Error-free Multi-valued Broadcast and Byzantine Agreement with Optimal Communication Complexity

Arpita Patra
Department of Computer Science
Aarhus University, Denmark.
arpita@cs.au.dk

Abstract

In this paper we present first ever *error-free, asynchronous* broadcast (called as A-cast) and Byzantine Agreement (called as ABA) protocols with optimal communication complexity and fault tolerance. Our protocols are multi-valued, meaning that they deal with ℓ bit input and achieve communication complexity of $\mathcal{O}(n\ell)$ bits for large enough ℓ for a set of $n \geq 3t+1$ parties in which at most t can be Byzantine corrupted.

In synchronous settings, Fitzi and Hirt (PODC'06) proposed probabilistically correct multi-valued broadcast (BC) and Byzantine Agreement (BA) protocols with optimal complexity of $\mathcal{O}(n\ell)$ bits. Recently, Liang and Vaidya (PODC'11) achieved the same *deterministically*, i.e. without error probability. In asynchronous settings, Patra and Rangan (Latincrypt'10, ICITS'11) reported similar protocols with error probability. Here we achieve optimal complexity of $\mathcal{O}(n\ell)$ bits for asynchronous error-free case.

Following all the previous works on multi-valued protocols, we too follow reduction-based approach for our protocols, meaning that our multi-valued protocols are designed given existing A-cast and ABA protocols for small message (possibly for single bit). However compared to existing reductions, our reductions are simple and elegant. More importantly, our reductions run in constant expected time, in contrast to $\mathcal{O}(n)$ of Patra and Rangan (ICITS'11). Furthermore our reductions invoke less or equal number of instances of protocols for single bit in comparison to the reductions of Patra and Rangan.

In synchronous settings, while the reduction of Fitzi and Hirt is constant-round and invokes $\mathcal{O}(n(n+\kappa))$ (κ is the error parameter) instances of protocols for single bit, the reduction of Liang and Vaidya calls for round complexity and number instances that are in fact function of the message size, $\mathcal{O}(\sqrt{\ell}+n^2)$ and $\mathcal{O}(n^2\sqrt{\ell}+n^4)$, respectively where $\ell = \Omega(n^6)$. By adapting our techniques from asynchronous settings, we present new *error-free* reduction in synchronous world that is constant-round and calls for only $\mathcal{O}(n^2)$ instances of protocols for single bit which is at least as good as Fitzi and Hirt.

Keywords: Asynchronous, Multi-valued, A-cast, Byzantine Agreement, Communication Complexity

1 Introduction

The problem of Broadcast (BC) and Byzantine Agreement (BA) (also popularly known as consensus) were introduced in [PSL80] and since then they have been considered as the most fundamental problems in distributed computing. In brief, a BC protocol allows a special party among a set of parties, called sender, to send some message identically to all other parties. The challenge lies in achieving the above task despite the presence of some faulty parties (possibly including the sender), who may deviate from the protocol arbitrarily. The BA primitive is slightly different from BC. A BA protocol allows a set of parties, each holding some input bit, to agree on a common bit, even though some of the parties may act maliciously in order to make the honest parties disagree. The BC and BA primitives have been used as building blocks in several important secure distributed computing tasks such as Secure Multiparty Computation (MPC) [BOGW88, BKR94, RBO89], Verifiable Secret Sharing (VSS) [CGMA85, BOGW88, RBO89] etc.

An important, practically motivated variant of BC and BA problem are asynchronous broadcast (known as A-cast) and asynchronous BA (known as ABA) that study the conventional BC and BA problems in asynchronous network settings. It is well-known that asynchronous network setting is considered to be more realistic than synchronous network setting. The works of [BO83, Rab83, Bra84, FM88, CR93, Can95, ADH08, PW92, PR11] have reported different A-cast and ABA protocols. In this paper, we focus on the communication complexity of *error-free* A-cast and ABA protocols and present first ever optimal protocols.

The Model. We follow the standard network model of [PSL80, CR93, Can95]. Specifically, our A-cast, ABA, BC and BA protocols are carried out among a set of n parties, say $\mathcal{P} = \{P_1, \ldots, P_n\}$, where every two parties are directly connected by an authenticated and secure communication channel and t out of the n parties can be under the influence of a *computationally unbounded Byzantine (active) adversary*, denoted as $\mathcal{A}_t$. We assume that $n = 3t + 1$ which is the minimum number of parties required to design *error-free* A-cast, ABA, BC and BA protocols [Lyn96, PSL80]. The adversary $\mathcal{A}_t$, completely dictates the parties under its control and can force them to deviate from the protocol in any arbitrary manner. The parties not under the influence of $\mathcal{A}_t$ are called *honest or uncorrupted.*

Since synchronous network is extremely well-studied in the literature, in the following we only recall the important features of an asynchronous network. In asynchronous network, the communication channels between the parties have arbitrary, yet finite delay (i.e. the messages are guaranteed to reach eventually). To model this, we assume that $\mathcal{A}_t$ controls the network and may delay messages between any two honest parties. However, it cannot read or modify these messages as the links are private and authenticated, and it also has to eventually deliver all the messages by honest parties. In asynchronous network, the inherent difficulty in designing a protocol comes from the fact that a party can not distinguish between a slow party (whose message is simply delayed in the network) and a corrupted party (who did not send the message at all). So an honest party can not wait for the values from all the parties, as it would potentially mean that the party is waiting for the values from corrupted parties who may never send any value at all. Therefore, it would make the honest party wait indefinitely. As a result, an honest party in asynchronous network should start computation upon receiving values from $n - t$ parties at any particular step. However, this may risk ignoring the values of up to t potentially honest parties at every step. The above reason makes it difficult to design protocols in asynchronous settings. For comprehensive introduction to asynchronous protocols, see [Can95].

We do not make any cryptographic assumptions such as public key infrastructure (PKI) etc for designing our protocols.

Definitions. We now define A-cast and ABA formally.

Definition 1.1 (A-cast [Can95]) *Let* Π *be an asynchronous protocol executed among the set of parties* $\mathcal{P}$ *and initiated by a special party called sender* $S \in \mathcal{P}$*, having input* m *(the message to be sent).* Π *is an A-cast protocol tolerating* $\mathcal{A}_t$ *if the following hold, for every behavior of* $\mathcal{A}_t$ *and every input* m*:*

- **Termination:** *If* S *is honest, then all honest parties in* $\mathcal{P}$ *will eventually terminate* Π. *If any honest party terminates* Π*, then all honest parties will eventually terminate* Π.
- **Correctness:** *If the honest parties terminate* Π*, then they do so with a common output* $m^\star$*. Furthermore, if the sender* S *is honest then* $m^\star = m$.

Definition 1.2 (ABA [CR93]) *Let* Π *be an asynchronous protocol executed among the set of parties* $\mathcal{P}$*, with each party having a private binary input. We say that* Π *is an ABA protocol tolerating* $\mathcal{A}_t$ *if the following hold, for every possible behavior of* $\mathcal{A}_t$ *and every possible input:*

- **Termination***: All honest parties eventually terminate the protocol.*
- **Correctness***: All honest parties who have terminated the protocol hold identical outputs. Furthermore, if all honest parties had same input, say* ρ*, then all honest parties output* ρ.

The celebrated result of Fischer, Lynch, and Paterson [FLP85] shows that any ABA protocol that never reaches disagreement (i.e., has no executions where two honest parties output different values) must have some nonterminating executions, where some honest party does not output a value. For a protocol that never reaches disagreement, the best we can hope for is that the set of nonterminating executions has probability zero. Such protocols are termed as almost-surely terminating by Abraham et al. [ADH08]. In this work, we construct ABA protocol that is almost-surely terminating and has no error in correctness. The important complexity measures of A-cast and ABA protocol are:

- **Communication Complexity**: It is the total number of bits communicated by the *honest* parties in the protocol;
- **Running Time**: The current definition is taken from [CR93, Can95]. Consider a virtual 'global clock' measuring time in the network. Note that the parties cannot read this clock. Let the *delay* of a message be the time elapsed from its sending to its receipt. Let the *period* of a finite execution of a protocol be the longest delay of a message in the execution. The *duration* of a finite execution is the total time measured by the global clock divided by the period of the execution.

  The *expected running time* of a protocol is the maximum over all inputs and applicable adversaries, of the average over the random inputs of the parties, of the duration of executions of the protocol.

While the basic definitions of A-cast and ABA consider message of single bit, *multi-valued* protocols allow message to be long string of bits and exploit the fact that the task is to be attained for the entire string and not bit by bit. This fact generally allows a *multi-valued* protocol to be considerably more efficient than many parallel executions of protocol for single bit.

Brief Literature. Error-free BC and BA protocol in synchronous network are possible if and only if $n \geq 3t+1$ [PSL80, Lyn96]. The same bound holds for A-cast and ABA both with and without error probability [Lyn96]. The seminal result of [DR85] shows that any error-free BA or BC must communicate $\Omega(n^2)$ bits (which again carry over for the case of A-cast and ABA). Since the message must be at least single bit, the lower bound on the communication complexity for single bit is $\Omega(n^2)$ bits. However, communication complexity of $\mathcal{O}(n\ell)$ bits can be achieved for large enough value of ℓ (at least $\ell \geq n$ bits) as shown in [FH06]. Requiring large value for ℓ is practically motivated in many distributed computing applications, like reaching agreement on a large file in fault-tolerant distributed storage system, distributed voting where ballots containing gigabytes of data is to be handled, MPC where many broadcasts and agreements are invoked which can be combined into fewer executions of multi-valued protocols.

Following the approach of Turpin and Coan [TC84], all the subsequent multi-valued protocols apply reduction-based approach [FH06, LV11, PR11, PR10], meaning that they are constructed based on access to protocols for small message or single bit message. The reductions presented in [FH06, LV11, PR11, PR10] achieve optimal communication complexity of $\mathcal{O}(n\ell)$ bits. In synchronous settings, while the reduction presented in [FH06] involves error probability, the reduction of [LV11] is error-free. However, the reduction of [FH06] is constant-round, as opposed to the reduction of [LV11] which has a round complexity of $\mathcal{O}(\sqrt{\ell}+n^2)$, where $\ell = \Omega(n^6)$. Furthermore, the reduction of [FH06] invokes $\mathcal{O}(n(n+\kappa))$ (κ is the error parameter of their protocol) instances of protocol for single bit, while the reduction of [LV11] invokes $\mathcal{O}(n^2\sqrt{\ell}+n^4)$ instances of protocol for single bit.

In asynchronous settings, [PR11, PR10] reported multi-valued protocols with error probability.

Our Contribution. In this work, we achieve optimal complexity of $\mathcal{O}(n\ell)$ bits for *error-free* A-cast and ABA with optimal fault-tolerance. We too follow reduction-based approach of [TC84]. Both the reductions run in constant time. Our reduction for A-cast calls for $\mathcal{O}(n^2 \log n)$ instances of A-cast for single bit which is exactly same as the reduction of [PR10]. However interestingly, our reduction for ABA runs in constant expected time and invokes only $\mathcal{O}(n)$ instances of ABA for single bit, whereas the reduction of [PR11] runs in $\mathcal{O}(t)$ expected time and invokes $\mathcal{O}(n^3)$ instances of ABA for single bit. Therefore our reductions are better than existing multi-valued asynchronous protocols in all the following aspects: (a) error-free, (b) constant time and (c) less (or equal) number of invocations to protocols for single bit. The comparison is summarized below:

Ref.	Type	Fault Tolerance	CC in bits	Running Time	No. Invocations to single-bit protocol
[PR10], A-cast	Probabilistic	$t < n/3$	$\mathcal{O}(n\ell)$	constant	$\mathcal{O}(n^2 \log n)$ A-cast
[PR11], ABA	Probabilistic	$t < n/3$	$\mathcal{O}(n\ell)$	$\mathcal{O}(n)$	$\mathcal{O}(n^3)$ ABA
This paper, A-cast	Error-free	$t < n/3$	$\mathcal{O}(n\ell)$	constant	$\mathcal{O}(n^2 \log n)$ A-cast
This paper, ABA	Error-free	$t < n/3$	$\mathcal{O}(n\ell)$	constant	$\mathcal{O}(n)$ ABA

Technically, our reductions are simple and are based on linear error correcting code (e.g. Reed-Solomon Code) and a graph theoretic algorithm for finding some special structure (called (n,t)-star; defined in Section 2) in undirected graph [CR93, Can95]. While the existing reductions for multi-valued protocols [PR11, LV11] are constructed in player-elimination [HMP00] or dispute control [BTH06] framework, our reductions do not require them and therefore they are more elegant. We recall that in player-elimination or dispute control framework, the executions are carried out in non-robust manner, in the sense that if all the parties including the corrupted parties behave according to the protocol then the execution successfully achieves its task; otherwise the execution may fail in which case a small subset of parties will be outputted such that at least one of them is guaranteed to be corrupted. Depending on the framework, either that subset of parties will be eliminated (in player-elimination framework) and the execution will be repeated with the remaining parties or the execution will be repeated again with the guarantee that the subset of the parties can never cause to fail the execution anymore (in dispute control framework). Finally, we note that the multi-valued A-cast protocol of [PR10] also employ the algorithm for finding (n,t)-star [CR93, Can95]. However, we mark an important and crucial observation about the outcome of the algorithm in our context that allows to achieve our goal.

We now compare our results with the current best *error-free* A-cast and ABA for single bit. The only error-free A-cast is due to [Bra84] that communicates $\mathcal{O}(n^2)$ bits and runs in constant time. Similarly, the only error-free ABA is due to [ADH08] that runs in $\mathcal{O}(n^2)$ time and requires communication of $\mathcal{O}(n^8 \log n)$ bits and A-cast of same number of bits. Our protocols in this paper show clear improvement over ℓ executions of these protocols.

Finally, we discuss our results in synchronous settings. In synchronous settings, the reduction of [FH06] is constant-round and it invokes $\mathcal{O}(n(n+\kappa))$ (κ is the error parameter) instances of protocols for single bit. However the reduction of [LV11] calls for round complexity and number instances that are function of the message size, i.e. $\mathcal{O}(\sqrt{\ell}+n^2)$ and $\mathcal{O}(n^2\sqrt{\ell}+n^4)$, respectively where $\ell = \Omega(n^6)$. By adapting our techniques from asynchronous settings, we present new *error-free* reduction in synchronous world that is constant-round and calls for only $\mathcal{O}(n^2)$ instances of protocols for single bit which is at least as good as [FH06]. The summary is presented below:

Ref.	Type	Fault Tolerance	Communication Complexity (CC) in bits	Round Complexity	No. Invocations to single-bit protocol
[FH06]	Probabilistic	$t < n/2$	$\mathcal{O}(n\ell)$	constant	$\mathcal{O}(n(n+\kappa))$
[LV11]	Error-free	$t < n/3$	$\mathcal{O}(n\ell)$	$\mathcal{O}(\sqrt{\ell}+n^2)$	$\mathcal{O}(n^2\sqrt{\ell}+n^4)$
This paper	Error-free	$t < n/3$	$\mathcal{O}(n\ell)$	constant	$\mathcal{O}(n^2)$

Road-map. In section 2 and 3, we present our construction for A-cast and ABA respectively. We present our BA and BC protocols in Section 4 and then conclude in Section 5.

2 Error-free A-cast with Optimal Communication Complexity

Here we present our A-cast protocol. We start with brief presentation of the tools that we use for the protocol: (a) The existing A-cast protocol of Bracha [Bra84]; (b) An algorithm for finding a graphical structure called (n,t)-star in an undirected graph; (c) Linear Error Correcting Code. We discuss them one by one.

Bracha's A-cast. The first ever protocol for A-cast is due to Bracha [Bra84]. The protocol is error-free, runs with $n \geq 3t+1$ in constant time and communicates $\mathcal{O}(n^2)$ bits for a *single bit* message. The description of Bracha's A-cast protocol is available in [Can95]. For the sake of completeness, the protocol is provided in Appendix A.

Notation 2.1 *By saying that 'P_i A-casts M', we mean that P_i as a sender, initiates Bracha's A-cast protocol with M as the message. Similarly 'P_j receives M from the A-cast of P_i' will mean that P_j terminates the A-cast protocol initiated by P_i and outputs M. By the property of A-cast, if some honest party P_j terminates the A-cast protocol of some sender P_i with M as the output, then every other honest party will eventually do so, irrespective of the behavior of the sender P_i.*

Finding (n,t)-star in an Undirected Graph. We now describe an existing solution for a graph theoretic problem, called finding (n,t)-star in an undirected graph $G=(V,E)$.

Definition 2.1 (**(n,t)-star [Can95, BOCG93]:**) *Let G be an undirected graph with the n parties in $\mathcal{P}$ as its vertex set. We say that a pair $(\mathcal{C},\mathcal{D})$ of sets with $\mathcal{C} \subseteq \mathcal{D} \subseteq \mathcal{P}$ is an (n,t)-star in G, if: (i) $|\mathcal{C}| \geq n-2t$; (ii) $|\mathcal{D}| \geq n-t$; (iii) for every $P_j \in \mathcal{C}$ and every $P_k \in \mathcal{D}$ the edge (P_j,P_k) exists in G.*

Following an idea of [GJ79], Ben-Or et al. [BOCG93] have presented an elegant and efficient algorithm for finding an (n,t)-star in a graph of n nodes, *provided that the graph contains a clique of size* $n-t$. The algorithm, called Find-STAR outputs either an (n,t)-star or the message `star-Not-Found`. Whenever, the input graph contains a clique of size $n-t$, Find-STAR will always output an (n,t)-star in the graph. Actually, algorithm Find-STAR takes the complementary graph $\overline{G}$ of G as input and tries to find (n,t)-$\overline{\text{star}}$ in $\overline{G}$, where (n,t)-$\overline{\text{star}}$ is a pair $(\mathcal{C},\mathcal{D})$ of sets with $\mathcal{C} \subseteq \mathcal{D} \subseteq \mathcal{P}$, satisfying the following conditions: (a) $|\mathcal{C}| \geq n-2t$; (b) $|\mathcal{D}| \geq n-t$; (c) There are no edges between the nodes in $\mathcal{C}$ and nodes in $\mathcal{C} \cup \mathcal{D}$ in $\overline{G}$. Clearly, a pair $(\mathcal{C},\mathcal{D})$ representing an (n,t)-$\overline{\text{star}}$ in $\overline{G}$, is an (n,t)-star in G. Recasting the task of Find-STAR in terms of complementary graph $\overline{G}$, we say that Find-STAR outputs either an (n,t)-$\overline{\text{star}}$, or a message `star-Not-Found`. Whenever, *the input graph* $\overline{G}$ *contains an independent set of size* $n-t$, Find-STAR always outputs an (n,t)-$\overline{\text{star}}$. For simple notation, we denote $\overline{G}$ by H. The algorithm Find-STAR is presented in Figure 1. It is easy to see that the algorithm requires polynomial computation.

Figure 1: Algorithm for Finding (n,t)-star.

Algorithm Find-STAR(H)

1. Find a maximum matching M in H. Let N be the set of matched nodes (namely, the endpoints of the edges in M), and let $\overline{N} = \mathcal{P} \setminus N$.
2. Compute output as follows (which could be either (n,t)-$\overline{\text{star}}$ or a message `star-Not-Found`):
   (a) Let $T = \{P_i \in \overline{N} | \exists P_j, P_k \text{ s.t } (P_j,P_k) \in M \text{ and } (P_i,P_j),(P_i,P_k) \in E\}$. T is called the set of triangle-heads.
   (b) Let $\mathcal{C} = \overline{N} \setminus T$.
   (c) Let B be the set of matched nodes that have neighbors in $\mathcal{C}$. So $B = \{P_j \in N | \exists P_i \in \mathcal{C} \text{ s. t. } (P_i,P_j) \in E\}$.
   (d) Let $\mathcal{D} = \mathcal{P} \setminus B$. If $|\mathcal{C}| \geq n-2t$ and $|\mathcal{D}| \geq n-t$, output $(\mathcal{C},\mathcal{D})$. Otherwise, output `star-Not-Found`.

Linear Error Correcting Code. We use Reed-Solomon (RS) codes in our protocols. We consider an $(n,t+1)$ RS code in Galois Field $\mathbb{F} = GF(2^c)$, where $n \leq 2^c$. Each element of $\mathbb{F}$ is represented by c bits. An $(n,t+1)$ RS code encodes $t+1$ elements of $\mathbb{F}$ into a codeword consisting of n elements from $\mathbb{F}$. We denote the encoding function as ENC() and the corresponding decoding function as DEC(). Let $m_0, m_1, \ldots, m_t$ be the input to ENC, then ENC computes a codeword of length n, $(s_1,\ldots,s_n)$, as follows: It constructs a polynomial of degree-t, $f(x) = m_0 + m_1x + \ldots + m_tx^t$. It then computes $s_i = f(i)$. We use the following syntax for ENC: $(s_1,s_2,\ldots,s_n) = \text{ENC}(m_0,m_1,\ldots,m_t)$. Each element of the codeword is computed as a linear combination of the $t+1$ input data elements, such that every subset of $(t+1)$ elements from the codeword uniquely determine the input data elements. Similarly, knowledge of any $t+1$ elements from the codeword suffices to determine the remaining elements of the codeword.

The decoding function DEC can be applied as long as $t+1$ elements from a codeword are available. A RS code is capable of error correction and detection. The task of error correction is to find the error locations and error values in a received vector. On the other hand, error detection means an indication that errors have occurred, without attempting to correct them. We recall the following well known result from coding theory. DEC can correct up to c Byzantine error and simultaneously detect up to additional d Byzantine errors in a vector of length N (where $N \leq n$) if and only if $N-t-1 \geq 2c+d$. In our protocols, we may invoke DEC on a vector of length $N \leq n$ with specific value of c and d. If c, d and N satisfy the above relation, then DEC returns back the correct data elements corresponding to the vector; otherwise DEC returns 'failure'.

2.1 Multi-valued A-cast Protocol

With the above tools, we are now ready to present our multi-valued A-cast protocol, called Multi-Valued-Acast. We assume that the sender S has a message m containing ℓ bits that he would like to communicate to all the parties in $\mathcal{P}$ identically. Our protocol is structured into two phases, (a) **S-dependent Phase** and (b) **S-independent Phase**. In the

S-dependent phase, S proves that it has communicated the same message to at least a set of $2t+1$ parties, say $CORE$. The S-dependent phase, as the name suggests, demands S to perform some special roles. For an honest S, this phase will always be completed successfully. However, a corrupted S may choose not to perform his actions and therefore this phase may not be terminated for a corrupted S. The second phase, called S-independent phase is initiated upon completion of the first phase. If S successfully proves the existence of some $CORE$ in the first phase, then the parties in $CORE$ propagates their common message to the remaining parties without any help from S.

In the first phase, S communicates his message m to every party over private channel. Upon receiving a message from S, a party applies ENC on the message to get a codeword and communicates elements of the codeword to different party. Intuitively, the parties here check if they received the same message from S. They A-cast [Bra84] their responses. Based on the response of the parties, a consistency graph is constructed by the parties individually. S now finds a special structure in the graph which essentially proves that there is a set of at least $2t+1$ parties, $CORE$, to whom he indeed communicated same message. On finding such a structure, S A-casts the same and all other parties can verify if indeed such structure exists in their individual graph. In this process, all the (honest) parties agree on $CORE$ and proceed to second phase. The algorithm for finding (n,t)-star and an important observation are combined intelligibly in order to find the special structure in a graph. Very abstractly, the observation is that *if S is honest then eventually, the set $\mathcal{C}$ of an (n,t)-star will contain at least $t+1$ honest parties.* A formal proof in support of this observation will appear in Lemma 2.3.

In the second phase, the parties uses error correction and detection property of RS code to compute and agree on the common message of the parties in $CORE$. We now present the protocol in Figure 2 and subsequently prove its properties.

Lemma 2.2 *The honest parties in $CORE$ hold same message of length ℓ. If S is honest then the message is S's message.*

Proof: Recall that $CORE$ is the $\mathcal{E}$ component of a quadruple $(\mathcal{C},\mathcal{D},\mathcal{F},\mathcal{E})$ such that $(\mathcal{C},\mathcal{D})$ is an (n,t)-star and every party in $\mathcal{F}$ has at least $t+1$ neighbors in $\mathcal{C}$ and every party in $\mathcal{E}$ has at least $2t+1$ neighbors in $\mathcal{F}$. We now argue that these conditions imply that the parties in $\mathcal{E}$ and therefore $CORE$ hold same message of length ℓ.

First we prove that the honest parties in $\mathcal{C}$ hold the same message of length ℓ. We start with recalling that $\mathcal{D}$ contains at least $t+1$ honest parties and every $P_i \in \mathcal{C}$ is neighbor of every party in $\mathcal{D}$. Let $\{P_{i_1},\ldots,P_{i_\alpha}\}$ be the set of α honest parties in $\mathcal{D}$, where $\alpha \geq t+1$. Then for every P_i in $\mathcal{C}$, s_{ii_k} is same as $s_{i_k i_k}$ of all $k=[1,\alpha]$. Therefore the codewords corresponding to the messages of the honest parties in $\mathcal{C}$ are same at least at $t+1$ locations corresponding to the identities of the honest parties in $\mathcal{D}$. Since the codewords belong to $(n,t+1)$ RS code, the messages of the honest parties in $\mathcal{C}$ are same. Let the common message be m, $|m|=\ell$. Let $(s_1,\ldots,s_n)=\mathsf{ENC}(m_0,m_1,\ldots,m_t)$, where $m=m_0|m_1|\ldots,|m_t$. Now we show that every honest party $P_i \in \mathcal{F}$ holds s_i. Recall that P_i has at least $t+1$ neighbors in $\mathcal{C}$ in which at least one is honest, say P_j. This implies s_{ii} of P_i is same as s_{ji} of P_j. However, $s_{ji}=s_i$, since P_j holds m. Hence $s_{ii}=s_i$. Therefore every honest P_i in $\mathcal{F}$ holds s_i which is same as s_{ii}. Finally, we show that every honest $P_i \in \mathcal{E}$ holds m. Recall that P_i has at least $2t+1$ neighbors in $\mathcal{F}$ in which at least $t+1$ are honest. Let $\{P_{i_1},\ldots,P_{i_\alpha}\}$ be the set of α honest parties in $\mathcal{F}$, where $\alpha \geq t+1$. Then s_{ii_k} of P_i is same as $s_{i_k i_k}$ of every honest P_{i_k} for $k=[1,\alpha]$. Now $s_{i_k i_k}$ of P_{i_k} is same as s_{i_k}. Therefore the codeword corresponding to the message of $P_i \in \mathcal{E}$ matches with $(s_1,\ldots,s_n)$ at least at $t+1$ locations corresponding to the identities of the honest parties in $\mathcal{F}$. This implies the codeword of P_i is identical to $(s_1,\ldots,s_n)$, since they belong to $(n,t+1)$ RS code. Hence $P_i \in \mathcal{E}$ holds m. This completes the proof for the first part of the lemma. The second part of the lemma is easy to prove. ■

To prove the lemma below, we will show that when S is honest then eventually an (n,t)-star can be found such that the set $\mathcal{C}$ will contain at least $t+1$ *honest* parties. This observation is very crucial and is at the heart of our protocol.

Lemma 2.3 *If S is honest, then all the parties terminate* **S-dependent Phase***, after agreeing on $CORE$.*

Proof: If S is honest, then he sends same message m to all the parties. Therefore, all honest parties generate same codeword, $(s_1,\ldots,s_n)=\mathsf{ENC}(m_0,\ldots,m_t)$, such that $m=m_0|m_1|\ldots|m_t$. Therefore eventually there will be an edge between every pair of honest parties. This implies that there will be a clique of size at least $2t+1$ eventually. This guarantees that S will eventually find at least one (n,t)-star in G_S. Now we show that S will eventually find a quadruple $(\mathcal{C},\mathcal{D},\mathcal{F},\mathcal{E})$ such that $(\mathcal{C},\mathcal{D})$ is an (n,t)-star and every party in $\mathcal{F}$ has at least $t+1$ neighbors in $\mathcal{C}$ and

Figure 2: Error-free Multi-valued A-cast with Optimal Communication Complexity.

Protocol Multi-Valued-Acast(S,m)

S-dependent Phase:

Code for S.

1. S sends his message m to every P_i.

Code for every P_i including S.

1. On receiving message m_i, divide the ℓ bit message m_i into $t+1$ blocks, $m_{i0}, \ldots, m_{it}$, each containing $\frac{\ell}{t+1}$ (assume this to be an integer for simplicity) bits. Compute $(s_{i1}, \ldots, s_{in}) = \mathsf{ENC}(m_{i0}, \ldots, m_{it})$.
2. Send s_{ii} to every party. Send s_{ij} to P_j for $j = [1, n]$.
3. On receiving s_{jj} and s_{ji} from P_j, A-cast $\texttt{OK}(P_i, P_j)$ if $s_{jj} = s_{ij}$ and $s_{ji} = s_{ii}$.
4. Construct a graph G_i with the parties in $\mathcal{P}$ as the vertices. Add an edge (P_j, P_k) in G_i if $\texttt{OK}(P_j, P_k)$ and $\texttt{OK}(P_k, P_j)$ are received from the A-cast of P_j and P_k respectively.

Code for S.

1. Upon every new receipt of some $\texttt{OK}(*, *)$, update G_S. If a new edge is added to G_S, then execute Find-STAR($\overline{G_S}$). Let there are $\alpha \geq 0$ distinct (n, t)-stars that are found in the past from different executions of Find-STAR($\overline{G_S}$).
   (a) Now if an (n, t)-star is found from the current execution of Find-STAR($\overline{G_S}$) that is distinct from all the α (n, t)-star obtained before, do the following:
      i. Call the new (n, t)-star as $(\mathcal{C}^{\alpha+1}, \mathcal{D}^{\alpha+1})$.
      ii. Create a list $\mathcal{F}^{\alpha+1}$ as follows: Add P_j to $\mathcal{F}^{\alpha+1}$ if P_j has at least $t+1$ neighbors in $\mathcal{C}^{\alpha+1}$ in G_S.
      iii. Create a list $\mathcal{E}^{\alpha+1}$ as follows: Add P_j to $\mathcal{E}^{\alpha+1}$ if P_j has at least $2t+1$ neighbors in $\mathcal{F}^{\alpha+1}$ in G_S.
      iv. For every γ, with $\gamma = 1, \ldots, \alpha$ update $\mathcal{F}^{\gamma}$ and $\mathcal{E}^{\gamma}$:
         A. Add P_j to $\mathcal{F}^{\gamma}$, if $P_j \notin \mathcal{F}^{\gamma}$ and P_j has at least $t+1$ neighbors in $\mathcal{C}^{\gamma}$ in G_S.
         B. Add P_j to $\mathcal{E}^{\gamma}$, if $P_j \notin \mathcal{E}^{\gamma}$ and P_j has at least $2t+1$ neighbors in $\mathcal{F}^{\gamma}$ in G_S.
   (b) If no (n, t)-star is found or an (n, t)-star that has been already found in the past is obtained, then execute step (a).iv(A-B) to update existing $\mathcal{F}^{\gamma}$'s and $\mathcal{E}^{\gamma}$'s.
   (c) Now let β be the first index among already generated $\{(\mathcal{E}^1, \mathcal{F}^1), \ldots, (\mathcal{E}^{\delta}, \mathcal{F}^{\delta})\}$ such that both $\mathcal{E}^{\beta}$ and $\mathcal{F}^{\beta}$ contains at least $2t+1$ parties (Note that if step (a) is executed, then $\delta = \alpha + 1$; else $\delta = \alpha$). Assign $CORE = \mathcal{E}^{\beta}$ and A-cast $(\mathcal{C}^{\beta}, \mathcal{D}^{\beta}, \mathcal{E}^{\beta}, \mathcal{F}^{\beta})$.

Code for P_i including S.

1. Assign $CORE = \mathcal{E}^{\beta}$, when all the following events occur: (a) $(\mathcal{C}^{\beta}, \mathcal{D}^{\beta}, \mathcal{E}^{\beta}, \mathcal{F}^{\beta})$ is received from the A-cast of S; (b) $(\mathcal{C}^{\beta}, \mathcal{D}^{\beta})$ becomes a valid (n, t)-star in G_i; (c) every party $P_j \in \mathcal{F}^{\beta}$ has at least $t+1$ neighbors in $\mathcal{C}^{\beta}$ in G_i; and (d) every party $P_j \in \mathcal{E}^{\beta}$ has at least $2t+1$ neighbors in $\mathcal{F}^{\beta}$ in G_i.

S-independent Phase:

Code for P_i including S.

1. If $CORE$ is constructed and $P_i \in CORE$, then assign $s_i = s_{ii}$.
2. If $CORE$ is constructed and $P_i \notin CORE$, then assign s_i to be the value s_{ji} that is received from at least $t+1$ P_j's in $CORE$.
3. Send s_i to all the parties.
4. On receiving $2t+1+r$, $r \geq 0$, s_j's apply DEC with $c = r$ and $d = t - r$, if DEC return 'failure', then wait for more values. If DEC returns data elements $m_0, \ldots, m_t$, then output $m = m_0 | m_1 | \ldots | m_t$, where $|$ denotes concatenation.

every party in $\mathcal{E}$ has at least $2t+1$ neighbors in $\mathcal{F}$. To prove this we start with proving that an honest S will eventually find an (n,t)-star such that the set $\mathcal{C}$ will contain at least $t+1$ honest parties. For an honest S, eventually the edges between each pair of honest parties will vanish from the complementary graph $\overline{G_S}$. So the edges in $\overline{G_S}$ will be either (a) between an honest and a corrupted party OR (b) between two corrupted parties. Let β be the first index, such that (n,t)-star $(\mathcal{C}^\beta, \mathcal{D}^\beta)$ is generated in $\overline{G_S}$, when $\overline{G_S}$ contains edges of above two types only. Now, by construction of $\mathcal{C}^\beta$ (see Algorithm Find-STAR), it excludes the parties in N (set of parties that are endpoints of the edges of maximum matching M) and T (set of parties that are triangle-heads). An honest P_i belonging to N implies that $(P_i, P_j) \in M$ for some P_j and hence P_j is corrupted (as the current $\overline{G_S}$ does not have edge between two honest parties). Similarly, an honest party P_i belonging to T implies that there is some $(P_j, P_k) \in M$ such that (P_i, P_j) and (P_j, P_k) are edges in $\overline{G_S}$. This clearly implies that both P_j and P_k are surely corrupted. So for every honest P_i *not* in $\mathcal{C}^\beta$, at least one (if P_i belongs to N, then one; if P_i belongs to T, then two) corrupted party also remains outside $\mathcal{C}^\beta$. As there are at most t corrupted parties, $\mathcal{C}^\beta$ may exclude at most t honest parties. Still $\mathcal{C}^\beta$ is bound to contain at least $t+1$ honest parties.

Now all honest parties will be neighbors of the $t+1$ honest parties in $\mathcal{C}^\beta$ in G_S. Therefore $\mathcal{F}^\beta$ will eventually contain all the honest parties. Finally since all honest parties are neighbors of each other, $\mathcal{E}^\beta$ will contain all honest parties eventually and therefore it is guaranteed to contain at least $2t+1$ parties. Hence we proved that S can find a quadruple $(\mathcal{C}, \mathcal{D}, \mathcal{F}, \mathcal{E})$ eventually such that $(\mathcal{C}, \mathcal{D})$ is an (n,t)-star and every party in $\mathcal{F}$ has at least $t+1$ neighbors in $\mathcal{C}$ and every party in $\mathcal{E}$ has at least $2t+1$ neighbors in $\mathcal{F}$. S now A-casts the quadruple.

We now argue that every honest party will find $(\mathcal{C}, \mathcal{D}, \mathcal{F}, \mathcal{E})$ in their graphs and agree on the same. Though the graphs are constructed and maintained by parties individually in their local memory, it is always guaranteed that if an edge appears in the graph of an honest party, then the edge will eventually appear in the graphs of the other honest parties. This is ensured since the graphs are updated based on the responses of the parties that are A-casted. The termination and correctness of A-cast guarantee that if the A-cast is terminated by some honest party with some output then eventually every honest party will terminate with the same output. The above argument also shows that if some honest party agree on $CORE$, then eventually all honest parties will also agree on the same.

So we proved that all the honest parties will terminate **S-dependent Phase**, after agreeing on $CORE = \mathcal{E}$. ■

Lemma 2.4 *If the honest parties initiate* **S-independent Phase***, then they will terminate the phase with the common message of the parties in $CORE$ as the output.*

Proof: An honest party initiates **S-independent Phase**, if he agrees on $CORE$. By Lemma 2.2, all the honest parties in $CORE$ hold common message, say m of length ℓ and therefore same codeword $(s_1, \ldots, s_n) = \mathsf{ENC}(m_0, m_1, \ldots, m_t)$, where $m = m_0|m_1|\ldots,|m_t$. Then every honest P_i in $CORE$ already holds s_i, the ith element in the codeword. Every party P_i *not* in $CORE$ would receive s_i from the $t+1$ honest parties of $CORE$. Therefore every honest P_i will eventually hold the ith component of the codeword. Now every P_i send his s_i to every other party. Now on receiving at least $2t+1+r$, $0 \le r \le t$ s_j's, party P_i applies DEC with $c = r$ and $d = t-r$. Note that $c + d = t$, where t is the maximum number of corruption. Therefore if there are more than r wrong values (sent by Byzantine corrupted parties), DEC will return 'failure'. However for at least one value of r, $0 \le r \le t$, there will be at most r errors in the received vector and then the message can be reconstructed back successfully. This technique has been previously used in [CR93, Can95]. They call it as *Online Error Correction.* ■

Theorem 2.2 *Multi-Valued-Acast is an A-cast protocol satisfying Definition 1.1.*

Proof: We first consider the case of an honest S. By Lemma 2.3, for an honest S all the parties terminate **S-dependent Phase**, after agreeing on $CORE$. By Lemma 2.2, the honest parties in $CORE$ hold the message of S, i.e. m. By Lemma 2.4, all honest parties will terminate with the common message of $CORE$ as the output, which is m.

For a corrupted S, all we need to show is that if some honest party terminates with message $m^\star$, then every other honest party do the same. Let P_i be the first honest party to terminate the protocol with $m^\star$ as output. Then P_i must have agreed on $CORE$ and the parties in $CORE$ holds $m^\star$. Then every other honest party will agree on the same $CORE$ and eventually terminate with $m^\star$ as the output (by Lemma 2.4). ■

Theorem 2.3 *Multi-Valued-Acast communicates $\mathcal{O}(n\ell)$ bits and invokes $\mathcal{O}(n^2 \log n)$ A-cast protocol for single bit.*

Proof: S communicates his message m, $|m| = \ell$ to all the parties. This requires $n\ell$ bits of communication. Every party P_i sends two values s_{ii} and s_{ij} to every other party P_j. The values are $\frac{\ell}{t+1}$ bits long each. Therefore in total there are $\frac{\ell}{t+1}\mathcal{O}(n^2) = \mathcal{O}(n\ell)$ bits of communication.

S A-casts $(\mathcal{C}, \mathcal{D}, \mathcal{F}, \mathcal{E})$. Each set in the quadruple can be represented by an n length bit vector. Therefore $4n$ invocations to A-cast protocol for single bit are required. Finally every party may A-cast `OK` signal for every other party. Each `OK` signal includes identities of two parties that can be represented by $2\log n$ bits. Therefore $\mathcal{O}(n^2 \log n)$ invocations to A-cast protocol for single bit are required. ■

We note that for an $(n, t+1)$ RS code, the field $\mathbb{F} = GF(2^c)$ in which the code is defined should satisfy $n \leq 2^c$ or $\log n \leq c$. In our case $c = \frac{\ell}{t+1}$ (recall that m is divided into $t+1$ parts each containing $\frac{\ell}{t+1}$ bits). Therefore $\frac{\ell}{t+1} \geq \log n \rightarrow \ell \geq (t+1)\log n$. Finally using Bracha's A-cast [Bra84], the communication complexity of our protocol turns out to be $\mathcal{O}(n\ell + n^4 \log n)$. So for all $\ell \geq n^3 \log n$, our protocol achieves optimal communication bound of $\Omega(n\ell)$ bits.

3 Error-free ABA with Optimal Communication Complexity

In this section, we present our ABA protocol. We use our multi-valued A-cast protocol Multi-Valued-Acast from the previous section as one of the sub-protocols. Similar to Multi-Valued-Acast that uses A-cast protocol for single bit, our new ABA uses existing error-free ABA for single bit as another sub-protocol. In fact we use a very well-known asynchronous primitive called Agreement on Common Subset (ACS) introduced by [BKR94] that uses ABA for single bit as black box. We recall that the only error-free ABA is due to [ADH08]. We will use the following notation for invoking Multi-Valued-Acast.

Notation 3.1 *By saying that 'P_i Multi-casts M', we mean that P_i as a sender, initiates Multi-Valued-Acast protocol with M as the message. Similarly 'P_j receives M from the Multi-cast of P_i' will mean that P_j terminates the execution of Multi-Valued-Acast protocol initiated by P_i and outputs M. By the property of Multi-Valued-Acast, if some honest party P_j terminates the Multi-Valued-Acast protocol of some sender P_i with M as the output, then every other honest party will eventually do so, irrespective of the behavior of the sender P_i.*

Agreement on a Common Subset (ACS). Consider the following scenario. The parties in $\mathcal{P}$ are asked to A-cast (or Multi-cast) some value. While the honest parties in $\mathcal{P}$ will eventually execute the A-cast (Multi-cast), the corrupted parties may or may not do the same. So the (honest) parties in $\mathcal{P}$ want to agree on a *common* set $\mathcal{T} \subset \mathcal{P}$, with $|\mathcal{T}| = 2t+1$, such that A-cast (Multi-cast) of each party in $\mathcal{T}$ will be eventually terminated by the (honest) parties in $\mathcal{P}$. For this, the parties use ACS primitive presented in [BKR94]. The ACS protocol uses n instances of ABA for single bit. For the sake of completeness, the protocol is provided in Appendix A.

3.1 Multi-valued ABA Protocol

Given the above sub-protocols, our ABA is very simple. Every party P_i on having a message m_i of length ℓ, computes an n length codeword $(s_{i1}, \ldots, s_{in}) = \mathsf{ENC}(m_{i0}, \ldots, m_{it})$ where $m_{i0}| \ldots |m_{it}$. P_i Multi-casts s_{ii}. Using ACS, the parties then agree on some subset of $2t+1$ parties, say $\mathcal{X}$ whose Multi-casts will be terminated eventually. Every party then verifies if the values Multi-casted by the parties in $\mathcal{X}$ match with their corresponding elements of the codeword and then A-cast their response. The parties again agree on some subset of $2t+1$ parties using ACS, say $\mathcal{Y}$. Based on the responses of the parties in $\mathcal{Y}$ and the values Multi-casted by the parties in $\mathcal{X}$, the agreement is reached. Note that we use Multi-cast for the elements of the codewords (i.e. s_{ii}'s) and A-cast for the responses. The reason is that s_{ii}'s are message dependent and therefore can be arbitrarily large. Therefore by appropriately setting the value of ℓ, we can implement Multi-casting of s_{ii} values in $\mathcal{O}(n\ell)$ overall complexity. However, we will see from the protocol given below, the response vector will be always n length bit vector. Therefore, using Multi-cast for this case will worsen the complexity, as compared to the case when A-cast of Bracha is used for the same purpose. The protocol is now presented in Figure 3 and its properties are proved subsequently.

Theorem 3.1 *Protocol Multi-Valued-ABA is an ABA protocol.*

Figure 3: Error-free Multi-valued ABA with Optimal Communication Complexity.

Protocol Multi-Valued-ABA()

Code for P_i.

1. On having message m_i, divide the ℓ bit message m_i into $t+1$ blocks, $m_{i0}, \ldots, m_{it}$, each containing $\frac{\ell}{t+1}$ (assume this to be an integer) bits. Compute $(s_{i1}, \ldots, s_{in}) = \mathsf{ENC}(m_{i0}, \ldots, m_{it})$.
2. Multi-cast s_{ii}.
3. Participate in an instance of ACS to agree on a common subset of $2t+1$ parties, $\mathcal{X}$ whose Multi-cast will be eventually terminated by all honest parties.
4. Construct a binary vector V_i of length n. Assign $V_i[j] = 1$, if $P_j \in \mathcal{X}$ and $s_{ij} = s_{jj}$ where s_{jj} received from the Multi-cast of P_j. Otherwise assign $V_i[j] = 0$.
5. A-cast V_i.
6. Participate in an instance of ACS to agree on a common subset of $2t+1$ parties, $\mathcal{Y}$ whose A-cast will be eventually terminated by all honest parties.
7. Check if there are at least $t+1$ parties in $\mathcal{Y}$, whose vectors are identical and have at least $t+1$ 1's. Let $\{i_1, \ldots, i_{i+1}\} \subseteq \mathcal{X}$ be the $t+1$ minimum indices where they all have 1's. If there is no such set of $t+1$ parties in $\mathcal{Y}$, then $\{i_1, \ldots, i_{i+1}\}$ be the $t+1$ minimum indices in $\mathcal{X}$. Then apply DEC on $s_{i_1, i_1}, \ldots, s_{i_{t+1} i_{t+1}}$ and let $m_0, m_1, \ldots, m_t$ be the data returned by DEC. Then output $m = m_0 | \ldots | m_t$.

Proof: The termination is guaranteed due to the termination properties of Multi-Valued-Acast, A-cast protocol of Bracha [Bra84] and ACS (whose termination is guaranteed due to the termination of the underlying ABA for single bit). More specifically, since Multi-Valued-Acast initiated by the honest parties will always eventually terminate, the set $\mathcal{X}$ will be agreed among the parties by the termination of ACS. Similarly, since A-cast (of Bracha) initiated by the honest parties will always eventually terminate, the set $\mathcal{Y}$ will be agreed among the parties by the termination of ACS. Once $\mathcal{X}$ and $\mathcal{Y}$ are agreed, the rest is local computation. Therefore termination of Multi-Valued-ABA is guaranteed.

We now argue about the correctness of Multi-Valued-ABA. First we show that all the honest parties will agree on the same message. This follows from the fact that all the honest parties agree on $\{i_1, \ldots, i_{t+1}\} \subseteq \mathcal{X}$ in the last step of the protocol. Furthermore, by the correctness property of Multi-Valued-Acast, all the honest parties also agree on the values Multi-casted by the parties in $\{P_{i_1}, \ldots, P_{i_{t+1}}\}$. Our claim now follows trivially. We now consider the case when all the honest parties start with same input message m of length ℓ and argue that all honest parties will agree on m eventually. If all the honest parties start with m, then they generate $(s_1, \ldots, s_n) = \mathsf{ENC}(m_0, \ldots, m_t)$ locally, where $m = m_0 | \ldots | m_t$. Every honest party P_i then Multi-casts s_i. By the property of Multi-Valued-Acast, all the parties will receive the same value Multi-casted by the parties in $\mathcal{X}$. Therefore the honest parties in $\mathcal{Y}$ will have identical V_i vectors. Furthermore the V_i vectors will have 1's at least at $t+1$ locations corresponding to the parties in $\mathcal{X}$ who Multi-casted correct value from the codeword $(s_1, \ldots, s_n)$. So $\{i_1, \ldots, i_{t+1}\} \subseteq \mathcal{X}$ in the last step of the above protocol denotes $t+1$ identities of the parties in $\mathcal{X}$ (having $t+1$ minimum indices) who Multi-casted correct value from the codeword $(s_1, \ldots, s_n)$. Therefore DEC when applied on the values Multi-casted by the parties $\{P_{i_1}, \ldots, P_{i_{t+1}}\}$ will return $m_0, m_1, \ldots, m_t$ such that $m = m_0 | \ldots | m_t$. ■

Theorem 3.2 *Protocol Multi-Valued-ABA communicates $\mathcal{O}(n\ell)$ bits, invokes $\mathcal{O}(n^3 \log n)$ instances of A-cast for single bit and invokes $2n$ instances of ABA for single bit.*

Proof: Every party Multi-casts $\frac{\ell}{t+1}$ bits. This requires a total communication of $\mathcal{O}(n\ell)$ bits and $\mathcal{O}(n^3 \log n)$ invocations to A-cast for single bit. Later every party A-casts an n length binary vector. Therefore in total n^2 invocations to A-cast for single bit is required. Finally two invocations to ACS calls for $2n$ instances of ABA for single bit. ■

To make the underlying protocol Multi-Valued-Acast work correctly, we require $\frac{\ell}{t+1} \geq (t+1)\log n$. Recall that when the input message size for Multi-Valued-Acast is ℓ, then we require that $\ell \geq (t+1)\log n$. In our ABA protocol, the input to Multi-valued-Acast is $\frac{\ell}{t+1}$. Therefore we have $\frac{\ell}{t+1} \geq (t+1)\log n \rightarrow \ell \geq (t+1)^2 \log n$ for our ABA.

4 Error-free BA and BC with Optimal Communication Complexity

Here we present our new multi-valued BA and BC protocol. We first present a BA protocol. A BC protocol with same complexity of the BA protocol can be achieve by letting the sender send the message to all the parties and then BA can be run to reach agreement. This is the standard reduction in synchronous settings from BA to BC [Lyn96]. In our BA protocol, we use the linear error correcting code and the algorithm for (n,t)-star presented in Section 2. We also use existing BC protocol for single bit with optimal complexity of $\mathcal{O}(n^2)$ bits [BGP09, CW92]. Our BA protocol follows the idea of Multi-Valued-Acast. We now present the protocol in Figure 4 and subsequently prove the properties.

Figure 4: Error-free Multi-valued BA with Optimal Communication Complexity.

Protocol Multi-Valued-BA()

Code for P_i.

1. On having message m_i, divide the ℓ bit message m_i into $t+1$ blocks, $m_{i0},\ldots,m_{it}$, each containing $\frac{\ell}{t+1}$ (assume this to be an integer) bits. Compute $(s_{i1},\ldots,s_{in}) = \mathsf{ENC}(m_{i0},\ldots,m_{it})$.
2. Send s_{ii} to every party. Send s_{ij} to P_j for $j=[1,n]$.
3. Construct a binary vector V_i of length n. Assign $V_i[j]=1$, if $s_{ij}=s_{jj}$ and $s_{ii}=s_{ji}$ where s_{jj} and s_{ji} are received from P_j. Otherwise assign $V_i[j]=0$.
4. Broadcast V_i using BC protocol for single bit.
5. Construct graph G_i using parties in $\mathcal{P}$ as the vertices. Add edge (P_j,P_k) if $V_j[k]=1$ and $V_k[j]=1$. Execute Find-STAR$(\overline{G_i})$. If `star-Not-Found` is returned, then set $b_i=0$. Else let $(\mathcal{C}_i,\mathcal{D}_i)$ be the (n,t)-star returned by Find-STAR. Let $\mathcal{F}_i$ be the set of parties who have at least $t+1$ neighbors in $\mathcal{C}_i$ in graph G_i. Let $\mathcal{E}_i$ be the set of parties who have at least $2t+1$ neighbors in $\mathcal{F}_i$ in graph G_i. If $|\mathcal{F}_i|\geq 2t+1$ and $|\mathcal{E}_i|\geq 2t+1$, then set $b_i=1$, else set $b_i=0$.
6. Broadcast only b_i when $b_i=0$; else broadcast b_i and $(\mathcal{C}_i,\mathcal{D}_i,\mathcal{F}_i,\mathcal{E}_i)$ using BC protocol for single bit.
7. If $t+1$ b_j's are zero, then agree on some predefined message $m^\star$ of length ℓ. Else let α be the minimum index of the party where $b_\alpha=1$ and $(\mathcal{C}_\alpha,\mathcal{D}_\alpha,\mathcal{F}_\alpha,\mathcal{E}_\alpha)$ be such that $(\mathcal{C}_\alpha,\mathcal{D}_\alpha)$ is an (n,t)-star in G_i, every party in $\mathcal{F}_\alpha$ has at least $t+1$ neighbors in $\mathcal{C}_\alpha$ and every party in $\mathcal{E}_\alpha$ has at least $2t+1$ neighbors in $\mathcal{F}_\alpha$. Assign $CORE=\mathcal{E}_\alpha$.
8. Assign s_i to be the value s_{ji} received from the majority of the parties in $CORE$.
9. Send s_i to every party.
10. Let $(s_1,\ldots,s_n)$ be the vector where s_j is received from P_j. Apply DEC on $(s_1,\ldots,s_n)$ with $c=t$ and $d=0$. Let $m_0,m_1,\ldots,m_t$ be the data returned by DEC. Output $m=m_0|\ldots|m_t$.

Lemma 4.1 *The honest parties in $CORE$ hold same message of length ℓ.*

The proof of the above lemma completely follow from the proof of Lemma 2.2.

Lemma 4.2 *If all honest parties start with same input m, then all the parties will agree on $CORE$, $|CORE|\geq 2t+1$.*

Proof Sketch: The proof here follows from the proof of Lemma 2.3. Briefly, when all honest parties start with same input, every pair of honest parties will have edge between them. In other words, the edges in the complementary graph will be either (a) between an honest and a corrupted party OR (b) between two corrupted parties. Therefore following the argument given in Lemma 2.3, $\mathcal{C}$ component of an (n,t)-star will contain at least $t+1$ honest parties, which subsequently will lead to the construction of $\mathcal{F}$ and $\mathcal{E}$ with size at least $2t+1$. Although it is not guaranteed that all honest parties find same quadruple $(\mathcal{C},\mathcal{D},\mathcal{F},\mathcal{E})$, but it is ensured that they will find some quadruple. So the honest parties never agree on predefined $m^\star$ in this case. Now since all the parties broadcast their quadruple, it is easy to reach agreement on a valid quadruple which the parties do by selecting the one broadcasted by the party with minimum index. Therefore all the parties will agree on $CORE$. ■

Lemma 4.3 *If $CORE$ is agreed, all honest parties output the common message of the parties in $CORE$.*

Proof Sketch: By Lemma 4.1, all honest parties in $CORE$ hold same message, say m. The proof now follows from the proof of Lemma 2.4. We still brief the proof here. Let $(s_1, \dots, s_n) = \mathsf{ENC}(m_0, \dots, m_t)$, where $m = m_0| \dots |m_t$. First note that all honest party in $CORE$ hold m and therefore the codeword $(s_1, \dots, s_n)$. Now every party P_i will receive s_i correctly as majority of the parties in $CORE$ are honest and they will send s_i to P_i. Once every honest P_i holds correct s_i, he sends that to everybody. Therefore a party will receive n values from n parties in which at most t can be wrong (sent by Byzantine corrupted party). However, DEC of $(n, t+1)$ RS code with $n = 3t+1$ allows to correct t errors. Therefore DEC will return $m_0, \dots, m_t$ such that $m = m_0| \dots |m_t$. ■

Theorem 4.1 *Multi-Valued-BA is a BA protocol.*

Proof: If $CORE$ is agreed, then all honest parties will output the common message of the parties in $CORE$ (by Lemma 4.3). If $CORE$ is not agreed, then there must be at least $t+1$ parties who broadcasted $b_i = 0$. Since b_i's are broadcasted, all honest parties will agree on predefined $m^\star$ of length. So agreement is always achieved at the end.

Now if all the honest parties start with same input message m, then they will agree on $CORE$ (by Lemma 4.2) and output m at the end (by Lemma 4.3). ■

Theorem 4.2 *Multi-valued-BA communicates $\mathcal{O}(n\ell)$ bits and invokes $\mathcal{O}(n^2)$ broadcast protocol for single bit.*

Proof: Every party P_i sends two values s_{ii} and s_{ij} to every other party P_j. The values are $\frac{\ell}{t+1}$ bits long each. Therefore in total there are $\frac{\ell}{t+1}\mathcal{O}(n^2) = \mathcal{O}(n\ell)$ bits of communication.

Every party P_i broadcasts n-length binary vector V_i, a bit b_i and quadruple $(\mathcal{C}_i, \mathcal{D}_i, \mathcal{F}_i, \mathcal{E}_i)$. Each set in the quadruple can be represented by n-length bit vector. Therefore every party invokes $5n+1$ instances of broadcast for single bit. This leads to total $\mathcal{O}(n^2)$ instances of broadcast for single bit. ■

We note that for an $(n, t+1)$ RS code, the field $\mathbb{F} = GF(2^c)$ in which the code is defined should satisfy $n \leq 2^c$ or $\log n \leq c$. In our case $c = \frac{\ell}{t+1}$ (recall that m is divided into $t+1$ parts each containing $\frac{\ell}{t+1}$ bits). Therefore $\frac{\ell}{t+1} \geq \log n \rightarrow \ell \geq (t+1)\log n$. Finally using bit optimal broadcast protocols of [BGP09, CW92] (they require $\mathcal{O}(n^2)$ bits of communication), the communication complexity of our protocol turns out to be $\mathcal{O}(n\ell + n^4)$. So for all $\ell \geq n^3$, our protocol achieves optimal communication bound of $\Omega(n\ell)$ bits.

5 Conclusion

In this paper we present first ever *error-free, asynchronous* multi-valued A-cast and ABA protocols with optimal communication complexity and fault tolerance. Our protocols are reduction-based. Our reductions are constant time and invokes equal or less number of protocols for single bit as compared to existing reductions with error probability [PR10, PR11]. Using our technique in asynchronous settings, we also present new reduction for error-free multi-valued BA and BC protocols that are constant-round and calls for only $\mathcal{O}(n^2)$ instances of protocol for single bit. This can be compared with the existing error-free reduction of [LV11] which calls for round complexity of $\mathcal{O}(\sqrt{\ell}+n^2)$ and $\mathcal{O}(n^2\sqrt{\ell}+n^4)$ invocations to existing protocol for single bit. In [LV11], the communication optimality is achieved for $\ell = \Omega(n^6)$. For our reduction we show that for all $\ell = \Omega(n^3)$ the reduction achieves optimality.

An important open question is to see whether we can achieve multi-valued protocols with less number of invocations to protocols for single bit in comparison to what we provide in this paper. Also it is important to improve the complexity of error-free protocols for single bit. In the case of BA, BC and A-cast, optimal communication complexity has been achieved [BGP09, CW92, Bra84]. However the same is not the case for error-free ABA. The only error-free ABA is due to [ADH08] that runs in $\mathcal{O}(n^2)$ time and requires communication of $\mathcal{O}(n^8 \log n)$ bits and A-cast of same number of bits. We observe that the running time of the protocol can be brought down by a factor of n and the communication complexity by a factor of n^2. But as mentioned in [ADH08], it is still a major open problem to design a constant time protocol with good communication complexity or even to prove whether such protocol exists or not.

References

[ADH08] I. Abraham, D. Dolev, and J. Y. Halpern. An almost-surely terminating polynomial protocol for asynchronous Byzantine Agreement with optimal resilience. In *PODC*, pages 405–414. ACM Press, 2008.

[BGP09] P. Berman, G. A. Garay, and K. J. Perry. Bit optimal distributed consensus. In Computer Science Research, 2009.

[BKR94] M. BenOr, B. Kelmer, and T. Rabin. Asynchronous secure computations with optimal resilience. In *PODC*, pages 183–192. ACM Press, 1994.

[BO83] M. Ben-Or. Another advantage of free choice: Completely asynchronous agreement protocols. In *PODC*, pages 27–30. ACM Press, 1983.

[BOCG93] M. Ben-Or, R. Canetti, and O. Goldreich. Asynchronous Secure Computation. In *STOC*, pages 52–61. ACM Press, 1993.

[BOGW88] M. Ben-Or, S. Goldwasser, and A. Wigderson. Completeness theorems for non-cryptographic fault-tolerant distributed computation (extended abstract). In *STOC*, pages 1–10. ACM Press, 1988.

[Bra84] G. Bracha. An asynchronous $\lfloor (n-1)/3 \rfloor$-resilient consensus protocol. In *PODC*, pages 154 – 162. ACM Press, 1984.

[BTH06] Z. Beerliová-Trubíniová and M. Hirt. Efficient multi-party computation with dispute control. In *TCC*, LNCS 3876, pages 305–328. Springer Verlag, 2006.

[Can95] R. Canetti. *Studies in Secure Multiparty Computation and Applications*. PhD thesis, Weizmann Institute, Israel, 1995.

[CGMA85] B. Chor, S. Goldwasser, S. Micali, and B. Awerbuch. Verifiable secret sharing and achieving simultaneity in the presence of faults (extended abstract). In *STOC*, pages 383–395. ACM Press, 1985.

[CR93] R. Canetti and T. Rabin. Fast asynchronous Byzantine Agreement with optimal resilience. In *STOC*, pages 42–51. ACM Press, 1993.

[CW92] B. A. Coan and J. L. Welch. Modular construction of a Byzantine Agreement protocol with optimal message bit complexity. *Information and Computation*, 97(1):61–85, 1992.

[DR85] D. Dolev and R. Reischuk. Bounds on information exchange for Byzantine Agreement. *Journal of ACM*, 32(1):191–204, 1985.

[FH06] M. Fitzi and M. Hirt. Optimally Efficient Multi-valued Byzantine Agreement. In *PODC*, pages 163–168, 2006.

[FLP85] M. J. Fischer, N. A. Lynch, and M. Paterson. Impossibility of distributed consensus with one faulty process. *JACM*, 32(2):374–382, 1985.

[FM88] P. Feldman and S. Micali. An Optimal Algorithm for Synchronous Byzantine Agreemet. In *STOC*, pages 639–648. ACM Press, 1988.

[GJ79] M. R. Garey and D. S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W. H. Freeman, 1979.

[HMP00] M. Hirt, U. Maurer, and B. Przydatek. Efficient Secure Multiparty Computation. In *ASIACRYPT*, LNCS 1976, pages 143–161. Springer Verlag, 2000.

[LV11] G. Liang and N. H. Vaidya. Error-Free Multi-Valued Consensus with Byzantine Failures. In *Accepted in PODC*, 2011.

[Lyn96] N. A. Lynch. *Distributed Algorithms*. Morgan Kaufmann, 1996.

[PR10] A. Patra and C. Pandu Rangan. Communication Optimal Multi-valued Asynchronous Broadcast Protocol. In *LATINCRYPT*, LNCS 6212, pages 162–177. Springer, 2010.

[PR11] A. Patra and C. Pandu Rangan. Communication Optimal Multi-valued Asynchronous Byzantine Agreement with Optimal Resilience. In *ICITS*, LNCS 6673, pages 206–226. Springer, 2011.

[PSL80] M. Pease, R. E. Shostak, and L. Lamport. Reaching agreement in the presence of faults. *JACM*, 27(2):228–234, 1980.

[PW92] B. Pfitzmann and M. Waidner. Unconditional Byzantine Agreement for any number of faulty processors. In *STACS*, LNCS 577, pages 339–350. Springer Verlag, 1992.

[Rab83] M. O. Rabin. Randomized Byzantine generals. In *FOCS*, pages 403–409. IEEE Computer Society, 1983.

[RBO89] T. Rabin and M. Ben-Or. Verifiable secret sharing and multiparty protocols with honest majority (extended abstract). In *STOC*, pages 73–85. ACM Press, 1989.

[TC84] R. Turpin and B. A. Coan. Extending binary Byzantine Agreement to multivalued Byzantine Agreement. *Information Processing Letters*, 18(2):73–76, 1984.

A Bracha's A-cast Protocol and ACS Protocol

Bracha's A-cast Protocol. For the sake of completeness, we recall Bracha's A-cast protocol [Bra84] from [Can95] and present it in Figure 5.

Figure 5: Bracha's A-cast Protocol for $n = 3t + 1$

Bracha-A-cast(S, M)

Code for the sender S (with input M): only S executes this code

1. Send message (MSG, M) privately to all the parties.

Code for party P_i: every party in $\mathcal{P}$ executes this code

1. Upon receiving a message (MSG, M), send $(ECHO, M)$ privately to all parties.
2. Upon receiving $n - t$ messages $(ECHO, M')$ that agree on the value of M', send $(READY, M')$ privately to all the parties.
3. Upon receiving $t + 1$ messages $(READY, M')$ that agree on the value of M', send $(READY, M')$ privately to all the parties.
4. Upon receiving $n - t$ messages $(READY, M')$ that agree on the value of M', send (OK, M') privately to all the parties, accept M' as the output message and terminate the protocol.

Theorem A.1 *Protocol Bracha-A-cast privately communicates $\mathcal{O}(\ell n^2)$ bits for an ℓ bit message.*

ACS Protocol. We now recall ACS protocol [BKR94].

Theorem A.2 ([BKR94]) *Using protocol ACS, the (honest) parties in $\mathcal{P}$ can agree on a common subset $\mathcal{T}$ of $\mathcal{P}$ containing $1 \leq |\mathcal{T}| \leq |\mathcal{P}| - t$ parties, whose instances of A-cast (Multi-cast) will be eventually terminated by all the (honest) parties in $\mathcal{P}$.*

Theorem A.3 *The communication complexity of ACS is equal to $|\mathcal{P}| = n$ executions of ABA protocol for single bit.*

Figure 6: Protocol for Agreement on a Common Subset with $n = 3t + 1$

Protocol ACS($\mathcal{P}, |\mathcal{T}|$)

Code for Party P_i: Each party in $\mathcal{P}$ executes this code

1. For each $P_j \in \mathcal{P}$ whose instance of A-cast (Multi-cast) is terminated by you, participate in ABA_j with input 1. Here for $j = 1, \ldots, |\mathcal{P}|$, ABA_j denotes the instance of ABA executed on behalf of $P_j \in \mathcal{P}$ to decide whether $P_j \in \mathcal{T}$.
2. Upon terminating $|\mathcal{T}|$ instances of ABA with output 1, enter input 0 to all other instances of ABA, for which you haven't entered a value yet.
3. Upon terminating all the $|\mathcal{P}|$ ABA protocols, let your $SubSet_i$ be the set of all indices j for which ABA_j had output 1.
4. Let $\mathcal{T}_i$ be the set of $|\mathcal{T}|$ parties corresponding to smallest $|\mathcal{T}|$ indices in $SubSet_i$. Output $\mathcal{T}_i$ and terminate.