l \in [\eta]$, where $\bar{w}_1, \bar{w}_2, \dots, \bar{w}_{\eta}$ are distinct non-empty ℓ -bit values and η is an integer. Let us consider the following example:

Example:
$$n = 3t + 1$$
, $|\mathcal{A}_1| = n - t - |\mathcal{F}|$, $|\mathcal{A}_2| = t$, $|\mathcal{F}| = t$.

In this asynchronous setting, the honest nodes in \mathcal{A}_1 may be unable to produce outputs $(\boldsymbol{w}, \mathbf{s}, \mathbf{v})$ in COOL-UA (see Algorithm 2). Intuitively, for this example, as shown in Algorithm 2, each honest node i in \mathcal{A}_2 eventually sets its success indicators to $\mathbf{s}_i^{[2]} = \mathbf{s}_i^{[1]} = 0$ (see Lines 9, 14, and 19) because of the mismatched initial messages between the honest nodes in \mathcal{A}_1 and \mathcal{A}_2 (see Line 9). However, each honest node i in \mathcal{A}_1 may be unable to update the values of $\mathbf{s}_i^{[2]}$ and $\mathbf{s}_i^{[1]}$ if the dishonest nodes do not send messages to the honest nodes in \mathcal{A}_1 (see Algorithm 2).

In the following, we introduce the OciorACOOL protocol, an adaptive variant of COOL, which resolves the termination challenge in the asynchronous setting while preserving the *Consistency* and *Validity* properties.

C. Overview of OciorACOOL for asynchronous stetting

The proposed OciorACOOL protocol is described in Algorithm 1 and supported by Algorithm 2. We here provide an overview of the proposed OciorACOOL, which mainly consists of the COOL-UA[1], COOL-UA[2], asynchronous binary BA (ABBA), binary RBA (BRBA), and COOL-HMDM[2] algorithms. Note that the original COOL protocol comprises the COOL-UA, binary BA (BBA), and COOL-HMDM algorithms.

As shown in the previous subsection, in the asynchronous setting, some honest nodes may be unable to produce outputs (\boldsymbol{w}, s, v) in COOL-UA[1] (see Algorithm 2). The design of OciorACOOL guarantees that all honest nodes eventually *terminate*, while preserving the *Consistency* and *Validity* properties. Before describing the OciorACOOL protocol, we first introduce several definitions.

describing the OciorACOOL protocol, we first introduce several definitions. In COOL-UA, the values of $\mathbf{s}_i^{[1]}$ and $\mathbf{s}_i^{[2]}$ are initially set to $\mathbf{s}_i^{[1]} = \bot$ and $\mathbf{s}_i^{[2]} = \bot$, respectively, where \bot denotes a default or empty value. Let

$$\ddot{\mathcal{A}}_{l}^{[p]} := \{ i \in [n] \setminus \mathcal{F} \mid \mathbf{w}_{i} = \bar{\mathbf{w}}_{l}, \ \mathbf{s}_{i}^{[p]} \neq 0 \}, \quad l \in [\ddot{\eta}^{[p]}], \quad p \in [2],$$

for some non-negative integers η , $\ddot{\eta}^{[1]}$, and $\ddot{\eta}^{[2]}$ satisfying $\ddot{\eta}^{[2]} \leq \ddot{\eta}^{[1]} \leq \eta$. In the definition of $\ddot{\mathcal{A}}_l^{[p]}$, we consider the final value of $\mathbf{s}_i^{[p]}$ if it has been updated. Intuitively, $\ddot{\mathcal{A}}_l^{[p]}$ represents the indices of honest nodes that eventually set the success indicator $\mathbf{s}_i^{[p]} = 1$ or never update $\mathbf{s}_i^{[p]}$.

We prove in Lemma 10 that if all honest nodes eventually input their initial messages and keep running the COOL-UA protocol, then it holds that

$$\ddot{\eta}^{[2]} \le 1,$$

i.e., every honest node i eventually sets $\mathbf{s}_i^{[2]} = 0$ for all $i \in \mathcal{A}_l$ and all $l \in [2, \eta]$. In the following, we consider two possible cases for COOL-UA[1]:

Case I:
$$|\ddot{\mathcal{A}}_1^{[2]}| \ge n - |\mathcal{F}| - t$$
,
Case II: $|\ddot{\mathcal{A}}_1^{[2]}| < n - |\mathcal{F}| - t$.

The OciorACOOL protocol guarantees that, by combining the following algorithms, for each of the above two cases, all honest nodes eventually *terminate*, while preserving the *Consistency* and *Validity* properties.

For notational convenience, we use $\bar{\mathbf{s}}_i^{[1]}, \bar{\mathbf{s}}_i^{[2]}, \bar{\mathbb{S}}_1^{[1]}, \bar{\mathbb{S}}_0^{[1]}, \bar{\mathbb{S}}_0^{[2]}, \bar{\mathbf{v}}_i, \boldsymbol{w}^{(i)}, \{(\bar{y}_i^{(j)}, \bar{y}_j^{(j)})\}_j$ to denote the corresponding values delivered from COOL-UA[1]. Similarly, we use $\tilde{\mathbf{s}}_i^{[1]}, \tilde{\mathbf{s}}_i^{[2]}, \tilde{\mathbb{S}}_1^{[1]}, \tilde{\mathbb{S}}_0^{[1]}, \tilde{\mathbb{S}}_0^{[1]}, \tilde{\mathbb{S}}_0^{[1]}, \tilde{\mathbb{S}}_0^{[2]}, \tilde{\mathbf{v}}_i, \boldsymbol{\tilde{w}}^{(i)}, \{(\tilde{y}_i^{(j)}, \tilde{y}_j^{(j)})\}_j$ to denote the corresponding values delivered from COOL-UA[2].

Here, $\bar{\mathbf{s}}_i^{[1]}$ and $\bar{\mathbf{s}}_i^{[2]}$ (and $\tilde{\mathbf{s}}_i^{[1]}$ and $\tilde{\mathbf{s}}_i^{[2]}$) denote the success indicators of Node i at Phase 1 and Phase 2, respectively (see Lines 12, 14, 19, and 21 of Algorithm 2). $\bar{\mathbf{S}}_b^{[p]}$ (and $\tilde{\mathbf{S}}_b^{[p]}$) denotes the index set containing the nodes that send $\bar{\mathbf{s}}_i^{[p]} = b$, for $b \in \{0, 1\}$ and $p \in \{1, 2\}$ (see Lines 17 and 24 of Algorithm 2). $\bar{\mathbf{v}}_i$ (and $\tilde{\mathbf{v}}_i$) denotes the binary vote value. $(\bar{y}_i^{(j)}, \bar{y}_j^{(j)})$ (and $(\tilde{y}_i^{(j)}, \tilde{y}_j^{(j)})$) denotes the pair consisting of the i-th and the j-th coded symbols sent from Node j.

1) COOL-UA[1]: COOL-UA[1] is a UA algorithm, presented in Algorithm 2, in which each node i inputs an initial value w_i and aims to produce an output (w, s, v), where w is the updated message, $s \in \{0, 1\}$ is a success indicator, and $v \in \{0, 1\}$ is a vote. COOL-UA[1] guarantees three properties: Unique Agreement, Majority Unique Agreement, and Validity (see Definition 6).

As mentioned earlier, some honest nodes may be unable to produce outputs $(\boldsymbol{w}, \mathbf{s}, \mathbf{v})$ in COOL-UA[1]. However, COOL-UA[1] may still update and deliver the following values and sets to the main protocol that invokes it: $\bar{\mathbf{s}}_i^{[1]}, \bar{\mathbf{s}}_i^{[2]}, \bar{\mathbb{S}}_1^{[1]}, \bar{\mathbb{S}}_0^{[2]}, \bar{\mathbb{S}}_1^{[2]}, \bar{\mathbb{S}}_0^{[2]},$ and $\{(\bar{y}_i^{(j)}, \bar{y}_j^{(j)})\}_j$.

- 2) Set the Input Value \tilde{w}_i for COOL-UA[2]: The goal of this phase is to set the input message for COOL-UA[2] by executing the steps in Lines 6-17 of Algorithm 1. This phase ensures that, in Case I, if all honest nodes eventually input their initial messages to OciorACOOL and keep running the OciorACOOL protocol, then all honest nodes eventually input the same value $\tilde{w}_i = \bar{w}_1$ into COOL-UA[2] as the input messages (see Lemma 4). Here, \bar{w}_1 is the initial message of all honest nodes within $\ddot{\mathcal{A}}_1^{[2]}$ for COOL-UA[1].
- 3) COOL-UA[2]: COOL-UA[2] is also a UA algorithm, which guarantees three properties: Unique Agreement, Majority Unique Agreement, and Validity. In Case I, if all honest nodes keep running the OciorACOOL protocol, and given the conclusion that in this case every honest node i eventually inputs the same message $\tilde{w}_i = \bar{w}_1$ to COOL-UA[2] (see Lemma 4), then every honest node i eventually sets $\tilde{s}_i^{[1]} = 1$ (see Lines 11 and 12 of Algorithm 2), every honest node i eventually sets $\tilde{s}_i^{[2]} = 1$ (see Lines 20 and 21 of Algorithm 2), and every honest node i eventually sets $\tilde{v}_i = 1$ (see Lines 25 and 26 of Algorithm 2). In this case, every honest node i eventually delivers $[\tilde{\boldsymbol{w}}^{(i)}, \tilde{s}_i^{[2]}, \tilde{v}_i = 1]$ from COOL-UA[2] (see Line 26 of Algorithm 2).
- 4) ABBA and BRBA: ABBA executes the steps in Lines 23-26 of Algorithm 1, while BRBA executes the steps in Lines 27-33.

In Case I, every honest node i eventually inputs the value 1 into ABBA, if node i has not already provided an input to ABBA (see Lines 23 and 24 of Algorithm 1). Hence, in Case I, ABBA eventually outputs a value $v^* \in \{1,0\}$ at each node, due to the *Termination* and *Consistency* properties of ABBA. Then, every honest node eventually sends (READY, ID, v^*) to all nodes in Line 28 of Algorithm 1, and every honest node eventually sets $v^{\diamond} = v^*$ in Line 32 of Algorithm 1.

In Case II, the condition $|\mathbb{S}_0^{[2]}| \ge t+1$ in Line 27 of Algorithm 2 is eventually satisfied at all honest nodes, and each honest node i eventually sets $\bar{\mathbf{v}}_i = 0$ and delivers $[\bar{\boldsymbol{w}}^{(i)}, \bar{\mathbf{s}}_i^{[2]}, \bar{\mathbf{v}}_i = 0]$ from COOL-UA[1].

Consequently, in Case II each honest node i eventually passes 0 into ABBA as its input message, if node i has not already provided an input to ABBA (see Lines 25 and 26 of Algorithm 1). In this case, every honest node eventually provides an input to ABBA, and hence ABBA eventually outputs a value $v^* \in \{1,0\}$ at each node, due to the *Termination* property of ABBA. Then, every honest node eventually sends (READY, ID, v^*) to all nodes in Line 28 of Algorithm 1, and every honest node eventually sets $v^{\diamond} = v^*$ in Line 32 of Algorithm 1.

If $v^* = 0$, then all honest nodes eventually output \perp and terminate in Line 33 of Algorithm 1. If $v^* = 1$, then all honest nodes eventually go to the COOL-HMDM[2] phase.

5) COOL-HMDM[2]: By the Consistency and Validity properties of the ABBA consensus, if any honest node enters the COOL-HMDM[2] phase, it is guaranteed that at least one honest node has voted $\mathbf{v}=1$ in the ABBA consensus. In this case, by the Unique Agreement and Majority Unique Agreement properties of COOL-UA[2], at least t+1 honest nodes must have output $(\boldsymbol{w},1,*)$ from COOL-UA[2] for the same \boldsymbol{w} . This condition satisfies the HMDM requirement in distributed multicast, i.e., at least t+1 senders are honest.

HMDM guarantees the *Validity* property: if all honest senders (with at least t+1 honest senders) input the same message \boldsymbol{w} , then every honest node eventually outputs \boldsymbol{w} in HMDM. COOL-HMDM[2] is an HMDM algorithm, executed in Lines 34-51 of Algorithm 1. Therefore, all honest nodes eventually decode and output the same final message.

D. Analysis of OciorACOOL

Our analysis closely follows that of COOL and OciorCOOL [3]–[5]. Here, we use notations consistent with that previously used for COOL and OciorCOOL. Here we use \boldsymbol{w}_i to denote the initial value of Node i. If Node i never sets an initial value, then \boldsymbol{w}_i is considered as $\boldsymbol{w}_i = \bot$, where \bot denotes a default value or an empty value. The values of $\mathbf{s}_i^{[1]}$ and $\mathbf{s}_i^{[2]}$ are initially set to $\mathbf{s}_i^{[1]} = \bot$ and $\mathbf{s}_i^{[2]} = \bot$, respectively. The set \mathcal{F} is defined as the set of indices of all dishonest nodes. We let $[n] := \{1, 2, \dots, n\}$. We define some groups of honest nodes as

$$\mathcal{A}_l := \{ i \in [n] \setminus \mathcal{F} \mid \boldsymbol{w}_i = \bar{\boldsymbol{w}}_l \}, \quad l \in [\eta]$$

$$\tag{1}$$

$$\mathcal{A}_{l}^{[p]} := \{ i \in [n] \setminus \mathcal{F} \mid \mathbf{w}_{i} = \bar{\mathbf{w}}_{l}, \ \mathbf{s}_{i}^{[p]} = 1 \}, \quad l \in [\eta^{[p]}], \quad p \in [2]$$
(2)

$$\ddot{\mathcal{A}}_{l}^{[p]} := \{ i \in [n] \setminus \mathcal{F} \mid \boldsymbol{w}_{i} = \bar{\boldsymbol{w}}_{l}, \ \mathbf{s}_{i}^{[p]} \neq 0 \}, \quad l \in [\ddot{\eta}^{[p]}], \quad p \in [2]$$

$$(3)$$

$$\mathcal{B}^{[p]} := \{ i \in [n] \setminus \mathcal{F} \mid \mathbf{s}_i^{[p]} = 0 \}, \quad p \in [2]$$
(4)

$$\ddot{\mathcal{B}}^{[p]} := \{ i \in [n] \setminus \mathcal{F} \mid \mathbf{s}_i^{[p]} \neq 1 \}, \quad p \in [2]$$

$$(5)$$

for some non-empty ℓ -bit distinct values $\bar{\boldsymbol{w}}_1, \bar{\boldsymbol{w}}_2, \cdots, \bar{\boldsymbol{w}}_\eta$ and some non-negative integers $\eta, \eta^{[1]}, \eta^{[2]}, \ddot{\eta}^{[1]}, \ddot{\eta}^{[2]}$ such that $\eta^{[2]} \leq \eta^{[1]} \leq \eta, \ddot{\eta}^{[2]} \leq \ddot{\eta}^{[1]} \leq \eta, \eta^{[1]} \leq \ddot{\eta}^{[1]}, \eta^{[2]}, \eta^{[2]} \leq \ddot{\eta}^{[2]}$. In the above definitions, we consider the final value of $\mathbf{s}_i^{[p]}$ if it has been updated. Group \mathcal{A}_l (and Groups $\mathcal{A}_l^{[p]}, \ddot{\mathcal{A}}_l^{[p]}$) can be divided into some possibly overlapping sub-groups defined as

$$\mathcal{A}_{l,j} := \{ i \in \mathcal{A}_l \mid \boldsymbol{h}_i^{\mathsf{T}} \bar{\boldsymbol{w}}_l = \boldsymbol{h}_i^{\mathsf{T}} \bar{\boldsymbol{w}}_j \}, \quad j \neq l, \ j, l \in [\eta]$$
(6)

$$\mathcal{A}_{l,l} := \mathcal{A}_l \setminus \left(\cup_{i=1, i \neq l}^{\eta} \mathcal{A}_{l,i} \right), \qquad l \in [\eta]$$

$$(7)$$

$$\mathcal{A}_{l,j}^{[p]} := \{ i \in \mathcal{A}_{l}^{[p]} \mid \mathbf{h}_{i}^{\mathsf{T}} \bar{\mathbf{w}}_{l} = \mathbf{h}_{i}^{\mathsf{T}} \bar{\mathbf{w}}_{j} \}, \quad j \neq l, \ j, l \in [\eta^{[p]}], \quad p \in [2]$$
(8)

$$\mathcal{A}_{l,l}^{[p]} := \mathcal{A}_{l}^{[p]} \setminus \left(\bigcup_{i=1, i \neq l}^{\eta^{[p]}} \mathcal{A}_{l,i}^{[p]} \right), \qquad l \in [\eta^{[p]}], \quad p \in [2]$$

$$(9)$$

$$\ddot{\mathcal{A}}_{l,j}^{[p]} := \{ i \in \ddot{\mathcal{A}}_{l}^{[p]} \mid \mathbf{h}_{i}^{\mathsf{T}} \bar{\mathbf{w}}_{l} = \mathbf{h}_{i}^{\mathsf{T}} \bar{\mathbf{w}}_{j} \}, \quad j \neq l, \ j, l \in [\ddot{\eta}^{[p]}], \quad p \in [2]$$
(10)

$$\ddot{\mathcal{A}}_{l,l}^{[p]} := \ddot{\mathcal{A}}_{l}^{[p]} \setminus \left(\cup_{j=1, j \neq l}^{\ddot{\eta}^{[p]}} \ddot{\mathcal{A}}_{l,j}^{[p]} \right), \qquad l \in [\ddot{\eta}^{[p]}], \quad p \in [2]$$
(11)

where h_i is the encoding vector of the error-correction code, such that the *i*-th encoded symbol is computed as $y_i = h_i^{\top} w$, given the input vector w, for $i \in [n]$. If a non-linear encoding is used, then the *i*-th encoded

symbol is computed as $y_i = f_i(\boldsymbol{w})$, where $f_i(\bullet)$ is a polynomial function evaluated at the *i*-th point. In this case, we can simply replace the term $\boldsymbol{h}_i^{\top} \bar{\boldsymbol{w}}_l$ with $f_i(\boldsymbol{w})$. For consistency, however, we retain the original definition with linear encoding.

We use $u_i(j) \in \{0,1\}$ to denote the link indicator between Node i and Node j, defined as

$$\mathbf{u}_{i}(j) = \begin{cases} 1 & \text{if } (y_{i}^{(j)}, y_{j}^{(j)}) = (y_{i}^{(i)}, y_{j}^{(i)}), \\ 0 & \text{otherwise,} \end{cases}$$
 (12)

(see Line 9 of Algorithm 2). In our setting, it holds true that $u_i(j) = u_j(i)$ for any $i, j \in [n] \setminus \mathcal{F}$. If $(y_i^{(i)}, y_j^{(i)})$ are never sent out by Node i or $(y_i^{(j)}, y_j^{(j)})$ are never sent out by Node j, then we consider $u_i(j) = u_j(i) = 0$.

Here we let

$$\mathbb{U}_{1}^{(i)} := \{ j \in [n] \mid (y_{i}^{(j)}, y_{j}^{(j)}) = (y_{i}^{(i)}, y_{j}^{(i)}) \}, \quad \text{and} \quad \mathbb{U}_{0}^{(i)} := \{ j \in [n] \mid (y_{i}^{(j)}, y_{j}^{(j)}) \neq (y_{i}^{(i)}, y_{j}^{(i)}) \}$$
 (13)

denote the link indicator sets \mathbb{U}_1 and \mathbb{U}_0 updated by Node i, as described in Line 9 of Algorithm 2. Here $y_i^{(j)} = \boldsymbol{h}_i^{\mathsf{T}} \boldsymbol{w}_j$ and $y_j^{(j)} = \boldsymbol{h}_j^{\mathsf{T}} \boldsymbol{w}_j$ are the i-th and the j-th coded symbols encoded from the input message \boldsymbol{w}_i of Node j. We define

$$\mathbb{V}_{1}^{(i)} := \mathbb{U}_{1}^{(i)} \setminus \mathcal{F}, \quad \text{and} \quad \mathbb{V}_{0}^{(i)} := \mathbb{U}_{0}^{(i)} \setminus \mathcal{F}. \tag{14}$$

Theorems 1-4 present the main results of OciorACOOL. Specifically, Theorems 1-3 show that OciorACOOL satisfies the Consistency, Validity, and Termination properties in all executions (*error-free*), under the assumption of an error-free ABBA invoked in this protocol. From Theorem 4, it reveals that OciorACOOL is optimal in terms of communication complexity, round complexity and resilience.

Theorem 1 (Consistency). In OciorACOOL, given $n \ge 3t + 1$, if one honest node outputs a value \mathbf{w}^* , then every honest node eventually outputs a value \mathbf{w}^* , for some \mathbf{w}^* .

Proof. In OciorACOOL, if an honest node outputs a value w^* , it must have set the variable v^{\diamond} in Line 32 of Algorithm 1. From Lemma 2, if an honest node sets $v^{\diamond} = v^*$ in Line 32 of Algorithm 1, for $v^* \in \{0, 1\}$, then all honest nodes eventually set $v^{\diamond} = v^*$. Therefore, if one honest node sets $v^{\diamond} = 0$, then all honest nodes eventually set $v^{\diamond} = 0$ and output \bot in Line 33 of Algorithm 1. In the following, we focus on the case where all honest nodes eventually set $v^{\diamond} = 1$.

When an honest node sets $v^{\diamond}=1$, it implies that ABBA must have output the value 1 (see Lines 27-32 of Algorithm 1). Due to the *Validity* and *Consistency* properties of ABBA, if ABBA outputs the value 1, then at least one honest node i has input the value 1 to ABBA and has delivered $\tilde{v}_i=1$ from COOL-UA[2] (see Lines 23 and 24 of Algorithm 1). When an honest node i has delivered $\tilde{v}_i=1$ from COOL-UA[2], at least n-2t honest nodes must have set $\tilde{s}_j^{[2]}=1$ in COOL-UA[2] (see Lines 25 and 26 of Algorithm 2).

From Lemma 10, the honest nodes that set $\tilde{\mathbf{s}}_i^{[2]} = 1$ in COOL-UA[2] must have the same input message $\tilde{\boldsymbol{w}}^{(i)} = \boldsymbol{w}^{\star}$, for some \boldsymbol{w}^{\star} . If an honest node i has set $\mathbf{v}^{\diamond} = 1$ and has delivered $[\tilde{\boldsymbol{w}}^{(i)}, \tilde{\mathbf{s}}_i^{[2]} = 1, \tilde{\mathbf{v}}_i]$ from COOL-UA[2], then this node outputs the value $\tilde{\boldsymbol{w}}^{(i)} = \boldsymbol{w}^{\star}$ (see Line 36 of Algorithm 1).

If an honest node i has set $\mathbf{v}^{\diamond}=1$ but has not yet delivered $[\tilde{\boldsymbol{w}}^{(i)},\tilde{\mathbf{s}}_i^{[2]}=1,\tilde{\mathbf{v}}_i]$ from COOL-UA[2], it can be shown that this node will eventually output the same value \boldsymbol{w}^{\star} in Line 42 of Algorithm 1. Recall that when an honest node has delivered $\tilde{\mathbf{v}}=1$ from COOL-UA[2], at least $n-2t\geq t+1$ honest nodes must have set $\tilde{\mathbf{s}}_i^{[2]}=1$ in COOL-UA[2]. In this case, from Lemma 10, at least t+1 honest nodes i must have set $\tilde{\mathbf{s}}_i^{[2]}=1$ and have sent (SYMBOL, $2, (\tilde{y}_j^{(i)}(\boldsymbol{w}^{\star}), \tilde{y}_i^{(i)}(\boldsymbol{w}^{\star})))$ to Node j, for all $j\in[n]$, where $\tilde{y}_j^{(i)}(\boldsymbol{w}^{\star})$ and $\tilde{y}_i^{(i)}(\boldsymbol{w}^{\star})$ denote the coded symbols generated from the message \boldsymbol{w}^{\star} .

From the above results, if an honest node i has set $\mathbf{v}^{\diamond}=1$ but has not yet delivered $[\tilde{\boldsymbol{w}}^{(i)},\tilde{\mathbf{s}}_i^{[2]}=1,\tilde{\mathbf{v}}_i]$ from COOL-UA[2], then it will eventually deliver from COOL-UA[2] at least t+1 matching (SYMBOL, $2, (\tilde{y}_i^{(j)}(\boldsymbol{w}^{\star}), *)$) messages for all $j \in \tilde{\mathbb{S}}_1^{[2]}$, where $\tilde{y}_i^{(j)}(\boldsymbol{w}^{\star}) = y_i^{\star}$ for some y_i^{\star} , which is the i-th symbol encoded from the message \boldsymbol{w}^{\star} . In this case, Node i sets $\tilde{y}_i^{(i)} = y_i^{\star}$ in Line 39 and sends

(CORRECT, $\mathrm{ID}, \tilde{y}_i^{(i)}$) to all nodes in Line 40. Thus, each symbol $y_j^{(j)}$ included in $\mathbb{Y}_{\mathrm{oec}} \setminus \mathcal{F}$ must be encoded from the same message \boldsymbol{w}^\star . Therefore, if an honest node i has set $\mathbf{v}^\diamond = 1$ but has not yet delivered $[\tilde{\boldsymbol{w}}^{(i)}, \tilde{\mathbf{s}}_i^{[2]} = 1, \tilde{\mathbf{v}}_i]$ from COOL-UA[2], it will eventually decode the message \boldsymbol{w}^\star using online error-correction decoding and output \boldsymbol{w}^\star in Line 42 of Algorithm 1.

Theorem 2 (Validity). Given $n \ge 3t + 1$, if all honest nodes input the same value w, then every honest node eventually outputs w in OciorACOOL.

Proof. From Theorem 3, if all honest nodes receive their inputs, then every honest node eventually outputs a value and terminates in OciorACOOL. Furthermore, from Theorem 1, in OciorACOOL, if one honest node outputs a value w^* , then every honest node eventually outputs the same value w^* , for some w^* . Thus, based on Theorem 3 and Theorem 1, what remains to prove for this theorem is that if all honest nodes input the same value w, and if an honest node i outputs a value w^* in OciorACOOL, then $w^* = w$.

First, we prove that if all honest nodes input the same message \boldsymbol{w} in OciorACOOL, then in COOL-UA[1], no honest node will set $s_i^{[1]}=0$ (see Lines 13 and 14 of Algorithm 2), no honest node will set $s_i^{[2]}=0$ (see Lines 18 and 19 of Algorithm 2), and no honest node will set $v_i=0$ (see Lines 27 and 28 of Algorithm 2). Specifically, in this case, if an honest node i sends out the message in Line 5 of Algorithm 2, then the message must be (SYMBOL, $\mathrm{ID}, (y_j^{(i)}(\boldsymbol{w}), y_i^{(i)}(\boldsymbol{w}))$), where $y_j^{(i)}(\boldsymbol{w})$ and $y_i^{(i)}(\boldsymbol{w})$ are symbols encoded from the message \boldsymbol{w} (see Lines 4 and 5 of Algorithm 2). Thus, the equality $(y_i^{(j)}(\boldsymbol{w}), y_j^{(j)}(\boldsymbol{w})) = (y_i^{(i)}(\boldsymbol{w}), y_j^{(i)}(\boldsymbol{w}))$ must hold for any $i, j \in [n] \setminus \mathcal{F}$ (see Line 9 of Algorithm 2). This implies that the condition $|\mathbb{U}_0| \geq t+1$ in Line 13 of Algorithm 2 will never be satisfied at any honest node, and therefore no honest node will set $s_i^{[1]}=0$ in Line 14 of Algorithm 2. Since no honest node will send out (SI1, $\mathrm{ID}, s_i^{[1]}=0$), the condition $|\mathbb{S}_0^{[1]}| \geq t+1$ in Line 13 of Algorithm 2 will never be satisfied at any honest node will send out (SI2, $\mathrm{ID}, s_i^{[2]}=0$), the condition $|\mathbb{S}_0^{[2]}| \geq t+1$ in Line 27 of Algorithm 2 will never be satisfied at any honest node, and hence no honest node will set $v_i=0$ in Line 28 of Algorithm 2.

Next, we prove that if all honest nodes input the same message \boldsymbol{w} in OciorACOOL, and if any honest node i passes $\tilde{\boldsymbol{w}}_i$ into COOL-UA[2] as an input message (see Lines 20 and 22 of Algorithm 1), then $\tilde{\boldsymbol{w}}_i = \boldsymbol{w}$. Specifically, the above results show that, in this case, no honest node will set $\bar{\mathbf{s}}_i^{[2]} = 0$. If any honest node i passes $\tilde{\boldsymbol{w}}_i$ into COOL-UA[2] as an input message, and if node i has set $\bar{\mathbf{s}}_i^{[2]} = 1$ from COOL-UA[1], then node i sets $\tilde{\boldsymbol{w}}_i = \boldsymbol{w}_i = \boldsymbol{w}$ and passes \boldsymbol{w} into COOL-UA[2] as its input message (see Lines 18-20 of Algorithm 1). Furthermore, from Lemma 1, if all honest nodes input the same value \boldsymbol{w} , and if an honest node i sets $\tilde{\boldsymbol{w}}_i = \hat{\boldsymbol{w}}$ in Line 10 of Algorithm 1, where $\hat{\boldsymbol{w}}$ is the message decoded from online error correction in Line 8, then $\hat{\boldsymbol{w}} = \boldsymbol{w}$. This results implies that, if all honest nodes input the same value \boldsymbol{w} , and if any honest node i passes $\tilde{\boldsymbol{w}}_i$ into COOL-UA[2] as an input message, then $\tilde{\boldsymbol{w}}_i = \boldsymbol{w}$ (see Lines 21-22 of Algorithm 1).

From the above results, if all honest nodes input the same message \boldsymbol{w} in OciorACOOL, then the input messages (if any) of all honest nodes in COOL-UA[2] must also be \boldsymbol{w} . Following from the previous arguments, in this case, within COOL-UA[2], no honest node will set $\tilde{\mathbf{s}}_i^{[1]} = 0$ (see Lines 13 and 14 of Algorithm 2), no honest node will set $\tilde{\mathbf{s}}_i^{[2]} = 0$ (see Lines 18 and 19 of Algorithm 2), and no honest node will set $\tilde{\mathbf{v}}_i = 0$ (see Lines 27 and 28 of Algorithm 2).

From Theorem 3, if all honest nodes input the same message \boldsymbol{w} in OciorACOOL, then every honest node eventually outputs a value and terminates in OciorACOOL. In this case, by combining the above results, we conclude that no honest node sets $\tilde{\mathbf{v}}_i = 0$, and that at least one honest node i delivers $[\tilde{\boldsymbol{w}}^{(i)} = \boldsymbol{w}, \tilde{\mathbf{s}}_i^{[2]}, \tilde{\mathbf{v}}_i = 1]$ from COOL-UA[2] and inputs 1 into ABBA as its input message in Line 24 of Algorithm 1. Since no honest node sets $\tilde{\mathbf{v}}_i = 0$, no honest node will input 0 into ABBA as its input message. This implies that ABBA eventually outputs 1, and all honest nodes eventually set $\mathbf{v}^{\diamond} = 1$ (see Lines 27-33 of Algorithm 1).

Consequently, if all honest nodes input the same message \boldsymbol{w} in OciorACOOL, and if an honest node i has set $\mathbf{v}^{\diamond}=1$ and has delivered $[\tilde{\boldsymbol{w}}^{(i)},\tilde{\mathbf{s}}_i^{[2]}=1,\tilde{\mathbf{v}}_i]$ from COOL-UA[2], then node i eventually outputs the value

 \boldsymbol{w} in Line 36 of Algorithm 1. If an honest node i has set $\mathbf{v}^{\diamond} = 1$ but has not yet delivered $[\tilde{\boldsymbol{w}}^{(i)}, \tilde{\mathbf{s}}_i^{[2]} = 1, \tilde{\mathbf{v}}_i]$ from COOL-UA[2], this node eventually outputs the same value \boldsymbol{w} in Line 42 of Algorithm 1 (see the last two paragraphs of the proof of Theorem 1). This completes the proof.

Lemma 1. In OciorACOOL, if all honest nodes input the same value \mathbf{w} , and if an honest node i sets $\tilde{\mathbf{w}}_i = \hat{\mathbf{w}}$ in Line 10 of Algorithm 1, where $\hat{\mathbf{w}}$ is the message decoded from online error correction in Line 8, then $\hat{\mathbf{w}} = \mathbf{w}$.

Theorem 3 (Termination). Given $n \ge 3t + 1$, if all honest nodes receive their inputs, then every honest node eventually outputs a value and terminates in OciorACOOL.

Proof. From Lemma 3, if all honest nodes receive their inputs, then at least one honest node eventually sets $v^{\diamond} = v^{\star}$ in Line 32 of Algorithm 1, for some $v^{\star} \in \{0,1\}$. From Lemma 2, if an honest node sets $v^{\diamond} = v^{\star}$ in Line 32 of Algorithm 1, then all honest nodes eventually set $v^{\diamond} = v^{\star}$. If one honest node sets $v^{\diamond} = 0$, then all honest nodes eventually set $v^{\diamond} = 0$, output \bot , and terminate in Line 33 of Algorithm 1. In the following, we focus on the case where all honest nodes eventually set $v^{\diamond} = 1$.

As in the proof of Theorem 1, if an honest node i has set $\mathbf{v}^{\diamond}=1$ and has delivered $[\tilde{\boldsymbol{w}}^{(i)},\tilde{\mathbf{s}}_i^{[2]}=1,\tilde{\mathbf{v}}_i]$ from COOL-UA[2], then this node outputs the value $\tilde{\boldsymbol{w}}^{(i)}$ and terminates in Line 36 of Algorithm 1. It has also been shown in the proof of Theorem 1 that if an honest node i has set $\mathbf{v}^{\diamond}=1$ but has not yet delivered $[\tilde{\boldsymbol{w}}^{(i)},\tilde{\mathbf{s}}_i^{[2]}=1,\tilde{\mathbf{v}}_i]$ from COOL-UA[2], then Node i eventually outputs a message and terminates in Line 42 of Algorithm 1 (see the last two paragraphs of the proof of Theorem 1).

Lemma 2 (Totality and Consistency Properties of RBA). In OciorACOOL, if an honest node sets $v^{\diamond} = v^{\star}$ in Line 32 of Algorithm 1, for $v^{\star} \in \{1, 0\}$, then all honest nodes eventually set $v^{\diamond} = v^{\star}$.

Proof. This result follows from the *Totality* and *Consistency* properties of RBA, as described in Lines 27-33 of Algorithm 1. Specifically, when an honest node sets $v^{\diamond} = v^{\star}$ in Line 32 of Algorithm 1, for $v^{\star} \in \{1, 0\}$, it implies that this node has received at least 2t + 1 (READY, ID, v^{\star}) messages (see Line 31). In this case, at least t + 1 honest nodes must have sent out (READY, ID, v^{\star}) messages.

In our setting, if an honest node has sent out a (READY, ID, v^*) message, then ABBA must have output a value v^* at at least one honest node (see Line 27). Due to the *Consistency* property of ABBA, if two messages (READY, ID, v^*) and (READY, ID, v') are sent out from two honest nodes, respectively, then $v^* = v'$.

Therefore, if an honest node sets $v^{\diamond} = v^{\star}$, then each honest node eventually sends out a message (READY, ID, v^{\star}) (see Lines 29 and 30). Consequently, each honest node eventually receives at least 2t+1 (READY, ID, v^{\star}) messages and subsequently sets $v^{\diamond} = v^{\star}$ in Line 32. Note that every honest node should have sent out a message (READY, ID, v^{\star}) and set $v^{\diamond} = v^{\star}$ before termination.

Lemma 3 (Termination). In OciorACOOL, if all honest nodes receive their inputs, then at least an honest node eventually sets $v^{\diamond} = v^{\star}$ in Line 32 of Algorithm 1, for some $v^{\star} \in \{1, 0\}$.

Proof. We prove this result by contradiction. Assume that no honest node sets the value of v^{\diamond} in Line 32 of Algorithm 1. Under this assumption, no honest node would terminate in OciorACOOL. Note that every honest node must set $v^{\diamond} = v^{\star}$ before termination (see Lines 32 and 33 of Algorithm 1).

From Lemma 10, if all honest nodes eventually input their initial messages and keep running the COOL-UA[1] protocol, then in COOL-UA[1] it holds true that $\ddot{\eta}^{[2]} \leq 1$, i.e., every honest node i eventually sets

$$\mathbf{s}_{i}^{[2]} = 0, \quad \forall i \in \mathcal{A}_{l}, \forall l \in [2, \eta]$$

$$\tag{15}$$

in COOL-UA[1]. Here, $\ddot{\mathcal{A}}_1^{[2]} := \{i \in [n] \setminus \mathcal{F} \mid \boldsymbol{w}_i = \bar{\boldsymbol{w}}_1, \ \mathbf{s}_i^{[2]} \neq 0\}$ (see (3)). From the result in (15), and based on our notation $\mathcal{B}^{[2]} := \{i \in [n] \setminus \mathcal{F} \mid \mathbf{s}_i^{[2]} = 0\}$ (see (4)), it is true that

$$|\ddot{\mathcal{A}}_{1}^{[2]}| + |\mathcal{B}^{[2]}| = n - |\mathcal{F}| \tag{16}$$

and that

$$\bigcup_{l\in[2,\eta]}\mathcal{A}_l\subseteq\mathcal{B}^{[2]}$$

for COOL-UA[1]. In the following, we consider each of the two cases for COOL-UA[1]:

Case I:
$$|\ddot{\mathcal{A}}_{1}^{[2]}| \ge n - |\mathcal{F}| - t$$
, (17)

Case II:
$$|\ddot{\mathcal{A}}_{1}^{[2]}| < n - |\mathcal{F}| - t.$$
 (18)

• Analysis for Case I:

Let us first consider Case I in (17) with $|\ddot{\mathcal{A}}_1^{[2]}| \geq n - |\mathcal{F}| - t$ for COOL-UA[1]. In this case, from Lemma 4, if all honest nodes eventually input their initial messages and keep running the OciorACOOL, COOL-UA[1] and COOL-UA[2] protocols, then every honest node i eventually inputs the same message $\tilde{\boldsymbol{w}}_i = \bar{\boldsymbol{w}}_1$ to COOL-UA[2], where $\bar{\boldsymbol{w}}_1$ is the initial message of all honest nodes within $\ddot{\mathcal{A}}_1^{[2]}$.

If all honest nodes keep running the OciorACOOL, COOL-UA[1] and COOL-UA[2] protocols, and given the conclusion that every honest node i eventually inputs the same message $\tilde{w}_i = \bar{w}_1$ to COOL-UA[2], then every honest node i eventually sets $\tilde{\mathbf{s}}_i^{[1]} = 1$ (see Lines 11 and 12 of Algorithm 2), every honest node i eventually sets $\tilde{\mathbf{s}}_i^{[2]} = 1$ (see Lines 20 and 21 of Algorithm 2), and every honest node i eventually sets $\tilde{\mathbf{v}}_i = 1$ (see Lines 25 and 26 of Algorithm 2). In this case, every honest node i eventually delivers $[\tilde{\boldsymbol{w}}^{(i)}, \tilde{\mathbf{s}}_i^{[2]}, \tilde{\mathbf{v}}_i = 1]$ from COOL-UA[2].

Consequently, in this case, every honest node i eventually passes 1 into ABBA as its input message, if node i has not already provided an input to ABBA (see Lines 23 and 24 of Algorithm 1). Hence, ABBA eventually outputs a value $v^* \in \{1,0\}$ at each node, due to the *Termination* property of ABBA. Then, every honest node eventually sends (READY, ID, v^*) to all nodes in Line 28 of Algorithm 1, and every honest node eventually sets $v^* = v^*$ in Line 32 of Algorithm 1. This conclusion contradicts the original assumption that no honest node sets the value of v^* in Line 32. Thus, the original assumption is false, and the statement of this lemma holds for Case I.

• Analysis for Case II:

Let us now consider Case II in (18). In this case, we have

$$|\mathcal{B}^{[2]}| = n - |\mathcal{F}| - |\ddot{\mathcal{A}}_1^{[2]}| \tag{19}$$

$$> n - |\mathcal{F}| - (n - |\mathcal{F}| - t) \tag{20}$$

$$=t$$
 (21)

where (19) follows from (16), while (20) uses the assumption in (18) for this Case II. The result in (21) implies that $|\mathcal{B}^{[2]}| \geq t+1$ in COOL-UA[1]. In this case, the condition $|\mathbb{S}_0^{[2]}| \geq t+1$ in Line 27 of Algorithm 2 is eventually satisfied at all honest nodes, and each honest node i eventually sets $\bar{\mathbf{v}}_i = 0$ and delivers $[\bar{\boldsymbol{w}}^{(i)}, \bar{\mathbf{s}}_i^{[2]}, \bar{\mathbf{v}}_i = 0]$ from COOL-UA[1]. Consequently, each honest node i eventually passes 0 into

ABBA as its input message, if node i has not already provided an input to ABBA (see Lines 25 and 26 of Algorithm 1). In this case, every honest node eventually provides an input to ABBA, and hence ABBA eventually outputs a value $v^* \in \{1,0\}$ at each node, due to the *Termination* property of ABBA. Then, every honest node eventually sends (READY, ID, v^*) to all nodes in Line 28 of Algorithm 1, and every honest node eventually sets $v^* = v^*$ in Line 32 of Algorithm 1. This conclusion contradicts the original assumption that no honest node sets the value of v^* in Line 32. Thus, the original assumption is false, and the statement of this lemma holds for Case II.

Lemma 4. Consider the case where $|\ddot{\mathcal{A}}_1^{[2]}| \geq n - |\mathcal{F}| - t$ for COOL-UA[1], and assume that all honest nodes eventually input their initial messages and keep running the OciorACOOL, COOL-UA[1] and COOL-UA[2] protocols. Under this assumption, every honest node i eventually inputs the same message $\tilde{\mathbf{w}}_i = \bar{\mathbf{w}}_1$ to COOL-UA[2], where $\bar{\mathbf{w}}_1$ is the initial message of all honest nodes within $\ddot{\mathcal{A}}_1^{[2]}$.

Proof. Let us assume $|\ddot{\mathcal{A}}_1^{[2]}| \geq n - |\mathcal{F}| - t$ for COOL-UA[1], and assume that all honest nodes eventually input their initial messages and keep running the OciorACOOL, COOL-UA[1] and COOL-UA[2] protocols. Under these assumptions, we consider each of the following cases for every honest node i:

Case A: Node
$$i$$
 sets $\bar{\mathbf{s}}_i^{[1]} \neq 1$, (22)

Case B: Node
$$i$$
 sets $\bar{\mathbf{s}}_i^{[1]} = 1$ (23)

where we consider the final value of $\bar{\mathbf{s}}_i^{[1]}$ updated in COOL-UA[1]. Under the assumptions considered here, we argue in the following that, for each case, the index of every honest node i is eventually included in $\bar{\mathbb{Y}}_{\text{oec}}$ at each honest node, as in Lines 6 or 12 of Algorithm 1, such that

$$\bar{\mathbb{Y}}_{\text{oec}}[i] = y_i(\bar{\boldsymbol{w}}_1) \quad \forall i \in [n] \setminus \mathcal{F}.$$
 (24)

Here, $y_i(\bar{w}_1)$ denotes the *i*-th coded symbol encoded from the message \bar{w}_1 , and \bar{w}_1 is the initial message of all honest nodes within $\ddot{\mathcal{A}}_1^{[2]}$. With the conclusion in (24), it follows that every honest node eventually decodes the message as \bar{w}_1 from the online error correction procedure in Line 8 of Algorithm 1, and then sets the input of COOL-UA[2] as

$$\tilde{\boldsymbol{w}}_i = \bar{\boldsymbol{w}}_1$$
 (in Line 10 of Algorithm 1). (25)

If an honest node i sets $\bar{\mathbf{s}}_i^{[2]} = 1$ in COOL-UA[1] and COOL-UA[2] has not yet received an input message, then node i sets $\tilde{\boldsymbol{w}}_i = \boldsymbol{w}_i$ as in Line 19 of Algorithm 1. Lemma 10 states that if all honest nodes eventually input their initial messages and keep running the COOL-UA[1] protocol, then in COOL-UA[1] it holds that $\ddot{\eta}^{[2]} \leq 1$. From Lemma 10, if an honest node i sets $\bar{\mathbf{s}}_i^{[2]} = 1$ in COOL-UA[1] and COOL-UA[2] has not yet received an input message, then it follows that $i \in \ddot{\mathcal{A}}_1^{[2]}$ and that node i sets

$$\tilde{\boldsymbol{w}}_i = \bar{\boldsymbol{w}}_1$$
 (in Line 19 of Algorithm 1). (26)

With the results in (25) and (26), it follows that every honest node i eventually inputs the same message $\tilde{w}_i = \bar{w}_1$ to COOL-UA[2] (see Lines 20 and 22 of Algorithm 1). What remains to be proven now is the conclusion in (24).

• Proof of (24) for Case A:

We first consider Case A, where an honest i never sets $\bar{\mathbf{s}}_i^{[1]} = 1$. Lemma 10 shows that if all honest nodes eventually input their initial messages and keep running the COOL-UA[1] protocol, then it holds that $\ddot{\eta}^{[2]} \leq 1$ for COOL-UA[1]. Recall that $\ddot{\mathcal{A}}_1^{[2]} := \{j \in [n] \setminus \mathcal{F} \mid \boldsymbol{w}_j = \bar{\boldsymbol{w}}_1, \; \mathbf{s}_j^{[2]} \neq 0\}$ for some $\bar{\boldsymbol{w}}_1$. From Lemma 10, it follows that the index of each honest node $j \in [n] \setminus (\mathcal{F} \cup \ddot{\mathcal{A}}_1^{[2]})$ is eventually included in the set $\bar{\mathbb{S}}_0^{[2]}$ at node i, i.e.,

$$[n] \setminus (\mathcal{F} \cup \ddot{\mathcal{A}}_1^{[2]}) \subseteq \bar{\mathbb{S}}_0^{[2]}. \tag{27}$$

Furthermore, the index of each honest node $j' \in \ddot{\mathcal{A}}_1^{[2]}$ is eventually included in the set $\mathbb{M}[y_i(\bar{\boldsymbol{w}}_1)]$ at node i (see Lines 14 and 15 of Algorithm 1), i.e.,

$$\ddot{\mathcal{A}}_1^{[2]} \subseteq \mathbb{M}[y_i(\bar{\boldsymbol{w}}_1)]. \tag{28}$$

From (28), and given the condition $|\ddot{\mathcal{A}}_1^{[2]}| \geq n - |\mathcal{F}| - t$ considered here for COOL-UA[1], the following bound eventually holds:

$$|\mathbb{M}[y_i(\bar{\mathbf{w}}_1)]| \ge |\ddot{\mathcal{A}}_1^{[2]}| \ge n - |\mathcal{F}| - t \ge n - 2t.$$
 (29)

Moreover, from (27) and (28), we have that the following bound eventually holds:

$$|\mathbb{M}[y_i(\bar{\boldsymbol{w}}_1)] \cup \bar{\mathbb{S}}_0^{[2]}| \ge |([n] \setminus (\mathcal{F} \cup \ddot{\mathcal{A}}_1^{[2]})) \cup \ddot{\mathcal{A}}_1^{[2]}| = |[n] \setminus \mathcal{F}| = n - |\mathcal{F}| \ge n - t. \tag{30}$$

Therefore, from (29) and (30), if $\bar{\mathbf{s}}_i^{[1]} \neq 1$, then the conditions

$$|\mathbb{M}[y_i(\bar{\boldsymbol{w}}_1)] \cup \bar{\mathbb{S}}_0^{[2]}| \geq n-t, \quad |\mathbb{M}[y_i(\bar{\boldsymbol{w}}_1)]| \geq n-2t, \quad \text{and} \quad \bar{\mathbf{s}}_i^{[1]} \neq 1$$

in Line 16 of Algorithm 1 are all eventually satisfied at the honest node i. In this case, the honest node i eventually sets $\check{y}_i = y_i(\bar{w}_1)$ and then sends (NewSYMBOL, ID, \check{y}_i) to all nodes (see Lines 16 and 17 of Algorithm 1). Therefore, in this case, the index of every honest node i that never sets $\bar{\mathbf{s}}_i^{[1]} = 1$ is eventually included in $\bar{\mathbb{Y}}_{\text{oec}}$ at each honest node, as in Line 6 of Algorithm 1, such that

$$\bar{\mathbb{Y}}_{\text{oec}}[i] = y_i(\bar{\boldsymbol{w}}_1), \quad \text{for } i \in [n] \setminus \mathcal{F}.$$

• Proof of (24) for Case B:

We now consider Case B, where an honest node i eventually sets $\bar{\mathbf{s}}_i^{[1]}=1$. We assume that $|\ddot{\mathcal{A}}_1^{[2]}| \geq n-|\mathcal{F}|-t$ for COOL-UA[1], and that all honest nodes eventually input their initial messages and keep running the OciorACOOL and COOL-UA[1] protocols. Under these assumptions, from Lemma 6, if an honest node i sets $\bar{\mathbf{s}}_i^{[1]}=1$ and sends (SYMBOL, $1,(*,\bar{y}_i^{(i)})$) in COOL-UA[1], then

$$\bar{y}_i^{(i)} = y_i(\bar{\boldsymbol{w}}_1).$$

Here, $\bar{\boldsymbol{w}}_1$ is the initial message of all honest nodes within $\ddot{\mathcal{A}}_1^{[2]}$. Note that under the assumptions considered here, every honest node i eventually sends (SYMBOL, $1, (*, \bar{y}_i^{(i)})$) in COOL-UA[1]. Therefore, under the same assumptions, every honest node i that sets $\bar{\mathbf{s}}_i^{[1]} = 1$ is eventually included in $\bar{\mathbb{Y}}_{\text{oec}}$ at each honest node, as in Line 12 of Algorithm 1 (or in Line 6, as in Case A), such that

$$\bar{\mathbb{Y}}_{\text{oec}}[i] = y_i(\bar{\boldsymbol{w}}_1), \quad \text{for } i \in [n] \setminus \mathcal{F}.$$

This completes the proof.

Lemma 5. Consider the case where $|\ddot{\mathcal{A}}_1^{[2]}| \geq n - |\mathcal{F}| - t$ for COOL-UA[1], and assume that all honest nodes eventually input their initial messages and keep running the OciorACOOL and COOL-UA[1] protocols. Under this assumption, if an honest node i sets $\check{y}_i \leftarrow y^*$ in Line 17 of Algorithm 1, then $y^* = y_i(\bar{w}_1)$. Here, $y_i(\bar{w}_1)$ is the i-th coded symbol encoded from the message \bar{w}_1 , while \bar{w}_1 is the initial message of all honest nodes within $\ddot{\mathcal{A}}_1^{[2]}$.

Proof. Let us now we consider the case where

$$|\ddot{\mathcal{A}}_1^{[2]}| \ge n - |\mathcal{F}| - t \tag{31}$$

for COOL-UA[1], and assume that all honest nodes eventually input their initial messages and keep running the OciorACOOL and COOL-UA[1] protocols. Here, $\ddot{\mathcal{A}}_1^{[2]} := \{j \in [n] \setminus \mathcal{F} \mid \boldsymbol{w}_j = \bar{\boldsymbol{w}}_1, \ \mathbf{s}_i^{[2]} \neq 0 \}$ for some

 \bar{w}_1 . Under this assumption, if an honest node i sets $\check{y}_i \leftarrow y^*$ in Line 17 of Algorithm 1, then the conditions in Line 16 must be satisfied, i.e.,

$$|\mathbb{M}[y^*] \cup \bar{\mathbb{S}}_0^{[2]}| \ge n - t, \quad \text{and} \quad |\mathbb{M}[y^*]| \ge n - 2t \tag{32}$$

for some y^\star . Here, $\mathbb{M}[y^\star]$ includes the indices of nodes that sent (SYMBOL, $1, (\bar{y}_i^{(j)}, *)$) to node i in COOL-UA[1] such that $\bar{y}_i^{(j)} = y^\star$ (see Lines 14 and 15 of Algorithm 1). The set $\bar{\mathbb{S}}_0^{[2]}$ includes the indices of nodes that sent $\mathbf{s}_j^{[2]} = 0$. Since each honest node $j \in \ddot{\mathcal{A}}_1^{[2]}$ never sends $\mathbf{s}_j^{[2]} = 0$, it follows that

$$\ddot{\mathcal{A}}_1^{[2]} \cap \bar{\mathbb{S}}_0^{[2]} = \emptyset. \tag{33}$$

On the other hand, it holds true that

$$|\ddot{\mathcal{A}}_1^{[2]} \cup \{\mathbb{M}[y^*] \cup \bar{\mathbb{S}}_0^{[2]}\}| \le n.$$
 (34)

Given the conditions in (31)-(34), we now argue that $\ddot{\mathcal{A}}_1^{[2]} \cap \mathbb{M}[y^\star] \neq \emptyset$. We prove this result by contradiction and assume, for the sake of contradiction, that $\ddot{\mathcal{A}}_1^{[2]} \cap \mathbb{M}[y^\star] = \emptyset$. Specifically, under the assumption $\ddot{\mathcal{A}}_1^{[2]} \cap \mathbb{M}[y^\star] = \emptyset$ and given the condition $\ddot{\mathcal{A}}_1^{[2]} \cap \bar{\mathbb{S}}_0^{[2]} = \emptyset$ in (33), we have the following bound:

$$|\ddot{\mathcal{A}}_{1}^{[2]} \cup \{\mathbb{M}[y^{\star}] \cup \bar{\mathbb{S}}_{0}^{[2]}\}| = \underbrace{|\ddot{\mathcal{A}}_{1}^{[2]}|}_{\geq n - |\mathcal{F}| - t} + \underbrace{|\mathbb{M}[y^{\star}] \cup \bar{\mathbb{S}}_{0}^{[2]}|}_{\geq n - t} - \underbrace{|\ddot{\mathcal{A}}_{1}^{[2]} \cap \{\mathbb{M}[y^{\star}] \cup \bar{\mathbb{S}}_{0}^{[2]}\}|}_{=0}$$
(35)

$$\geq (n - |\mathcal{F}| - t) + (n - t) \tag{36}$$

$$\geq n - t - t + 3t + 1 - t \tag{37}$$

$$> n+1$$
 (38)

where (36) follows from the inequalities in (31) and (32), as well as the identity

$$|\ddot{\mathcal{A}}_1^{[2]} \cap \{\mathbb{M}[y^*] \cup \bar{\mathbb{S}}_0^{[2]}\}| = 0$$

under the assumption $\ddot{\mathcal{A}}_1^{[2]}\cap\mathbb{M}[y^\star]=\emptyset$ and given the condition $\ddot{\mathcal{A}}_1^{[2]}\cap\bar{\mathbb{S}}_0^{[2]}=\emptyset$ in (33). Here (37) uses the assumptions $n\geq 3t+1$ and $|\mathcal{F}|\leq t$. One can see that the conclusion in (38) contradicts the result in (34). Therefore, the original assumption $\ddot{\mathcal{A}}_1^{[2]}\cap\mathbb{M}[y^\star]=\emptyset$ is false, and the statement $\ddot{\mathcal{A}}_1^{[2]}\cap\mathbb{M}[y^\star]\neq\emptyset$ holds true. The conclusion $\ddot{\mathcal{A}}_1^{[2]}\cap\mathbb{M}[y^\star]\neq\emptyset$ implies that there exists at least one honest node $j\in\ddot{\mathcal{A}}_1^{[2]}$ whose index also belongs to $\mathbb{M}[y^\star]$. In other words, node j has sent (SYMBOL, $1,(\bar{y}_i^{(j)},*)$) to node i in COOL-UA[1] with $\bar{y}_i^{(j)}=y_i(\bar{w}_1)=y^\star$ (see Lines 14 and 15 of Algorithm 1). Consequently, if an honest node i sets $\check{y}_i\leftarrow y^\star$ in Line 17 of Algorithm 1, then $y^\star=y_i(\bar{w}_1)$.

Lemma 6. Consider the case where $|\ddot{\mathcal{A}}_1^{[2]}| \geq n - |\mathcal{F}| - t$ for COOL-UA[1], and assume that all honest nodes eventually input their initial messages and keep running the OciorACOOL and COOL-UA[1] protocols. Under this assumption, if an honest node i sets $\mathbf{s}_i^{[1]} = 1$ and sends (SYMBOL, $1, (*, \bar{y}_i^{(i)})$) in COOL-UA[1], then $\bar{y}_i^{(i)} = y_i(\bar{\mathbf{w}}_1)$. Here, $\bar{\mathbf{w}}_1$ is the initial message of all honest nodes within $\ddot{\mathcal{A}}_1^{[2]}$.

Proof. Similar to the proof of Lemma 5, we also consider the case where

$$|\ddot{\mathcal{A}}_1^{[2]}| \ge n - |\mathcal{F}| - t \tag{39}$$

for COOL-UA[1], and assume that all honest nodes eventually input their initial messages and continue executing the OciorACOOL and COOL-UA[1] protocols. Suppose further that there exists an honest node i, where $i \in \mathcal{A}_{l^{\star}}$ for some $l^{\star} \in [\eta]$, such that node i sets $\mathbf{s}_{j}^{[1]} = 1$ and sends (SYMBOL, $1, (*, \bar{y}_{i}^{(i)})$) in COOL-UA[1]. We now argue that $\bar{y}_{i}^{(i)} = y_{i}(\bar{w}_{1})$. Specifically, when an honest node $i \in \mathcal{A}_{l^{\star}}$ sets $\mathbf{s}_{i}^{[1]} = 1$, it implies that at least n-t nodes in \mathbb{U}_{1} have sent (SYMBOL, $1, (\bar{y}_{i}^{(j)}, *)$) to node i in COOL-UA[1] with

$$\bar{y}_i^{(j)} = \bar{y}_i^{(i)}, \quad \forall j \in \mathbb{U}_1 \tag{40}$$

(see Lines 9 and 12 of Algorithm 2), where

$$|\mathbb{U}_1| \ge n - t. \tag{41}$$

On the other hand, it holds true that

$$|\ddot{\mathcal{A}}_1^{[2]} \cup \mathbb{U}_1| \le n. \tag{42}$$

Given the conditions in (39)-(42), we argue that $\ddot{\mathcal{A}}_1^{[2]} \cap \mathbb{U}_1 \neq \emptyset$. This proof is similar to that of Lemma 5. Specifically, if $\ddot{\mathcal{A}}_1^{[2]} \cap \mathbb{U}_1 = \emptyset$, then we have

$$|\ddot{\mathcal{A}}_{1}^{[2]} \cup \mathbb{U}_{1}| = \underbrace{|\ddot{\mathcal{A}}_{1}^{[2]}|}_{>n-|\mathcal{F}|-t} + \underbrace{|\mathbb{U}_{1}|}_{\geq n-t} - \underbrace{|\ddot{\mathcal{A}}_{1}^{[2]} \cap \mathbb{U}_{1}|}_{=0} \ge n+1 \tag{43}$$

which contradicts the result in (42). Therefore, the statement $\ddot{\mathcal{A}}_1^{[2]} \cap \mathbb{U}_1 \neq \emptyset$ holds true. The conclusion $\ddot{\mathcal{A}}_1^{[2]} \cap \mathbb{U}_1 \neq \emptyset$ implies that there exists at least one honest node $j \in \ddot{\mathcal{A}}_1^{[2]}$ whose index also belongs to \mathbb{U}_1 such that the condition $\bar{y}_i^{(j)} = \bar{y}_i^{(i)}$ in (40) is satisfied. In other words, node $j \in \ddot{\mathcal{A}}_1^{[2]}$ has sent (SYMBOL, $1, (\bar{y}_i^{(j)}, *)$) to node i in COOL-UA[1] with $\bar{y}_i^{(j)} = y_i(\bar{w}_1) = \bar{y}_i^{(i)}$. Consequently, in this case, if an honest node i sets $\mathbf{s}_i^{[1]} = 1$ and sends (SYMBOL, $1, (*, \bar{y}_i^{(i)})$) in COOL-UA[1], then $\bar{y}_i^{(i)} = y_i(\bar{w}_1)$. \square

Theorem 4 (Communication, Round, Resilience, and Computation). OciorACOOL achieves the ABA consensus with total $O(\max\{n\ell, nt \log q\})$ communication bits, O(1) rounds, and a single invocation of a binary BA protocol, under the optimal resilience assumption $n \geq 3t + 1$. Moreover, the computation complexity of OciorACOOL is $\tilde{O}(\ell+t)$ in the good case and $\tilde{O}(t\ell+t^2)$ in the worst case, measured in bit-level operations per node, where the good case occurs when the actual number of Byzantine nodes is small or the network is nearly synchronous.

Proof. From Theorems 1-3, it follows that the proposed OciorACOOL protocol satisfies the Consistency, Validity, and Termination properties in all executions, under the assumption of an asynchronous binary BA protocol invoked within it, given that $n \ge 3t + 1$.

Regarding the analysis of communication complexity, we first focus on the case where $t = \Omega(n)$. Recall that the size of the message to be agreed upon is ℓ bits. In OciorACOOL, each coded symbol carries

$$c = \max\left\{\frac{\ell}{k}, \log q\right\}$$

bits, where k=t/3, and q denotes the alphabet size of the error correction code used in the protocol. The total communication complexity of OciorACOOL is computed as

Total Comm. =
$$O(cn^2 + n^2)$$

= $O(\max\{\ell n, n^2 \log q\})$ bits. (44)

In the above communication complexity analysis, we do not include the communication cost of the binary BA protocol, which is assumed to have a total communication complexity bounded by $O(n^2)$ bits and an expected O(1) number of rounds.

Similar to COOL, when t is very small compared to n, we first select n' := 3t + 1 nodes (e.g., the first n' nodes), denoted by the set S', from the n nodes to run the OciorACOOL protocol (see OciorACOOL protocol in Algorithm 3). In this setting, each coded symbol still carries c bits. After the OciorACOOL protocol reaches agreement on a message w' within S', the i-th node in S' sends a coded symbol y'_i , encoded from the agreed message w', to the n - n' nodes outside S', where the symbols are encoded as

$$[y'_1, y'_2, \dots, y'_{n'}] \leftarrow \text{ECCEnc}(n', k, \boldsymbol{w}').$$

Each node outside S' eventually decodes the agreed message w' using online error-correction decoding from the symbols $\{y_i'\}_{i\in S'}$ collected from the nodes within S' (see Algorithm 3).

Note that the OciorACOOL protocol executed within S' satisfies the Consistency, Validity, and Termination properties, given n' = 3t + 1. Furthermore, based on the decoding property of the error correction code, the overall protocol OciorACOOL* (see Algorithm 3)–comprising OciorACOOL within S' and the subsequent Strongly-Honest-Majority Distributed Multicast (SHMDM) from <math>S'-also satisfies the Consistency, Validity, and Termination properties. In this case, the total communication complexity of the overall protocol OciorACOOL* is computed as

Total Comm. =
$$O(cn' \cdot n' + n' \cdot n' + cn'(n - n'))$$

= $O(\max\{\ell n, nt \log q\})$ bits. (45)

Thus, by combining the result in (44) for the case $t = \Omega(n)$ and the result in (45) for the case of very small t, the total communication complexity of OciorACOOL can be expressed as

Total Comm. =
$$O(\max\{n\ell, nt \log q\})$$
 bits.

The proposed OciorACOOL protocol uses O(1) rounds and a single invocation of a binary BA protocol that is assumed to have an expected O(1) number of rounds.

The computation complexity of the proposed OciorACOOL is nominated by the online error-correction decoding in Lines 7-10 and 45-48 of Algorithm 1. Fast (n,k) Reed-Solomon decoding and polynomial error-correction decoding of degree-(k-1) polynomials from \bar{n} evaluation points (with $\bar{n} \leq n$) are achievable in

$$O(\bar{n}\log^2 \bar{n}\log\log \bar{n})$$

field operations using Fast Fourier Transform (FFT)-based polynomial arithmetic over the finite field \mathbb{F}_q [23], [26]–[28].

If each symbol is represented using $c = \max\left\{\frac{\ell}{k}, \log q\right\}$ bits, where k = t/3, then multiplication of two symbols can be done in $O(c\log c\log\log c)$ bit-level operations using fast multiplication algorithms, while inversion or division can be done in $O(c\log^2 c\log\log c)$ bit-level operations using fast inversion algorithms. Therefore, fast (n,k) Reed-Solomon decoding of the message w can be achieved in

$$O((\bar{n}\log^2 \bar{n}\log\log \bar{n}) \cdot (c\log^2 c\log\log c)) = \tilde{O}(\bar{n}c) = \tilde{O}(\frac{\bar{n}\ell}{k} + \bar{n}\log q) = \tilde{O}(\ell + \bar{n}\log q)$$

bit-level operations. For the Reed-Solomon codes, q can be set such that $q \ge n+1$. In OciorACOOL, for the error-correction decoding in Lines 7-10 and 45-48 of Algorithm 1, \bar{n} is set to O(t), and the decoding may be repeated up to t times. Hence, the computational complexity of the proposed OciorACOOL in the worst case is

$$t \cdot \tilde{O}(\ell + \bar{n} \log q) = t \cdot \tilde{O}(\ell + t \log n) = \tilde{O}(t\ell + t^2)$$

bit-level operations per node. The computation complexity of OciorACOOL in the good case is

$$\tilde{O}(\ell + t)$$

bit-level operations per node, where the good case occurs when the actual number of Byzantine nodes is small or the network is nearly synchronous. \Box

E. Proofs of Lemma 10

In the following, we present Lemma 10, which is used in the above analysis. First, we provide several lemmas that will be used in the proof of Lemma 10. Lemma 10 extends the result of [5, Lemma 3] (see Lemma 9 below).

Lemma 7. [4, Lemma 7] For $\eta \geq 2$, it is true that

$$|\mathcal{A}_{l,j}| + |\mathcal{A}_{j,l}| < k, \quad \forall j \neq l, \ j,l \in [\eta]$$

$$\tag{46}$$

where k is a parameter of (n, k) error correction code.

Lemma 8. [5, Lemma 7] [5, Lemma 16] For the COOL-UA protocol with $n \ge 3t + 1$ and $k \le t/3$, if $\eta^{[1]} = 2$ then it holds true that $\eta^{[2]} \leq 1$.

Lemma 9. [5, Lemma 3] [5, Lemma 11] For the COOL-UA protocol with $n \ge 3t + 1$, it holds true that $\eta^{[2]} < 1.$

Lemma 10. For the COOL-UA protocol with $n \geq 3t + 1$ and $k \leq t/3$, it holds true that $\eta^{[2]} \leq 1$. Furthermore, if all honest nodes eventually input their initial messages and keep running the COOL-UA protocol, then it holds true that $\ddot{\eta}^{[2]} \leq 1$, i.e., every honest node i eventually sets $s_i^{[2]} = 0$ for all $i \in A_l$ and for all $l \in [2, \eta]$.

Proof. We first prove the first statement. For this statement, each honest node can terminate at any point in time, and some nodes may not have input their initial messages before termination. From Lemma 11, it follows that $\eta^{[1]} \leq 2$. Then, from Lemma 8, if $\eta^{[1]} = 2$, it holds that $\eta^{[2]} \leq 1$. If $\eta^{[1]} \leq 1$, it follows immediately that $\eta^{[2]} < 1$.

We now prove the second statement. From Lemma 11, if all honest nodes eventually input their initial messages and keep running the COOL-UA protocol, then $\ddot{\eta}^{[1]} < 2$. Then, from Lemma 12, if all honest nodes eventually input their initial messages and keep running the COOL-UA protocol, and if $\ddot{\eta}^{[1]} = 2$, it follows that $\ddot{\eta}^{[2]} < 1$. If $\ddot{\eta}^{[1]} < 1$, it follows immediately that $\ddot{\eta}^{[2]} < 1$. This completes the proof.

Lemma 11. For the COOL-UA protocol, it is true that $\eta^{[1]} \leq 2$. Furthermore, if all honest nodes eventually input their initial messages and keep running the COOL-UA protocol, then it is also true that $\ddot{\eta}^{[1]} \leq 2$.

 $\textit{Proof.} \ \ \text{Recall that} \ \ \mathbb{U}_{1}^{(i)} \ := \ \{j \in [n] \ \mid \ (y_{i}^{(j)}, y_{j}^{(j)}) \ = \ (y_{i}^{(i)}, y_{j}^{(i)}) \} \ \ \text{and} \ \ \mathbb{U}_{0}^{(i)} \ := \ \{j \in [n] \ \mid \ (y_{i}^{(j)}, y_{j}^{(j)}) \ \neq \ \ \text{ond} \ \ \mathbb{U}_{0}^{(i)} \ := \ \{j \in [n] \ \mid \ (y_{i}^{(j)}, y_{j}^{(j)}) \ \neq \ \ \text{ond} \ \ \mathbb{U}_{0}^{(i)} \ := \ \{j \in [n] \ \mid \ (y_{i}^{(j)}, y_{j}^{(j)}) \ \neq \ \ \text{ond} \ \ \mathbb{U}_{0}^{(i)} \ := \ \{j \in [n] \ \mid \ (y_{i}^{(i)}, y_{j}^{(i)}) \ \neq \ \ \text{ond} \ \ \mathbb{U}_{0}^{(i)} \ := \ \{j \in [n] \ \mid \ (y_{i}^{(i)}, y_{j}^{(i)}) \ \neq \ \ \text{ond} \ \ \mathbb{U}_{0}^{(i)} \ := \ \{j \in [n] \ \mid \ (y_{i}^{(i)}, y_{j}^{(i)}) \ \neq \ \ \mathbb{U}_{0}^{(i)} \ := \ \{j \in [n] \ \mid \ (y_{i}^{(i)}, y_{j}^{(i)}) \ \neq \ \ \mathbb{U}_{0}^{(i)} \ := \ \{j \in [n] \ \mid \ (y_{i}^{(i)}, y_{j}^{(i)}) \ \neq \ \ \mathbb{U}_{0}^{(i)} \ := \ \{j \in [n] \ \mid \ (y_{i}^{(i)}, y_{j}^{(i)}) \ \neq \ \ \mathbb{U}_{0}^{(i)} \ := \ \{j \in [n] \ \mid \ (y_{i}^{(i)}, y_{j}^{(i)}) \ \neq \ \ \mathbb{U}_{0}^{(i)} \ := \ \{j \in [n] \ \mid \ (y_{i}^{(i)}, y_{j}^{(i)}) \ = \ \ \mathbb{U}_{0}^{(i)} \ := \ \{j \in [n] \ \mid \ (y_{i}^{(i)}, y_{j}^{(i)}) \ = \ \ \mathbb{U}_{0}^{(i)} \ := \ \{j \in [n] \ \mid \ (y_{i}^{(i)}, y_{j}^{(i)}) \ = \ \ \mathbb{U}_{0}^{(i)} \ := \ \{j \in [n] \ \mid \ (y_{i}^{(i)}, y_{j}^{(i)}) \ = \ \ \mathbb{U}_{0}^{(i)} \ := \ \mathbb{U}_{0}^{(i)} \ :=$ $(y_i^{(i)}, y_j^{(i)})$ } denote the link indicator sets \mathbb{U}_1 and \mathbb{U}_0 updated by Node i, as described in Line 9 of Algorithm 2 (see (13)). Here we define $\mathbb{V}_1^{(i)} := \mathbb{U}_1^{(i)} \setminus \mathcal{F}$ and $\mathbb{V}_0^{(i)} := \mathbb{U}_0^{(i)} \setminus \mathcal{F}$ (see (14)). Recall that $\mathcal{A}_l := \{i \in [n] \setminus \mathcal{F} \mid \boldsymbol{w}_i = \bar{\boldsymbol{w}}_l\}$ for $l \in [\eta]$ and for some non-empty ℓ -bit distinct values $\bar{\boldsymbol{w}}_1, \bar{\boldsymbol{w}}_2, \cdots, \bar{\boldsymbol{w}}_{\eta}$. Let us first prove the second statement and then the first of this lemma.

• Proof for $\ddot{\eta}^{[1]} \leq 2$, if all honest nodes eventually input messages and keep running COOL-UA: Let us first assume that all honest nodes eventually input their initial messages and keep running the COOL-UA protocol, and that there already exist an honest Node $i_1 \in \mathcal{A}_1$ and an honest Node $i_2 \in \mathcal{A}_2$ such that they *never* set $\mathbf{s}_{i_1}^{[1]} = 0$ and $\mathbf{s}_{i_2}^{[1]} = 0$, respectively. We now argue that each honest Node $i_3 \in \mathcal{A}_l$, for all $l \in [3, \eta]$, will eventually set $\mathbf{s}_{i_3}^{[1]} = 0$. Given $\mathbf{s}_{i_1}^{[1]} \neq 0$ and $\mathbf{s}_{i_2}^{[1]} \neq 0$, we have the following bounds on $|\mathbb{V}_0^{(i_1)}|$ and $|\mathbb{V}_0^{(i_2)}|$:

$$|\mathbb{V}_0^{(i_1)}| \le t$$
, and $|\mathbb{V}_0^{(i_2)}| \le t$. (47)

If all honest nodes eventually input their initial messages and keep running the COOL-UA protocol, then for each $i \in [n] \setminus \mathcal{F}$, eventually the following equality holds true:

$$|\mathbb{V}_{1}^{(i)}| + |\mathbb{V}_{0}^{(i)}| = n - |\mathcal{F}|. \tag{48}$$

Then, the above results in (47) and (48) imply:

$$|\mathbb{V}_{1}^{(i_{1})}| \ge n - t - |\mathcal{F}|, \quad \text{and} \quad |\mathbb{V}_{1}^{(i_{2})}| \ge n - t - |\mathcal{F}|.$$
 (49)

Furthermore, for each $j \in \mathbb{V}_1^{(i_1)} \cap \mathbb{V}_1^{(i_2)}$, it is true that $\boldsymbol{h}_j^{\mathsf{T}} \bar{\boldsymbol{w}}_1 = \boldsymbol{h}_j^{\mathsf{T}} \boldsymbol{w}_j$ and $\boldsymbol{h}_j^{\mathsf{T}} \bar{\boldsymbol{w}}_2 = \boldsymbol{h}_j^{\mathsf{T}} \boldsymbol{w}_j$ (see Line 9 of Algorithm 2), which implies that $\boldsymbol{h}_j^{\mathsf{T}} \bar{\boldsymbol{w}}_1 = \boldsymbol{h}_j^{\mathsf{T}} \bar{\boldsymbol{w}}_2$, $\forall j \in \mathbb{V}_1^{(i_1)} \cap \mathbb{V}_1^{(i_2)}$. Thus, we conclude:

$$|\mathbb{V}_{1}^{(i_{1})} \cap \mathbb{V}_{1}^{(i_{2})}| \le k - 1,\tag{50}$$

otherwise $\bar{w}_1 = \bar{w}_2$ due to a property of linear algebra, which contradicts our assumption $\bar{w}_1 \neq \bar{w}_2$. Note that if there exists a full-rank matrix H of size $k \times k$ such that Hx = 0, then it follows that x = 0.

Similarly, we have

$$|\mathbb{V}_{1}^{(i_{1})} \cap \mathbb{V}_{1}^{(i_{3})}| \le k - 1, \quad \text{and} \quad |\mathbb{V}_{1}^{(i_{2})} \cap \mathbb{V}_{1}^{(i_{3})}| \le k - 1.$$
 (51)

In our setting, the following identities hold true:

$$|\mathbb{V}_{1}^{(i_{1})} \cup \mathbb{V}_{1}^{(i_{2})} \cup \mathbb{V}_{1}^{(i_{3})}| \le n - |\mathcal{F}|,\tag{52}$$

$$|\mathbb{V}_{1}^{(i_{1})} \cup \mathbb{V}_{1}^{(i_{2})} \cup \mathbb{V}_{1}^{(i_{3})}| \geq |\mathbb{V}_{1}^{(i_{1})}| + |\mathbb{V}_{2}^{(i_{2})}| + |\mathbb{V}_{1}^{(i_{3})}| - |\mathbb{V}_{1}^{(i_{1})} \cap \mathbb{V}_{1}^{(i_{2})}| - |\mathbb{V}_{1}^{(i_{1})} \cap \mathbb{V}_{1}^{(i_{3})}| - |\mathbb{V}_{1}^{(i_{2})} \cap \mathbb{V}_{1}^{(i_{3})}|, \quad (53)$$

where the first identity follows from the fact $\mathbb{V}_1^{(i_1)} \cup \mathbb{V}_1^{(i_2)} \cup \mathbb{V}_1^{(i_3)} \subseteq [n] \setminus \mathcal{F}$ and the second identity follows from the inclusion-exclusion identity for the union of three sets. At this point, $|\mathbb{V}_1^{(i_3)}|$ can be bounded by:

$$|\mathbb{V}_{1}^{(i_{3})}| \leq |\mathbb{V}_{1}^{(i_{1})} \cup \mathbb{V}_{1}^{(i_{2})} \cup \mathbb{V}_{1}^{(i_{3})}| - |\mathbb{V}_{1}^{(i_{1})}| - |\mathbb{V}_{2}^{(i_{1})}| + |\mathbb{V}_{1}^{(i_{1})} \cap \mathbb{V}_{1}^{(i_{2})}| + |\mathbb{V}_{1}^{(i_{1})} \cap \mathbb{V}_{1}^{(i_{3})}| + |\mathbb{V}_{1}^{(i_{2})} \cap \mathbb{V}_{1}^{(i_{3})}|$$
 (54)

$$\leq n - |\mathcal{F}| - 2(n - t - |\mathcal{F}|) + 3(k - 1)$$
 (55)

$$\leq n - |\mathcal{F}| - 2 \cdot (3t + 1 - t - t) + 3 \cdot (t/3 - 1) \tag{56}$$

$$= n - |\mathcal{F}| - t - 5 \tag{57}$$

$$< n - |\mathcal{F}| - t \tag{58}$$

where the first inequality follows from the identity in (53); the second inequality follows from the results in (49)-(52); and the third inequality uses the identities $n \ge 3t + 1$, $|\mathcal{F}| \le t$ and $k \le t/3$.

Since $\mathbb{V}_0^{(i)} \subseteq \mathbb{U}_0^{(i)}$, and from (48) and (57), if all honest nodes eventually input their initial messages and keep running the COOL-UA protocol, then we have

$$|\mathbb{U}_0^{(i_3)}| \ge |\mathbb{V}_0^{(i_3)}|$$

$$= n - |\mathcal{F}| - |\mathbb{V}_1^{(i_3)}| \tag{59}$$

$$\geq n - |\mathcal{F}| - (n - |\mathcal{F}| - t - 5) \tag{60}$$

$$> t+1$$
 (61)

where (59) follows from (48); and (60) is from (57). The result in (61) reveals that, if all honest nodes eventually input their initial messages and keep running the COOL-UA protocol, then each honest Node $i_3 \in \mathcal{A}_l$, for all $l \in [3, \eta]$, will eventually set $\mathbf{s}_{i_3}^{[1]} = 0$. Thus, it is true that $\ddot{\eta}^{[1]} \leq 2$.

• Proof for $\eta^{[1]} \leq 2$:

Let us assume that there already exist an honest node $i_1 \in \mathcal{A}_1$ and an honest node $i_2 \in \mathcal{A}_2$ such that they have set $\mathbf{s}_{i_1}^{[1]} = 1$ and $\mathbf{s}_{i_2}^{[1]} = 1$, respectively, in COOL-UA protocol. In this case, we assume that each honest node can terminate at any point in time, and that some nodes may not have input their initial messages before termination. We now argue that every honest node $i_3 \in \mathcal{A}_l$, for all $l \in [3, \eta]$, will never set $\mathbf{s}_{i_3}^{[1]} = 1$.

Under the assumption of $s_{i_1}^{[1]} = 1$ and $s_{i_2}^{[1]} = 1$, and from the condition in Line 11 of Algorithm 2, the following inequities hold true:

$$|\mathbb{U}_1^{(i_1)}| \ge n - t$$
, and $|\mathbb{U}_1^{(i_2)}| \ge n - t$. (62)

Then, the above result implies:

$$|\mathbb{V}_{1}^{(i_{1})}| \ge n - t - |\mathcal{F}|, \quad \text{and} \quad |\mathbb{V}_{1}^{(i_{2})}| \ge n - t - |\mathcal{F}|.$$
 (63)

One can check that, under this assumption that $\mathbf{s}_{i_1}^{[1]}=1$ and $\mathbf{s}_{i_2}^{[1]}=1$, the results in (49)-(58) hold true and it is concluded that

$$|\mathbb{V}_1^{(i_3)}| < n - |\mathcal{F}| - t \tag{64}$$

(see (58)). This result reveals that each honest Node $i_3 \in \mathcal{A}_l$, for all $l \in [3, \eta]$, will never set $\mathbf{s}_{i_3}^{[1]} = 1$. Thus, it is true that $\eta^{[1]} \leq 2$. This completes the proof.

Lemma 12. For the COOL-UA protocol with $n \ge 3t+1$ and $k \le t/3$, and assuming that all honest nodes eventually input their initial messages and keep running the COOL-UA protocol, if $\ddot{\eta}^{[1]} = 2$ then it holds true that $\ddot{\eta}^{[2]} \le 1$.

Proof. This proof follows closely that of [5, Lemma 7] [5, Lemma 16]. Here we assume that all honest nodes eventually input their initial messages and keep running the COOL-UA protocol. We also assume that $\ddot{\eta}^{[1]} = 2$. Under this assumption, the definition in (1)-(11) suggests that

$$\ddot{\mathcal{A}}_{1}^{[1]} := \{ i \in [n] \setminus \mathcal{F} \mid \mathbf{w}_{i} = \bar{\mathbf{w}}_{1}, \ \mathbf{s}_{i}^{[1]} \neq 0 \},$$
 (65)

$$\ddot{\mathcal{A}}_{2}^{[1]} := \{ i \in [n] \setminus \mathcal{F} \mid \mathbf{w}_{i} = \bar{\mathbf{w}}_{2}, \ \mathbf{s}_{i}^{[1]} \neq 0 \},$$
 (66)

$$\mathcal{B}^{[1]} := \{ i \in [n] \setminus \mathcal{F} \mid \mathbf{s}_i^{[1]} = 0 \}$$
(67)

$$\ddot{\mathcal{A}}_{2,1}^{[1]} := \{ i \in \ddot{\mathcal{A}}_{2}^{[1]} \mid \mathbf{h}_{i}^{\mathsf{T}} \bar{\mathbf{w}}_{2} = \mathbf{h}_{i}^{\mathsf{T}} \bar{\mathbf{w}}_{1} \} \tag{68}$$

$$\ddot{\mathcal{A}}_{2,2}^{[1]} := \ddot{\mathcal{A}}_{2}^{[1]} \setminus \ddot{\mathcal{A}}_{2,1}^{[1]} = \{ i \in \ddot{\mathcal{A}}_{2}^{[1]} \mid \mathbf{h}_{i}^{\mathsf{T}} \bar{\mathbf{w}}_{2} \neq \mathbf{h}_{i}^{\mathsf{T}} \bar{\mathbf{w}}_{1} \}$$

$$(69)$$

$$\ddot{\mathcal{A}}_{1,2}^{[1]} := \{ i \in \ddot{\mathcal{A}}_{1}^{[1]} \mid \mathbf{h}_{i}^{\mathsf{T}} \bar{\mathbf{w}}_{1} = \mathbf{h}_{i}^{\mathsf{T}} \bar{\mathbf{w}}_{2} \}$$
(70)

$$\ddot{\mathcal{A}}_{1,1}^{[1]} := \ddot{\mathcal{A}}_{1}^{[1]} \setminus \ddot{\mathcal{A}}_{1,2}^{[1]} = \{ i \in \ddot{\mathcal{A}}_{1}^{[1]} \mid \boldsymbol{h}_{i}^{\mathsf{T}} \bar{\boldsymbol{w}}_{1} \neq \boldsymbol{h}_{i}^{\mathsf{T}} \bar{\boldsymbol{w}}_{2} \} \tag{71}$$

Given the identity $|\mathcal{A}_1| + |\mathcal{A}_2| = n - |\mathcal{F}| - \sum_{l=3}^{\eta} |\mathcal{A}_l|$, we will consider each of the following two cases:

Case 1:
$$|A_2| \le \frac{n - |\mathcal{F}| - \sum_{l=3}^{\eta} |A_l|}{2}$$
, (72)

Case 2:
$$|\mathcal{A}_1| \le \frac{n - |\mathcal{F}| - \sum_{l=3}^{\eta} |\mathcal{A}_l|}{2}$$
. (73)

• Analysis for Case 1:

We first consider Case 1. Under the assumption that all honest nodes eventually input their initial messages and keep running the COOL-UA protocol, for each Node $i \in \ddot{\mathcal{A}}_2^{[1]}$, it is true that

$$\mathcal{B}^{[1]} \in \mathbb{S}_0^{[1]} \cup \mathbb{U}_0,\tag{74}$$

$$\ddot{\mathcal{A}}_{1,1}^{[1]} \in \mathbb{S}_0^{[1]} \cup \mathbb{U}_0, \tag{75}$$

and that

$$|\mathbb{S}_{0}^{[1]} \cup \mathbb{U}_{0}| \geq |\mathcal{B}^{[1]} \cup \ddot{\mathcal{A}}_{1,1}^{[1]}|$$

$$= n - |\mathcal{F}| - (|\ddot{\mathcal{A}}_{1,2}^{[1]}| + |\ddot{\mathcal{A}}_{2}^{[1]}|)$$

$$= n - \underbrace{|\mathcal{F}|}_{\leq t} - (\underbrace{|\ddot{\mathcal{A}}_{1,2}^{[1]}| + |\ddot{\mathcal{A}}_{2,1}^{[1]}|}_{\leq k-1} + \underbrace{|\ddot{\mathcal{A}}_{2,2}^{[1]}|}_{\leq 2(k-1)})$$

$$\geq n - t - 3(k-1)$$

$$> t + 1. \tag{77}$$

Here, (76) follows from Lemma 13 and the identity that $|\ddot{\mathcal{A}}_{1,2}^{[1]}| + |\ddot{\mathcal{A}}_{2,1}^{[1]}| \leq |\mathcal{A}_{1,2}| + |\mathcal{A}_{2,1}| \leq k-1$ (see Lemma 7), while (77) uses the assumption of $n \geq 3t+1$ and $k \leq t/3$. The result $|\mathbb{S}_0^{[1]} \cup \mathbb{U}_0| > t+1$ in (77) reveals that each Node $i \in \ddot{\mathcal{A}}_2^{[1]}$ eventually sets $\mathbf{s}_i^{[2]} = 0$, as described in Lines 18 and 19 of Algorithm 2, for Case 1.

• Analysis for Case 2:

By interchanging the roles of A_1 and A_2 and following the proof for Case 1, one can show that each Node $i \in \ddot{\mathcal{A}}_1^{[1]}$ eventually sets $\mathbf{s}_i^{[2]} = 0$, as described in Line 19 of Algorithm 2, for Case 2.

Lemma 13. Given $|A_i| \leq \frac{n-|\mathcal{F}|-\sum_{l=3}^{\eta}|A_l|}{2}$ and $\eta \geq 2$, and assuming that all honest nodes eventually input their initial messages and keep running the COOL-UA protocol, it is true that

$$|\ddot{\mathcal{A}}_{i,i}^{[1]}| \le 2(k-1) \tag{78}$$

for $i \in \{1, 2\}$.

Proof. This proof closely follows that of [5, Lemma 8]. Here we assume that all honest nodes eventually input their initial messages and keep running the COOL-UA protocol. Without loss of generality, we just focus on the proof of $|\ddot{\mathcal{A}}_{2,2}^{[1]}| \leq 2(k-1)$, given the condition $|\mathcal{A}_2| \leq \frac{n-|\mathcal{F}|-\sum_{l=3}^{\eta}|\mathcal{A}_l|}{2}$ and $\eta \geq 2$. We first consider the case where $\eta \geq 3$. Recall that $\mathbb{U}_1^{(i)} := \{j \in [n] \mid (y_i^{(j)}, y_j^{(j)}) = (y_i^{(i)}, y_j^{(i)})\}$ and

We first consider the case where $\eta \geq 3$. Recall that $\mathbb{U}_1^{(i)} := \{j \in [n] \mid (y_i^{(j)}, y_j^{(j)}) = (y_i^{(i)}, y_j^{(i)})\}$ and $\mathbb{U}_0^{(i)} := \{j \in [n] \mid (y_i^{(j)}, y_j^{(j)}) \neq (y_i^{(i)}, y_j^{(i)})\}$ denote the link indicator sets \mathbb{U}_1 and \mathbb{U}_0 updated by Node i, as described in Line 9 of Algorithm 2 (see (13)). Also recall that $\mathbb{V}_1^{(i)} := \mathbb{U}_1^{(i)} \setminus \mathcal{F}$ and $\mathbb{V}_0^{(i)} := \mathbb{U}_0^{(i)} \setminus \mathcal{F}$. Here $u_i(j) \in \{0,1\}$ denotes the link indicator between Node i and Node j, defined in (12). Under the assumption that all honest nodes eventually input their initial messages and keep running the COOL-UA protocol, since each Node $i \in \ddot{\mathcal{A}}_{2,2}^{[1]}$ never set $\mathbf{s}_i^{[1]} = 0$, it holds true that

$$|V_0^{(i)}| \le t, \quad \forall i \in \ddot{\mathcal{A}}_{2,2}^{[1]}.$$
 (79)

If all honest nodes eventually input their initial messages and keep running the COOL-UA protocol, it is true that $|\mathbb{V}_0^{(i)}| + |\mathbb{V}_0^{(i)}| = n - |\mathcal{F}|$. Thus, from the above identity and from (79), we have

$$|\mathbb{V}_{1}^{(i)}| = n - |\mathcal{F}| - |\mathbb{V}_{0}^{(i)}| \ge n - |\mathcal{F}| - t, \quad \forall i \in \ddot{\mathcal{A}}_{2,2}^{[1]}.$$
 (80)

The result in (80) implies that

$$\sum_{j \in [n] \setminus \mathcal{F}} \mathbf{u}_i(j) \ge n - |\mathcal{F}| - t, \quad \forall i \in \ddot{\mathcal{A}}_{2,2}^{[1]}.$$
(81)

and that

$$\sum_{j \in [n] \setminus (\mathcal{F} \cup \mathcal{A}_2)} \mathbf{u}_i(j) \ge n - t - |\mathcal{F}| - |\mathcal{A}_2|, \quad \forall i \in \ddot{\mathcal{A}}_{2,2}^{[1]}.$$
(82)

In this setting, since $\boldsymbol{h}_i^{\mathsf{T}}\bar{\boldsymbol{w}}_1 \neq \boldsymbol{h}_i^{\mathsf{T}}\bar{\boldsymbol{w}}_2$ for any $i \in \ddot{\mathcal{A}}_{2,2}^{[1]}$ (see (69)), we conclude that $u_i(j) = 0$, $\forall i \in \ddot{\mathcal{A}}_{2,2}^{[1]}$, $\forall j \in \mathcal{A}_1$. This result implies that the inequality in (82) can be updated as

$$\sum_{j \in [n] \setminus (\mathcal{F} \cup \mathcal{A}_2 \cup \mathcal{A}_1)} \mathbf{u}_i(j) \ge n - t - |\mathcal{F}| - |\mathcal{A}_2|, \quad \forall i \in \ddot{\mathcal{A}}_{2,2}^{[1]}.$$
(83)

Given $[n] \setminus (\mathcal{F} \cup \mathcal{A}_2 \cup \mathcal{A}_1) = \bigcup_{l=3}^{\eta} \mathcal{A}_l$, the result in (83) can be rewritten as

$$\sum_{j \in \cup_{l=3}^{\eta} \mathcal{A}_l} \mathbf{u}_i(j) \ge n - t - |\mathcal{F}| - |\mathcal{A}_2|, \quad \forall i \in \ddot{\mathcal{A}}_{2,2}^{[1]}$$

$$(84)$$

which implies the following bound

$$\sum_{i \in \ddot{\mathcal{A}}_{0,2}^{[1]}} \sum_{j \in \cup_{l=3}^{\eta} \mathcal{A}_{l}} \mathbf{u}_{i}(j) \ge (n - t - |\mathcal{F}| - |\mathcal{A}_{2}|) \cdot |\ddot{\mathcal{A}}_{2,2}^{[1]}|. \tag{85}$$

On the other hand, for any given $j \in A_{l^*}$ and $l^* \in [3, \eta]$, we have

$$\sum_{i \in \ddot{\mathcal{A}}_{2,2}^{[1]}} \mathbf{u}_i(j) \le \sum_{i \in \mathcal{A}_2^{[1]}} \mathbf{u}_i(j) = \sum_{i \in \mathcal{A}_{2,l^{\star}}^{[1]}} \mathbf{u}_i(j) + \sum_{i \in \mathcal{A}_2^{[1]} \setminus \mathcal{A}_{2,l^{\star}}^{[1]}} \mathbf{u}_i(j) = \sum_{i \in \mathcal{A}_{2,l^{\star}}^{[1]}} \mathbf{u}_i(j) \le |\mathcal{A}_{2,l^{\star}}^{[1]}| \le k - 1$$
 (86)

where the above result uses the identity that $\boldsymbol{h}_{i}^{\mathsf{T}}\bar{\boldsymbol{w}}_{l^{\star}} \neq \boldsymbol{h}_{i}^{\mathsf{T}}\bar{\boldsymbol{w}}_{2}$ for $i \in \mathcal{A}_{2}^{[1]} \setminus \mathcal{A}_{2,l^{\star}}^{[1]}$ (see (8)), and from Lemma 7. The result in (86) then implies that

$$\sum_{j \in \cup_{l=3}^{\eta} \mathcal{A}_l} \sum_{i \in \ddot{\mathcal{A}}_{2,2}^{[1]}} \mathbf{u}_i(j) \le (k-1) \cdot \sum_{l=3}^{\eta} |\mathcal{A}_l|$$
 (87)

From the inequalities (85) and (87), we have $(n-t-|\mathcal{F}|-|\mathcal{A}_2|)\cdot |\ddot{\mathcal{A}}_{2,2}^{[1]}| \leq (k-1)\cdot \sum_{l=3}^{\eta} |\mathcal{A}_l|$ and

$$|\ddot{\mathcal{A}}_{2,2}^{[1]}| \le \frac{(k-1) \cdot \sum_{l=3}^{\eta} |\mathcal{A}_l|}{n-t-|\mathcal{F}|-|\mathcal{A}_2|},$$
 (88)

where $n-t-|\mathcal{F}|-|\mathcal{A}_2|>0$ is true given $|\mathcal{A}_2|\leq \frac{n-|\mathcal{F}|-\sum_{l=3}^{\eta}|\mathcal{A}_l|}{2}$ and $n\geq 3t+1$. The bound in (88) can be further extended as

$$|\ddot{\mathcal{A}}_{2,2}^{[1]}| \le \frac{(k-1) \cdot \sum_{l=3}^{\eta} |\mathcal{A}_l|}{n-t-|\mathcal{F}| - \frac{n-|\mathcal{F}| - \sum_{l=3}^{\eta} |\mathcal{A}_l|}{2}} = \frac{2(k-1)}{\frac{n-2t-|\mathcal{F}|}{\sum_{l=3}^{\eta} |\mathcal{A}_l|} + 1} \le \frac{2(k-1)}{0+1} = 2(k-1)$$
(89)

where the first inequality follows from the condition $|\mathcal{A}_2| \leq \frac{n-|\mathcal{F}|-\sum_{l=3}^{\eta}|\mathcal{A}_l|}{2}$; and the second inequality results from the fact that $\frac{n-2t-|\mathcal{F}|}{\sum_{l=3}^{\eta}|\mathcal{A}_l|} > 0$ in this case with $\eta \geq 3$.

We now focus on the case where $\eta = 2$. We assume that $|\ddot{\mathcal{A}}_{2,2}^{[1]}| > 0$. Under this assumption, by following the steps in (79)-(84), and given $\eta = 2$, we have

$$0 = \sum_{j \in \cup_{l=3}^{n} \mathcal{A}_l} \mathbf{u}_i(j) \ge n - t - |\mathcal{F}| - |\mathcal{A}_2|, \quad \forall i \in \ddot{\mathcal{A}}_{2,2}^{[1]}.$$
(90)

The bound in (90) contradicts the condition $|\mathcal{A}_2| \leq \frac{n-|\mathcal{F}|-\sum_{l=3}^{\eta}|\mathcal{A}_l|}{2} < n-t-|\mathcal{F}|$. Hence, the assumption $|\ddot{\mathcal{A}}_{2,2}^{[1]}| > 0$ leads to a contradiction; therefore, it is true that $|\ddot{\mathcal{A}}_{2,2}^{[1]}| = 0$ for this case with $\eta = 2$. This completes the proof.

```
Algorithm 3 OciorACOOL* protocol with identifier ID for a small t. Code is shown for Node i \in [n].
```

```
// ** This protocol is designed for the case where t is very small compared to n **
     // ** It includes OciorACOOL within S' := [n'], and a Strongly-Honest-Majority Distributed Multicast (SHMDM) from S'. **
     // ** Here n' := 3t + 1. **
 1: Initially set n' \leftarrow 3t + 1; k \leftarrow t/3; \mathbb{Y}_{oec} \leftarrow \{\}
     // *************** OciorACOOL ************
 2: upon receiving a non-empty message input w_i and i \in [n'] do:
 3:
         pass w_i into OciorACOOL as an input value
 4:
         run the OciorACOOL protocol with other nodes within [n']
     upon outputting a message w' from OciorACOOL protocol, and i \in [n'] do:
 5:
         [y_1', y_2', \dots, y_{n'}'] \leftarrow \text{ECCEnc}(n', k, \boldsymbol{w}')
 6:
         send (SHMDM, ID, y_i) to all nodes within [n] \setminus [n']
 7:
         output w' and terminate
 8:
 9: upon receiving (SHMDM, ID, y'_i) from Node j \in [n'] for the first time, and i \notin [n'] do:
         \begin{aligned} \mathbb{Y}_{\text{oec}}[j] \leftarrow y_j' \\ \text{if } |\mathbb{Y}_{\text{oec}}| \geq k + t \text{ then} \end{aligned}
10:
11:
                                                                                                                          // online error correcting
12:
              \hat{\boldsymbol{w}} \leftarrow \text{ECCDec}(n', k, \mathbb{Y}_{\text{oec}})
13:
              [y_1, y_2, \cdots, y_{n'}] \leftarrow \text{ECCEnc}(n', k, \hat{\boldsymbol{w}})
14:
              if at least k+t symbols in [y_1,y_2,\cdots,y_{n'}] match with those in \mathbb{Y}_{\text{oec}} then
15:
                  output \hat{w} and terminate
```

Algorithm 4 OciorRBA protocol with identifier ID. Code is shown for Node i for $i \in [n]$.

1: Initially set $k \leftarrow t/3$; $\mathbf{w}^{(i)} \leftarrow \bot$; $I_{\text{oecfinal}} \leftarrow 0$; $\mathbb{Y}_{\text{oec}} \leftarrow \{\}$; $I_3 \leftarrow 0$

```
2: upon receiving a non-empty message input w_i do:
 3:
         pass w_i into COOL-UA as an input value
     4: upon delivery of \mathbb{S}_{v^*}^{[2]} from COOL-UA such that |\mathbb{S}_{v^*}^{[2]}| \ge n - t, for a v^* \in \{1, 0\}, and (READY, ID, *) not yet sent do:
         send (READY, ID, v*) to all nodes // For some application in [6], the value of v* is delivered to the protocol invoking OciorRBA
 5:
 6: upon receiving t+1 (READY, ID, v) messages from different nodes for the same v, and (READY, ID, *) not yet sent do:
 7:
         send (READY, ID, v) to all nodes
 8:
     upon receiving 2t + 1 (READY, ID, v) messages from different nodes for the same v do:
         set v^{\diamond} \leftarrow v
 9:
         if v^{\diamond} = 0 then output w^{(i)} = \bot and terminate else set I_3 \leftarrow 1
10:
     11: upon I_3 = 1 do: // For some application in [6], I_3 may be set I_3 \leftarrow 1 by receiving a binary input b = 1 (other than message input)
         if COOL-UA has delivered [\boldsymbol{w}^{(i)}, \mathbf{s}_i^{[2]}, \mathbf{v}_i] with \mathbf{s}_i^{[2]} = 1 then
12:
13:
              output w^{(i)} and terminate
14:
              wait until COOL-UA delivering at least t+1 (SYMBOL, ID, (y_i^{(j)}, *)), \forall i \in \mathbb{S}_1^{[2]}, for the same y_i^{(j)} = y^*, for some y^*
15:
                                                                                                    // update coded symbol based on majority rule
16:
              send (CORRECT, ID, y_i^{(i)}) to all nodes
17:
              wait until I_{\text{oecfinal}} = 1
18:
              output w^{(i)} and terminate
19:
     upon receiving (CORRECT, ID, y_i^{(j)}) from Node j for the first time, j \notin \mathbb{Y}_{oec}, and I_{oecfinal} = 0 do:
20:
         \begin{split} \mathbb{Y}_{\text{oec}}[j] &\leftarrow y_j^{(j)} \\ \text{if } & |\mathbb{Y}_{\text{oec}}| \geq k+t \text{ then} \\ & \hat{\boldsymbol{w}} \leftarrow \text{ECCDec}(n,k,\mathbb{Y}_{\text{oec}}) \end{split}
21:
22:
                                                                                                                        // online error correcting
23:
24:
              [y_1, y_2, \cdots, y_n] \leftarrow \text{ECCEnc}(n, k, \hat{\boldsymbol{w}})
              if at least k+t symbols in [y_1,y_2,\cdots,y_n] match with those in \mathbb{Y}_{oec} then set \mathbf{w}^{(i)}\leftarrow\hat{\mathbf{w}} and I_{oecfinal}\leftarrow 1
25:
     upon COOL-UA having delivered (SYMBOL, ID, (*, y_j^{(j)})) and \mathbb{S}_1^{[2]} such that j \in \mathbb{S}_1^{[2]}, and j \notin \mathbb{Y}_{oec}, and I_{oecfinal} = 0 do:
26:
          \mathbb{Y}_{\text{oec}}[j] \leftarrow y_i^{(j)}
27:
         run the OEC steps as in Lines 22-25
28:
```

REFERENCES

- [1] M. Pease, R. Shostak, and L. Lamport, "Reaching agreement in the presence of faults," *Journal of the ACM*, vol. 27, no. 2, pp. 228–234, Apr. 1980.
- [2] L. Lamport, R. Shostak, and M. Pease, "The Byzantine generals problem," ACM Transactions on Programming Languages and Systems (TOPLAS), vol. 4, no. 3, pp. 382–401, Jul. 1982.
- [3] J. Chen, "Optimal error-free multi-valued Byzantine agreement," in *International Symposium on Distributed Computing (DISC)*, Oct. 2021.
- [4] —, "Fundamental limits of Byzantine agreement," 2020, available on arXiv: https://arxiv.org/pdf/2009.10965.pdf.
- [5] —, "OciorCOOL: Faster Byzantine agreement and reliable broadcast," Sep. 2024, available on arXiv: https://arxiv.org/abs/2409.06008.
- [6] —, "OciorMVBA: Near-optimal error-free asynchronous MVBA," Dec. 2024, available on arXiv: https://arxiv.org/abs/2501.00214.
- [7] —, "OciorABA: Improved error-free asynchronous Byzantine agreement via partial vector agreement," Jan. 2025, available on arXiv: https://arxiv.org/abs/2501.11788.
- [8] —, "Ocior: Ultra-fast asynchronous leaderless consensus with two-round finality, linear overhead, and adaptive security," Sep. 2025, available on arXiv: https://arxiv.org/abs/2509.01118.
- [9] F. Li and J. Chen, "Communication-efficient signature-free asynchronous Byzantine agreement," in *Proc. IEEE Int. Symp. Inf. Theory (ISIT)*, Jul. 2021.
- [10] J. Zhu, F. Li, and J. Chen, "Communication-efficient and error-free gradecast with optimal resilience," in *Proc. IEEE Int. Symp. Inf. Theory (ISIT)*, Jun. 2023, pp. 108–113.
- [11] M. Fitzi and M. Hirt, "Optimally efficient multi-valued Byzantine agreement," in *Proceedings of the ACM Symposium on Principles of Distributed Computing (PODC)*, Jul. 2006, pp. 163–168.
- [12] G. Liang and N. Vaidya, "Error-free multi-valued consensus with Byzantine failures," in *Proceedings of the ACM Symposium on Principles of Distributed Computing (PODC)*, Jun. 2011, pp. 11–20.
- [13] C. Ganesh and A. Patra, "Optimal extension protocols for Byzantine broadcast and agreement," in Distributed Computing, Jul. 2020.
- [14] A. Loveless, R. Dreslinski, and B. Kasikci, "Optimal and error-free multi-valued Byzantine consensus through parallel execution," 2020, available on: https://eprint.iacr.org/2020/322.

Algorithm 5 OciorRBC protocol with identifier ID. Code is shown for Node i for $i \in [n]$.

```
// ** OciorRBC is a balanced RBC protocol that balances communication between the leader and the other nodes. **
     // ** Without balancing, in the initial phase the leader simply broadcasts the entire message to each node (see Lines 14-17). **
 1: Initially set k \leftarrow t/3; \mathbf{w}_i \leftarrow \bot; I_{\text{oec}} \leftarrow 0; \mathbb{Z}_{\text{oec}} \leftarrow \{\}; I_3 \leftarrow 0
     2: upon receiving a non-empty message input w, and if this node is the leader do:
 3:
         [z_1, z_2, \cdots, z_n] \leftarrow \text{ECCEnc}(n, k, \boldsymbol{w})
         send (LEADER, ID, z_j) to Node j, \forall j \in [n]
 4:
 5: upon receiving (LEADER, ID, z_i) from the leader for the first time do:
 6:
         send (INITIAL, ID, z_i) to all nodes
                                                                                                                        // echo coded symbol
 7: upon receiving message (INITIAL, ID, z_i) from Node j for the first time, and I_{oec} = 0 do:
 8:
          \begin{aligned} & \text{if} & & |\mathbb{Z}_{\text{oec}}| \geq k + t \text{ then} \\ & & & \tilde{\boldsymbol{w}} \leftarrow \text{ECCDec}(n, k, \mathbb{Z}_{\text{oec}}) \end{aligned} 
 9:
                                                                                                                     // online error correcting (OEC)
10:
             [z'_1, z'_2, \cdots, z'_n] \leftarrow \text{ECCEnc}(n, k, \tilde{\boldsymbol{w}})
11:
             if at least k+t symbols in [z_1', z_2', \cdots, z_n'] match with those in \mathbb{Z}_{\text{oec}}, and \tilde{\boldsymbol{w}} is non-empty then
12:
13:
                 \boldsymbol{w}_i \leftarrow \tilde{\boldsymbol{w}}; I_{\text{oec}} \leftarrow 1
     upon receiving a non-empty message input w, and if this node is the leader do:
14:
15:
         send (MESSAGE, ID, \boldsymbol{w}) to all nodes
16: upon receiving (MESSAGE, ID, w) from the leader for the first time do:
17:
         oldsymbol{w}_i \leftarrow 	ilde{oldsymbol{w}}
     // ****************** OciorRBA*************
18: upon w_i \neq \bot do:
         pass w_i into OciorRBA as an input value
19:
    upon delivery of the output value w^{(i)} from OciorRBA do:
20:
         output w^{(i)} and terminate
21:
```

- [15] K. Nayak, L. Ren, E. Shi, N. Vaidya, and Z. Xiang, "Improved extension protocols for Byzantine broadcast and agreement," in *International Symposium on Distributed Computing (DISC)*, Oct. 2020.
- [16] A. Patra, "Error-free multi-valued broadcast and Byzantine agreement with optimal communication complexity," in *International Conference on Principles of Distributed Systems (OPODIS)*, 2011, pp. 34–49.
- [17] C. Cachin and S. Tessaro, "Asynchronous verifiable information dispersal," in *IEEE Symposium on Reliable Distributed Systems (SRDS)*, Oct. 2005.
- [18] P. Civit, M. A. Dzulfikar, S. Gilbert, R. Guerraoui, J. Komatovic, M. Vidigueira, and I. Zablotchi, "Efficient signature-free validated agreement," in *International Symposium on Distributed Computing (DISC)*, vol. 319, Oct. 2024, pp. 14:1–14:23.
- [19] M. Mizrahi Erbes and R. Wattenhofer, "Brief announcement: Extending asynchronous Byzantine agreement with crusader agreement," in *Proceedings of the ACM Symposium on Principles of Distributed Computing (PODC)*, Jun. 2025, pp. 50–53.
- [20] I. Abraham and G. Asharov, "ABEL: Perfect asynchronous Byzantine extension from list-decoding," in *International Symposium on Distributed Computing (DISC)*, Oct. 2025, pp. 1:1–1:20.
- [21] M. Sudan, "Decoding of reed solomon codes beyond the error-correction bound," *Journal of Complexity*, vol. 13, no. 1, pp. 180–193, Mar. 1997.
- [22] V. Guruswami and C. Wang, "Linear-algebraic list decoding for variants of reed?solomon codes," *IEEE Trans. Inf. Theory*, vol. 59, no. 6, pp. 3257–3268, 2013.
- [23] I. Reed and G. Solomon, "Polynomial codes over certain finite fields," Journal of the Society for Industrial and Applied Mathematics, vol. 8, no. 2, pp. 300–304, Jun. 1960.
- [24] M. Sipser and D. Spielman, "Expander codes," IEEE Trans. Inf. Theory, vol. 42, no. 6, pp. 1710–1722, Nov. 1996.
- [25] M. Ben-Or, R. Canetti, and O. Goldreich, "Asynchronous secure computation," in *Proceedings of the Twenty-Fifth Annual ACM Symposium on Theory of Computing*, 1993, pp. 52–61.
- [26] E. Berlekamp, "Nonbinary BCH decoding (abstr.)," IEEE Trans. Inf. Theory, vol. 14, no. 2, pp. 242-242, Mar. 1968.
- [27] S. Gao, "A new algorithm for decoding Reed-Solomon codes," in *Communications, Information and Network Security*. Springer, 2003, pp. 55–68.
- [28] R. Roth, Introduction to coding theory. Cambridge University Press, 2006.