2017 IEEE 22nd Pacific Rim International Symposium on Dependable Computing

Voting in the Presence of Byzantine Faults

Lewis Tseng
Department of Computer Science
University of Illinois at Urbana-Champaign
Email: ltseng3@illinois.edu

Abstract—**Voting (or election) algorithms are used widely in many safety-critical systems to mask errors. Most systems only tolerate malicious (or Byzantine) voters – these systems assume the existence of a correct and centralized mechanism to collect the votes and propagate the voting output to each voter. However, in many realistic scenarios, such a centralized voting mechanism is not feasible. Thus, we study the *Byzantine voting* problem – no centralized mechanism exists in the system, and voters may become Byzantine faulty. We first present impossibility results in both synchronous and asynchronous systems. To circumvent the impossibility results presented in this paper, we propose two *relaxed* voting properties that are achievable and present *optimal* voting algorithms that satisfy the relaxed properties. Finally, we show that it is possible to design Byzantine voting algorithms that produce the voting output in *one communication step* under contention-free scenarios.**

Keywords – Byzantine; Voting; Optimal Algorithms; Impossibility Results; Distributed Algorithms; Consensus

I. Introduction

Fault-masking by voting is one of the major approaches to ensure reliability in many safety-critical systems [25]. Voting (or election) algorithms over redundant nodes (or modules/components) are used to hide the occurrence of erroneous inputs or faulty behavior from the system output.[1] In most prior work, the systems using voting algorithms assume the existence of a centralized voting mechanism to collect the votes and then choose and propagate the voting output [25], [34]. However, in large-scale distributed systems like sensor networks, peer-to-peer systems or even fleets of autonomous trucks, it is not practical to implement a centralized voting mechanism. This observation motivates our study of the *Byzantine voting* problem [12], where a centralized mechanism is not feasible, and some nodes many become Byzantine faulty and have arbitrary behaviors. The goal is to develop distributed voting algorithms in which nodes will agree on an *identical* output that depends on their *inputs* (or *votes*).

In the *Byzantine voting* problem [12], a node proposes an input (or a *vote*) initially. Then, after a finite amount of time, all fault-free nodes terminate with an identical output. The output must satisfy the *voting property* as discussed later in Section I-B. Even though the voting problem has been studied extensively in the fault-free case (all nodes behave correctly) [3], [17], [16], [4], the Byzantine voting problem is not well studied in the literature. On the one hand, the properties of fault-free voting algorithms do not extend to the Byzantine voting problem naturally, since Byzantine faulty nodes may behave arbitrarily. Note that the *strategy-proofness* property in fault-free voting algorithms [21], [16] is orthogonal to the Byzantine voting problem, since by assumption, Byzantine nodes may not necessarily seek to optimize their utility. On the other hand, the correctness conditions of traditional fault-tolerant distributed problems are not adequate for the voting problem. For example, validity for consensus problem [7], [29] do not guarantee common voting properties like *majority rule* – output must be a value proposed by the majority of fault-free nodes. We discuss other related work in Section VI. The closest work is by Chauhan and Garg [12] who presented some early results on the Byzantine voting problem in synchronous systems. We generalize their results and have the following contributions:

- We present more general impossibility results in both synchronous and asynchronous systems. The impossibility result in [12] is a special case of Theorem 1.
- To circumvent the impossibility result (Theorem 1), we propose two relaxed voting properties that are achievable in both synchronous and asynchronous systems. We also discuss how these results imply the necessary number of nodes for solving meaningful tasks in practical safety-critical systems in Section VII.
- In Section III, we present a synchronous voting algorithm that uses binary Byzantine broadcast algorithms [37], [24], [7], [29] as a building block. For asynchronous systems, we present a voting algorithm that uses randomized multi-valued consensus algorithms [35] in Section IV. Our algorithms satisfy the proposed relaxed voting properties. Moreover, they are *optimal* with respect to the relaxed properties.
- Finally, we present results on *one-step* voting algorithms in Section V. These algorithms are designed to generate voting output in one communication step in the *contention-free* scenarios, i.e., when all fault-free nodes propose the same input (or vote). Note that such one-step voting algorithms are useful in safety-critical systems that

[1]In this paper, we use the term "node" to mean a single computing entity in the system – a "node" may be a module, component, or process.

978-1-5090-5652-1/17 $31.00 © 2017 IEEE
DOI 10.1109/PRDC.2017.11

use voting to mask faults, since when failure is rare, fault-free nodes will propose the same input in most cases and generate a voting output in one communication step.

Byzantine consensus [37], [24] is a special case of Byzantine voting [12]; hence, we borrow techniques/algorithms from the literature. One main contribution of our paper is carefully formalizing the voting problem, and working out the connection between Byzantine voting and practical voting mechanisms used in safety-critical systems [25], [34].

A. Byzantine Voting

Below, we discuss the Byzantine voting problem [12]. Consider a system of n nodes in an all-to-all message-passing network. Of these n nodes, up to f may suffer Byzantine failures, namely, *faulty* nodes. The other nodes – *fault-free* nodes – follow the algorithm specification throughout the execution of the algorithm. The input domain $\mathbb{C}$ refers to the set of possible input values. The size of the domain is denoted by k $(k \geq 2)$, i.e., $|\mathbb{C}| = k$. In the beginning of the voting algorithm, each node proposes an input value in $\mathbb{C}$. For clarity of discussion, we consider the case when each node proposes a *single* value instead of a linear order (or a permutation) on $\mathbb{C}$. The results can be generalized to the case when nodes propose linear orders that specify the preference level over values in $\mathbb{C}$. Then, the voting algorithm is *correct* if all the following conditions are satisfied:

- *Termination*: All fault-free nodes must terminate with a single output value in a finite amount of time.
- *Agreement*: Output at fault-free nodes must be identical.
- *Voting*: Output at fault-free nodes must satisfy the chosen voting property.

B. Voting Properties

We first present the following voting properties that are studied in economics [16], [4] and distributed computing literature [7], [29]:

- *Weak*: If $v \in \mathbb{C}$ is proposed by all fault-free nodes, then v must be the output.
- *Majority* (*Plurality*): If $v \in \mathbb{C}$ is proposed by *majority* (*plurality*) of fault-free nodes, then v must be the output (break tie arbitrarily).
- *Strong*: If v is the output, then v is proposed by some fault-free nodes.

Weak and Strong properties are from the traditional consensus problem [7], [29]; however, they are often *not* adequate for the voting problems, since they cannot be used to mask faults in safety-critical systems. For example, weak property allows the voting output to be a faulty input, whereas strong property allows the output to be a value that is proposed by a single fault-free node (even when there are more preferable values proposed by most fault-free nodes). Majority and Plurality properties were studied in fault-free voting problems [16], [4], and are natural properties for Byzantine voting as well.

It turns out that except for Weak property, it is impossible to achieve other properties in most cases. The impossibility of Strong property is implied by [33], which showed that to achieve multi-valued Byzantine consensus with strong validity (essentially identical to Strong property), it is necessary to have $n \geq \max\{k, 3\}f + 1$ to tolerate up to f faults. Recall that n and k are the number of nodes, and the domain size $|\mathbb{C}|$, respectively. Since Byzantine consensus is a special case of Byzantine voting, Strong property cannot be achieved when $n < \max\{k, 3\}f + 1$. We prove in this paper that Majority and Plurality properties are impossible to achieve when $f \geq 1$ and $k \geq 2$ in Theorem 1 (Section II). Note that reference [12] showed that Majority property is not achievable when $f \geq n/4$ and $k \geq 2$, which is subsumed by Theorem 1.

To circumvent these impossibility results, we propose two new relaxed voting properties:

- *l-Majority* (*l-Plurality*): the *error bound* between the output and the value proposed by a *majority* (*plurality*) of fault-free nodes is $\leq l$.
- *c-Verified*: If $v \in \mathbb{C}$ is proposed by $\geq c$ fault-free nodes, then v must be the output (break tie arbitrarily).

Suppose value v is proposed by the majority (or a plurality) of fault-free nodes, and value o is the output of an algorithm. Then, the *error bound* of the algorithm is the difference between the number of fault-free nodes proposing o and the number of fault-free nodes proposing v. Below, we illustrate the value l using an example. Consider a system of five fault-free nodes with inputs $1, 1, 1, 2,$ and 2. Value 1 is proposed by the majority of fault-free nodes; yet, 1-Majority property allows an algorithm to pick either 1 or 2 as an output, whereas, 0-Majority property (identical to Majority property) requires an algorithm to pick 1. In the later case, the error bound is $3 - 2 = 1$ if value 2 is picked. This violates 0-Majority. We formally define the value l and error bound in Section III.

Applications of Relaxed Properties: We believe that the two proposed relaxed properties have wide practical applications. First, voting algorithms with l-Majority (l-Plurality) property is useful for generating a "good-enough" output, since the output is guaranteed to receive enough votes from fault-free nodes. Such a "good-enough" design is inspired by the centralized voting algorithms used in many safety-critical systems, e.g., *2:3-way voting* or *median voting* [25] – all these algorithms produce outputs that are close (not necessarily identical) to the best result.

Second, the voting algorithms with c-Verified property can be used to filter out bad value(s) proposed by faulty nodes as long as enough fault-free nodes propose the same input value. Note that Weak property only ensures that no faulty input is chosen to be an output when *all* fault-free nodes propose the same input. Therefore, c-Verified property can be viewed as a generalization of Weak property and a relaxation of Strong property. c-Verified property is similar to *weighted voting* [25], [34] that are commonly adopted in safety-critical systems in the sense that all these properties require a pre-defined threshold for a value to be selected as the output. In Section VII, we compare these properties in more details.

C. Models

We now discuss the models, which are typical in the literature [7], [29], [25], [34].

a) Fault Model: In a system of n nodes, up to f nodes may suffer Byzantine failures. Byzantine nodes may behave arbitrarily. Possible misbehavior includes sending incorrect and mismatching (or inconsistent) messages to different nodes. Byzantine nodes may potentially collude with each other. Moreover, Byzantine nodes are assumed to have a complete knowledge of the execution of the algorithm, including the states of all the nodes, contents of messages the other nodes send to each other, and the algorithm specification. We assume that $n \geq 3f + 1$ nodes; otherwise, it is impossible to achieve Byzantine consensus [37], [24], which implies that it is impossible to solve the Byzantine voting problem with any aforementioned voting property.

b) Communication Model: Nodes communicate with each other by sending messages via a point-to-point network. And, each pair of nodes have a reliable and FIFO channel between them (i.e., all-to-all communication network). We assume that each channel is secure; as a result, Byzantine nodes cannot tamper nor forge the message sent by a fault-free node. Also, for each message, the receiver knows who the sender is. Therefore, for each pair of fault-free nodes, they are able to communicate with each other reliably.

c) System Synchrony: We consider both synchronous and asynchronous systems. In synchronous systems, nodes proceed in a lock-step fashion. That is, all the message delays and computation steps have a finite and known bound. In asynchronous systems, the bound is unknown. Yet, we assume asynchronous but reliable communication channels. In other words, messages may be arbitrarily delayed, but between two fault-free nodes, messages will eventually be delivered.

In addition, we assume a strong network adversary [39], [22]. That is, the network is also controlled by an adversary that may choose to arbitrarily delay messages as long as between any two fault-free nodes, messages are eventually delivered. The adversary can also choose to deliver messages to some node i *after* node i has decided its output [39], [22].

II. Main Results

A. Majority and Plurality Properties

We first prove the impossibility results of Majority and Plurality voting properties. Recall that $\mathbb{C}$ is the input domain. Note that the impossibility result in [12] is subsumed by the theorem below.

Theorem 1. *It is impossible to ensure Termination, Agreement, and Majority (Plurality) property at the same time when $f \geq 1$ and $|\mathbb{C}| = k \geq 2$.*

The reason behind the impossibility result is that Byzantine faulty nodes can manipulate the output by deliberating proposing (incorrect) inputs. Since fault-free nodes do not know the identity of faulty nodes, they are forced to choose a less favorable output in some circumstances.

We use the indistinguishability argument [6], [7], [29] to formally prove Theorem 1 in our technical report [40]. Here, we briefly discuss the proof intuition. By way of contradiction, we assume that there exists a correct voting algorithm $\mathbb{A}$ that achieves Termination, Agreement, and Majority property. Then, we identify scenarios that cannot be distinguished by fault-free nodes. For example, $f = 1, n = 4$ and $\mathbb{C} = \{0, 1\}$ and the inputs are $0, 0, 1, 1$. Then, it is impossible to distinguish whether 0 or 1 is proposed by the majority of fault-free nodes, since any node may be faulty. As a result, each fault-free node must output the same value in all these scenarios (when different node is faulty). However, Majority property would be violated in one of the scenarios, and we derive a contradiction. Thus, algorithm $\mathbb{A}$ cannot exist. This also implies that Plurality property cannot be achieved.

B. Relaxed Voting Properties

Recall that we propose two relaxed voting properties in Section I-B. Here, we present the relevant results in both synchronous and asynchronous systems.

a) Synchronous Systems: We propose a Byzantine consensus algorithm: Alg-OV (abbreviating Algorithm-Optimal Voting) and prove the theorem below in Section III. Recall that f is the bound on the number of faults. Theorem 2 shows that f is also the *error bound* of Alg-OV.

Theorem 2. *Voting algorithm Alg-OV achieves Termination, Agreement, f-Majority and f-Plurality properties in synchronous systems given that $n \geq 3f + 1$.*

Algorithm Alg-OV is *optimal* with respect to l-Majority and l-Plurality property. That is, no other algorithm can achieve a better error bound.

Theorem 3. *It is impossible to achieve Termination, Agreement, l-Majority (l-Plurality) property with $l < f$ in synchronous systems.*

We prove that Alg-OV ensures c-Verified property with a suitable choice of c.

Theorem 4. *Voting algorithm Alg-OV achieves Termination, Agreement, and c'-Verified property for $c' > \frac{n}{2}$ in synchronous systems given that $n \geq 3f + 1$.*

Note that $(\frac{n}{2} + 1)$-Verified property is different from Majority property, since the majority of fault-free nodes proposing the same value may be less than $n/2 + 1$ when $|\mathbb{C}| = k > 2$. This is why Majority property cannot be achieved in general, whereas, $(\frac{n}{2} + 1)$-Verified property can be achieved using Alg-OV. The theorem below states that Alg-OV is optimal with respect to c-Verified property as well.

Theorem 5. *It is impossible to achieve Termination, Agreement, c-Verified property with $c \leq \frac{n}{2}$ in synchronous systems.*

In Section V-A, we present an *one-step voting* algorithm, in which fault-free nodes generate the voting output in one communication step in contention-free scenarios. Yet, it requires f more nodes, i.e., the algorithm is correct given $n \geq 4f + 1$.

b) Asynchronous Systems: The Fischer-Lynch-Paterson (FLP) result [18] implies that for asynchronous systems, no deterministic algorithm can solve the Byzantine voting problem. Thus, we propose a randomized algorithm Alg-AOV (abbreviating Algorithm-Asynchronous Optimal Voting) that achieves relaxed voting properties in Section IV.

Theorem 6. *Voting algorithm Alg-AOV achieves Termination, Agreement, $(\frac{n+f}{2}+1)$-Majority and $(\frac{n+f}{2}+1)$-Plurality properties in asynchronous systems given that $n \geq 3f+1$.*

Theorem 7. *Voting algorithm Alg-AOV achieves Termination, Agreement, and c'-Verified property for $c' > \frac{n+f}{2}$ in asynchronous systems given that $n \geq 3f+1$.*

The asynchronous algorithm has a larger error bound compared with the synchronous algorithm. We prove that Alg-AOV is asymptotically optimal with respect to l-Majority and l-Plurality property, and optimal with respect to c-Verified property in [40]. In Section V-A, we present an asynchronous *one-step voting* algorithm, in which fault-free nodes generate voting output in one communication step in contention-free scenarios. Yet, the algorithm requires $n \geq 7f+1$. Note that it is proved in [39] that $n \geq 7f+1$ is necessary to solve one-step Byzantine consensus. Thus, our asynchronous voting algorithm has optimal fault-tolerance.

III. Byzantine Voting in Synchronous Systems

We first introduce the formal definition of l-Majority and l-Plurality voting properties, and then present a voting algorithm in synchronous systems – Alg-OV (Algorithm-Optimal Voting) – which achieves both relaxed properties, and is optimal with respect to these two properties.

A. Formal Definitions of l-Majority and l-Plurality Properties

Given an execution, let $\#(v)$ be the number of *fault-free nodes* that propose value v in the execution. For brevity, we do not include the execution in the notation.

Definition 1. (Error Bound) *Given an execution of a voting algorithm $\mathbb{A}$ such that there are m input values proposed by fault-free nodes ($m \geq 1$). Without loss of generality, assume that these values form a set $\{1, \cdots, m\}$ and $\#(1) \geq \cdots \geq \#(m)$. Then the error bound of algorithm $\mathbb{A}$ in this particular execution is*

$$\#(1) - \#(o),$$

where o is the output of algorithm A, and the value o may or may not be in $\{1, \cdots, m\}$. It is possible for fault-free nodes to output $\perp$ or value that is not an input at fault-free nodes. In these cases, $\#(\cdot)$ returns 0.

Definition 2. (l-Majority and l-Plurality property) *In any execution of the algorithm such that (i) v is the value proposed by the majority (or a plurality) of the fault-free nodes; and (ii) o is the output at fault-free nodes, then*

$$\#(v) - \#(o) \leq l \qquad (1)$$

B. Algorithm-Optimal Voting (Alg-OV)

We present the Byzantine voting algorithm Alg-OV, which achieves voting given that $n \geq 3f+1$. The algorithm uses a binary Byzantine broadcast algorithm as a sub-routine, namely *Alg BBB* (Algorithm Binary Byzantine Broadcast). The broadcast algorithm has a single node (potentially faulty) to send its input to the rest of nodes, and it ensures the following properties: (i) *Termination*: All fault-free nodes must terminate with a single output value in a finite amount of time; (ii) *Agreement*: The output values at all fault-free nodes must be identical; and (iii) *Validity*: If the source is fault-free, then output value at each fault-free node equals the input value at the source. Our algorithm is modular, and can use any binary broadcast algorithms that are correct when $n \geq 3f+1$, e.g., [7], [29], [37], [24].

Below, we present the pseudo-code of Alg-OV. The code should be executed by each node i. Denote by v_i the input proposed by node i. In Alg-OV, node i keeps track of an n-entry vector V, where $V[j]$ denotes the j-th entry and will store the output of *Alg BBB* with node j being the source of the binary broadcast.

Algorithm 1 Alg-OV to be executed by node i

```
1: Use Alg BBB to broadcast input $v_i$
2: Participate in all $n$ invocations of Alg BBB
3: for all node $j$ in the system do
4:     $V[j] \leftarrow$ output of Alg BBB with $j$ being the source
5: end for
6: $x \leftarrow$ the value appears most often in $V$
   $\triangleright$ breaking tie arbitrarily
7: Output $x$
```

By "breaking tie arbitrarily", we mean that all fault-free nodes use the same predetermined rules to break tie deterministically. For readers familiar with the literature, the line 1 through line 5 of Alg-OV are used to achieve the interactive consistency property [37]. We use *Alg BBB* instead of algorithms in [37] as a sub-routine, because the algorithms presented later follow a similar structure to Alg-OV.

C. Correctness of Alg-OV

Termination and agreement follow directly from the correctness of *Alg BBB* (proofs in tech-report [40]). Below, we prove the key lemmas that show Alg-OV achieves two relaxed voting properties with suitable choice of parameters. These two lemmas prove Theorems 2 and 4 in Section II, respectively.

Lemma 1. *Alg-OV achieves f-Plurality and f-Majority properties given that $n \geq 3f+1$.*

Proof. Consider any execution of Alg-OV. Suppose that (i) the input value at the plurality of fault-free nodes is z; and (ii) the output value of Alg-OV is x. If $z = x$, then the proof is complete, since $\#(z) - \#(x) = 0 \leq f$. Recall that $\#(v)$ denotes the number of fault-free nodes proposing value v in the given execution.

Now, suppose $z \neq x$. Consider a fault-free node i. At line 6, node i chooses a value that appears most often in V. Denote by $\#_V(v)$ the number of times that value v appears in the vector V. For brevity, we do not include the node index i in the notation. Since by assumption Alg-OV outputs x, it means that at node i,

$$\#_V(x) \ \geq \ \#_V(z) \tag{2}$$

Therefore, there are at least $\#_V(x) - f$ of fault-free nodes that have value x as the input, since (i) by the validity condition of *Alg BBB*, if a source is fault-free, then node i receives the input correctly; and (ii) up to f nodes may be faulty and propose an incorrect input. In other words,

$$\#(x) \ \geq \ \#_V(x) - f \tag{3}$$

Thus, we have

$$\begin{aligned} \#(x) \ &\geq \ \#_V(x) - f && \text{by (3)} \\ &\geq \ \#_V(z) - f && \text{by (2)} \\ &\geq \ \#(z) - f && \text{by validity of } \textit{Alg BBB} \end{aligned} \tag{4}$$

From (4), we have

$$\#(z) - \#(x) \ \leq \ f \tag{5}$$

This proves that Alg-OV satisfies f-Plurality property as per Definition 2. This implies that Alg-OV also satisfies f-Majority property. □

Lemma 2. *Alg-OV achieves c'-Verified property given that $c' > n/2$ and $n \geq 3f + 1$.*

Proof. Assume value v is proposed by $\geq c'$ fault-free nodes. Fix a fault-free node i. Since $c' > n/2$, v must be the value that appears most in vector V at line 6 of Alg-OV. Therefore, each fault-free node i correctly picks value v as the output. □

D. Optimality of Alg-OV

Definition 3. (Optimality) *A voting algorithm $\mathbb{A}$ is optimal with respect to l-Majority (or l-Plurality) property if (i) algorithm $\mathbb{A}$ achieves l'-Majority (or l'-Plurality) property; and (ii) for any other correct voting algorithm $\mathbb{B}$, the error bound of $\mathbb{B}$ is $\geq l'$ in at least one execution.*

In other words, the definition states that an optimal algorithm should have an error bound that is *no larger than* the error bound of any other algorithm $\mathbb{B}$. Note that the condition does not require that in *any* execution, algorithm $\mathbb{A}$ produces the result with smallest error. Instead, it only requires that there exists an execution such that $\mathbb{B}$'s error bound in the execution is $\geq l'$, the error bound of algorithm $\mathbb{A}$. Theorem 3 states that Alg-OV is in fact optimal as per Definition 3. The proof is similar to the proof of Theorem 1 and is presented in our tech-report [40]. We also prove that Alg-OV is optimal with respect to c-Verified property (as stated in Theorem 5), and the proof is presented in [40].

IV. Byzantine Voting in Asynchronous Systems

Here, we consider asynchronous systems, where messages may be delayed arbitrarily by a strong network adversary as discussed in Section I-C. We present a voting algorithm – Alg-AOV (Algorithm-Asynchronous Optimal Voting) – which achieves the two proposed relaxed voting properties. Our algorithm relies on asynchronous consensus algorithms; thus, we begin with a brief discussion on the topic.

A. Consensus in Asynchronous Systems

The famous Fischer-Lynch-Paterson (FLP) result states that it is impossible to solve consensus in asynchronous systems with a single node failure [18]. The impossibility result applies to the Byzantine voting problem with any voting properties discussed in this paper. There are mainly three mechanisms to circumvent the FLP result in the literature: (i) algorithm ensures termination only in partially synchronous periods [23]; (ii) termination property is relaxed to a probabilistic termination property – all fault-free nodes terminate with a single output with probability 1 [11]; or (iii) agreement property is relaxed to an approximate agreement [15] or k-set consensus [14]. Since we do not assume partially synchronous periods, nor approximate/k-set consensus, we focus on Byzantine voting algorithms that ensure probabilistic termination.

B. Algorithm-Asynchronous Optimal Voting (Alg-AOV)

Observe that we cannot directly apply the same technique from Alg-OV, since binary broadcast algorithms may never return and nodes are stuck at line 2 of Alg-OV. Instead, we use a communication primitive called *stable vector* [5], [31] to "try to learn" the input value proposed by each node. Because of asynchrony and faults, one cannot ensures that a node can learn all the input values. The *stable vector* primitive requires at least $3f + 1$ nodes, with at most f being Byzantine faulty. Alg-AOV proceeds in asynchronous rounds, i.e., each node may potentially perform the same round at very different real times. In Round 0, each node i first broadcasts a message consisting of the tuple $(v_i, i, 0)$, where v_i is the input at node i. In this tuple, 0 indicates the (asynchronous) round index. Node i then waits for the *stable vector* primitive to return a set R_i containing round 0 messages. We will rely on the following properties of the *stable vector* primitive to ensure the correctness and optimality of Alg-AOV.

- *Liveness*: At each fault-free node i, *stable vector* returns a set R_i containing at least $n - f$ distinct tuples of the form $(v, k, 0)$.
- *Integrity*: If a fault-free node i never broadcasts $(v', i, 0)$, no fault-free node ever receives $(v', i, 0)$.
- *Uniqueness*: If two fault-free nodes i and j receives $(v, p, 0)$ and $(v', p, 0)$, respectively, from node p, then $v = v'$. This holds even if node p is Byzantine faulty.
- *Containment*: For fault-free nodes i, j, let R_i, R_j be the set of messages returned to nodes i, j by *stable vector* in round 0, respectively. Then, either $R_i \subseteq R_j$ or $R_j \subseteq R_i$. (Also, by the previous property, $|R_i| \geq n - f$ and $|R_j| \geq n - f$.)

A description of the implementation of the *stable vector* primitive is omitted for lack of space. Please refer to [5], [31] for details. In addition to the *stable vector* primitive, we use a multi-valued consensus algorithm as a sub-routine, namely *Alg AMVC* (Algorithm Asynchronous Multi-Valued Consensus). Particularly, *Alg AMVC* achieves weak validity and probabilistic termination property as discussed in Section IV-A. Our algorithms can use any asynchronous multi-valued consensus algorithms with weak validity, e.g., [35].

Below, we present the pseudo-code of Alg-AOV, which should be executed by each node i. Recall that v_i is the input at node i. In the algorithm, all sets are initialized to be empty sets. Among two sets used, X_i is a multiset.

Algorithm 2 Alg-AOV to be executed by node i

```
1: Send message $(v_i, i, 0)$ to all the nodes
2: Wait until *stable vector* returns a set $R_i$    ▷ $|R_i| \geq n - f$
3: Multiset $X_i \leftarrow \{\, v \mid (v, k, 0) \in R_i \}$    ▷ $|X_i| = |R_i|$
4: $v'_i \leftarrow$ the value appears most often in $X_i$
     ▷ breaking tie arbitrarily
5: Run *Alg AMVC* using $v'_i$ as the input
6: $o_i \leftarrow$ output of *Alg AMVC*
7: Output $o_i$
```

Alg-AOV achieves c'-Verified property with $c' > (n+f)/2$ (Theorem 7), and the proof is similar to the one for Alg-OV, so we present it in our technical report [40]. Now, we prove the key lemma for Theorem 6.

Lemma 3. *Alg-AOV achieves* $(\frac{n+f}{2}+1)$*-Majority and* $(\frac{n+f}{2}+1)$*-Plurality properties.*

Proof. The proof is by contradiction. Suppose that Alg-AOV does not satisfy the properties, i.e., there exists an execution E such that (i) there are m fault-free nodes proposing the value v; (ii) there are m' fault-free nodes proposing the output value v_o, where $m' \geq 0$ and $v_o \neq v$; (iii) $m - m' > (n+f)/2 + 1$; and (iv) the output of Alg-AOV is v_o.

Assumptions (ii) and (iii) imply that $m > (n+f)/2 + 1$. However, since in Theorem 7, we have shown that Alg-AOV satisfies $(\frac{n+f}{2}+1)$-Verified property. As a result, in the execution E, Alg-AOV must choose the value v as the output, since it is proposed by m fault-free nodes. This violates assumption (iv), a contradiction. Thus, Alg-AOV achieves $(\frac{n+f}{2}+1)$-Majority and $(\frac{n+f}{2}+1)$-Plurality properties. □

One may wonder why the weak validity of *Alg AMVC* suffices, since if two fault-free nodes propose different values at line 5, *Alg AMVC* may output a null value at line 6. The reason is that it is fine if Alg-AOV also outputs a null value in this case. This is because there must be $\leq (n+f)/2+1$ fault-free nodes proposing the same value. Therefore, even though no fault-free node proposes a null value, Alg-AOV still satisfies $(\frac{n+f}{2}+1)$-Majority/Plurality as per Definition 2.

In [40], we also show that Alg-AOV is *asymptotically optimal* with respect to l-Majority and l-Plurality properties and *optimal* with respect to c-Verified property.

V. One-Step Byzantine Voting Algorithm

While the consensus-based algorithms Alg-OV and Alg-AOV are straightforward, they may not be suitable for some safety-critical systems due to its time complexity. It is well-known that any consensus algorithms require $\geq f+1$ rounds (communication steps) [7], [29], which may be prohibitively high for some systems. As addressed in Section I, voting algorithms are commonly used to mask faults in systems with redundant components [25], [34]. In this scenario, the *common case* is when most components behave correctly and have the same input (i.e., contention-free scenario). Therefore, it is of interest to optimize the performance for the contention-free scenario. In this section, we propose voting algorithms that satisfy the following property:

Definition 4. ***(One-Step Voting)*** *If all fault-free nodes propose the same input value v, then all fault-free nodes output v in one communication step.*

This property is inspired by early-stopping property [7], [29] and one-step property [39] in consensus algorithms. However, the formulation is different; hence, the algorithms are different.

A. Synchronous Systems

We address how to improve the time complexity of voting algorithms in synchronous systems.

a) Algorithm-One-Step Optimal Voting (Alg-OSOV): Clearly, Alg-OV does not satisfy one-step voting property, since there is no known one-step Byzantine broadcast algorithm. We present a new Byzantine voting algorithm Alg-OSOV, which achieves voting and satisfies one-step voting given that $n \geq 4f+1$.

Below, we present the pseudo-code of Alg-OSOV. The code should be executed by each node i. Here, v_i the input at node i. If a node finishes early at line 4, it still needs to participate in Alg-OV to help other nodes to generate the correct output. Even though the running time is one round worse than Alg-OV and more nodes are required, the advantage is to have an output early in the contention-free scenario.

Algorithm 3 Alg-OSOV to be executed by node i

```
1: Broadcast input value $v_i$ to all nodes;    $t \leftarrow v_i$
2: Multiset $R_i \leftarrow$ values received from Step 1
3: **if** some value $v$ appears $\geq n - f$ times in $R_i$ **then**
4:     Output $v$;    $t \leftarrow v$
5: **else if** some value $v$ appears $\geq n - 2f$ times in $R_i$ **and**
   value $v$ appears most often in $R_i$ (break tie arbitrarily)
   **then**
6:     $t \leftarrow v$
7: **end if**
8: Run *Alg-OV* using $t$ as the input
9: $x \leftarrow$ output of *Alg-OV*
10: Output $x$
```

Note that at line 1, "Broadcast" is different from the *Alg BBB* used in Alg-OV. Here, Broadcast means sending a value to every other node, which only requires one round of communication; hence, faulty node may choose to send different values to different nodes.

b) Correctness of Alg-OSOV: Termination follows from the correctness of *Alg-OV*. One-step property is obvious due to line 4.

Lemma 4. *Alg-OSOV achieves agreement property given that* $n \geq 4f + 1$.

Proof. If no fault-free node has an early output at line 4, then *Alg-OSOV* satisfies agreement due to the correctness of *Alg-OV.* Now, consider the case when some fault-free node j outputs value v at line 4.

Claim 1. *If a fault-free node* i *also outputs value* v' *at line 4, then* $v' = v$.

Proof. Suppose not. Then, there are $\geq n - f$ v''s in R_i and $\geq n - f$ v's in R_j. Since there are only n nodes, there are $2n - 2f - n = n - 2f$ values that are broadcast from the same nodes (namely, common nodes). Among these $n - 2f$ common nodes, at most f of them are faulty; hence, there are at least $n - 3f \geq f + 1$ fault-free common nodes. Since fault-free nodes broadcast the same value to all nodes (at line 1), we have derived a contradiction. □

Moreover, all other fault-free nodes i that do not output at line 4 must observe that value v appears $\geq n-2f$ in multiset R_i at line 5, because only faulty nodes may send different values to different nodes. Also, we can show the following claim for v.

Claim 2. *Value* v *is the value that appears most often in* R_i.

Proof. In the multiset R_i, there are at most $f + f$ values that are different from v (up to f values from faulty nodes, and up to f values from fault-free nodes). Since $n \geq 4f+1$, we have $n - 2f > 2f$. Therefore, v appears most often in R_i. □

By Claim 2, all other fault-free nodes will use value v as the input of Alg-OV (line 6 and line 8). Therefore, by the property of Alg-OV, every fault-free node outputs value v, i.e., agreement property is satisfied. □

Lemma 5. *Alg-OSOV achieves* c'*-Verified property given that* $c' > n/2$ *and* $n \geq 4f + 1$.

Proof. Suppose that a value v is proposed by $\geq c'$ fault-free nodes. If some fault-free node outputs some value v' at line 4, then v' must equal value v, since value v is proposed by $\geq c'$ fault-free nodes, and $c' > n/2 > f$, i.e., there are $> f$ v's in R_i for any fault-free node i. By Lemma 4, every fault-free node outputs value v correctly.

If no fault-free node outputs at line 4, then we have:

Claim 3. *No fault-free node can set variable* t *to some value* v' *such that* $v' \neq v$ *at line 6.*

Proof. Observe that there are $\geq c'$ v's appear in R_i for each fault-free node i. Then, we have $c' > \frac{n}{2}$. Value v must appear most often in R_i. Therefore, node i will not update t to value other than v at line 6 due to condition at line 5. □

Claim 3 implies that all fault-free nodes with input v will set variable t to v by the end of line 7. Hence, there are at least $\geq c'$ fault-free nodes have set variable t to value v. This together with the correctness of Alg-OV proves Lemma 5. □

Lemma 6. *Alg-OSOV achieves* f*-Plurality and* f*-Majority property given that* $n \geq 4f + 1$.

Proof. We first prove that Alg-OSOV achieves f-Plurality. Denote by z the input value at the plurality of fault-free node. Now, consider three cases:

- Some fault-free node i outputs some value v' at line 4:
  In this case, $v' = z$. Due to the condition at line 3, there must have at least $n-2f$ fault-free nodes proposing value v', which means that there are at most $2f$ fault-free nodes proposing values other than v'. Since $n-2f \geq 2f+1 > 2f$, v' is proposed by plurality of fault-free nodes. By Lemma 4, every fault-free nodes output $v' = z$. Thus, f-Plurality is satisfied.
- No fault-free node outputs at line 4, and no fault-free node updates variable t at line 6:
  In this case, f-Plurality is satisfied due to the correctness of Alg-OV and the fact that each fault-free node uses the same input for both Alg-OSOV and Alg-OV.
- No fault-free node outputs at line 4, and some fault-free node updates variable t at line 6:
  By way of contradiction, suppose that Alg-OSOV outputs some value y such that
  $$\#(z) - \#(y) > f \tag{6}$$
  Recall that $\#(v)$ denotes the number of fault-free nodes proposing value v. Since Alg-OV satisfies f-Plurality, some fault-free nodes must update variable t to value y at line 6. Due to the condition at line 5, it means that there are $\geq n - 3f$ fault-free nodes proposing value y, i.e., $\#(y) \geq n - 3f$. This together with (6) imply that
  $$\#(z) > f + \#(y) \geq n - 2f \tag{7}$$
  The fact that $n-2f > 2f$ and (7) imply that every fault-free node i sees a majority of value z in R_i. Consequently, no fault-free node would update variable t to value other than z, a contradiction. Therefore, under this case, Alg-OSOV can only output some value y such that $\#(z) - \#(y) \geq f$. Hence, f-Plurality is satisfied.

In all three cases, we show that Alg-OSOV satisfies f-Plurality, which implies Alg-OSOV satisfies f-Majority. □

B. Asynchronous Systems

a) Algorithm-Asynchronous One-Step Optimal Voting (Alg-AOSOV): Below, we present the pseudo-code of Alg-AOSOV. The code should be executed by each node i. Denote by v_i the input proposed by node i. If a node finishes early at line 4, it still needs to participate in Alg-AOV to help other nodes to generate the correct output. The main differences from Alg-OSOV are line 2 to line 7. To accommodate asynchrony, each node can wait at most $n-f$ values. The exact parameter in each if-condition has to be adjusted accordingly.

Algorithm 4 Alg-AOSOV to be executed by node i

```
1: Broadcast input value $v_i$ to all nodes;    $t \leftarrow v_i$
2: Wait until $n-f$ values from Step 1 have been received.
   Denote by $R_i$ the multiset of received values.
3: if some value $v$ appears $\geq n-2f$ times in $R_i$ then
4:     Output $v$;    $t \leftarrow v$
5: else if some value $v$ appears $\geq n-4f$ times in $R_i$ and
   value $v$ appears most often in $R_i$ (break tie arbitrarily)
   then
6:     $t \leftarrow v$
7: end if
8: Run Alg-AOV using $t$ as the input
9: $x \leftarrow$ output of Alg-AOV
10: Output $x$
```

b) Correctness of Alg-AOSOV: Termination follows from the correctness of *Alg-AOV*. One-step property is obvious due to line 4. The correctness proof requires $n \geq 7f+1$, which is optimal due to the impossibility result proved in [39].

Lemma 7. *Alg-AOSOV achieves agreement property given that $n \geq 7f+1$.*

Proof. If no fault-free node has an early output at line 4, then *Alg-AOSOV* satisfies agreement due to the correctness of *Alg-AOV.* Now, consider the case when some fault-free node j outputs value v at line 4. First, similar to Claim 1, we can show that any other fault-free node that outputs at line 4 must output v. Now, consider fault-free nodes that do not output at line 4. In this case, all other fault-free nodes must observe that value v appears $\geq n-4f$ in multiset R_i at line 5 for a fault-free node i, since (i) by assumption, $\geq n-3f$ fault-free nodes propose v; and (ii) node i waits until $|R_i| = n-f$.

Then, we can show

Claim 4. *Value v is the value that appears most often in R_i*

Proof. Since $n \geq 7f+1$, there are $\geq 3f+1$ v's in R_i, and there are $\leq (n-f)-(n-4f) = 3f$ values that are different from v. Therefore, v appears most often in R_i. ☐

By Claim 4, all other fault-free nodes will use value v as the input of Alg-AOV (line 6 and line 8). Therefore, by the property of Alg-AOV, every fault-free node outputs value v, i.e., agreement property is satisfied. ☐

Lemma 8. *Alg-AOSOV achieves c'-Verified property given that $c' > \frac{n+f}{2}$ and $n \geq 7f+1$.*

Proof. Suppose that a value v is proposed by $\geq c'$ fault-free nodes. If some fault-free node outputs some value v' at line 4, then v' must equal value v, since (i) value v is proposed by $\geq c'$ fault-free nodes, and a fault-free node i would receive $\geq c'-f$ in R_i at line 2; and (ii) $c'-f > \frac{n-f}{2} > 3f$, i.e., there are $> 3f$ v's and $< n-4f \leq 3f+1$ values other than v in R_i. By Lemma 7, every fault-free node outputs value v correctly.

If no fault-free node outputs at line 4, then we have:

Claim 5. *No fault-free node can set variable t to some value v' such that $v' \neq v$ at line 6.*

Proof. Observe that there are $\geq c'-f$ v's appear in R_i for each fault-free node i. Then, we have $c'-f > \frac{n-f}{2}$. Value v appears most often in R_i. Therefore, node i will not update t to value other than v at line 6 due to condition at line 5. ☐

Claim 5 implies that by the end of line 7, there are at least $\geq c'$ fault-free nodes have set variable t to value v. Therefore, c-Verified property is satisfied due to Alg-AOV's property ☐

Lemma 9. *Alg-AOSOV achieves $(\frac{n+f}{2}+1)$-Plurality and $(\frac{n+f}{2}+1)$-Majority property given that $n \geq 7f+1$*

Proof. We first prove that Alg-AOSOV achieves $(\frac{n+f}{2}+1)$-Plurality. Denote by z the input value at the plurality of fault-free node. Now, consider three cases:

- Some fault-free node i outputs some value v' at line 4: Suppose in the execution, exactly $f^* \leq f$ nodes become faulty. Condition at line 3 together with the assumption that node i outputs value v' at line 4, there must have at least $n-2f-f^*$ fault-free nodes proposing value v'. This implies that there are at most $(n-f^*)-(n-2f-f^*) = 2f$ fault-free nodes proposing values other than v'. Then,
$$n-2f-f^* \geq n-3f \geq 2f+1 > 2f$$
Therefore, value v' is proposed by plurality of fault-free nodes. By Lemma 7, every fault-free node outputs value $v' = z$ correctly. Thus, $(\frac{n+f}{2}+1)$-Plurality is satisfied.
- No fault-free node outputs at line 4, and no fault-free node updates variable t at line 6:
In this case, $(\frac{n+f}{2}+1)$-Plurality is satisfied due to the correctness of Alg-AOV and the fact that each fault-free node uses the same input for Alg-AOSOV and Alg-AOV.
- No fault-free node outputs at line 4, and some fault-free node updates variable t at line 6:
By way of contradiction, suppose that Alg-AOSOV outputs some value y such that
$$\#(z) - \#(y) > \frac{n+f}{2} + 1 \tag{8}$$
Recall that $\#(v)$ denotes the number of fault-free nodes proposing value v. Since Alg-AOV satisfies $(\frac{n+f}{2}+1)$-Plurality, some fault-free nodes must update variable t to value y at line 6. Due to the condition at line 5, it means that there are $\geq n-5f$ fault-free nodes proposing value y, i.e., $\#(y) \geq n-5f$. This together with (8) imply that
$$\#(z) > \frac{n+f}{2} + 1 + (n-5f) \tag{9}$$
(9) together with the fact that each fault-free node i only waits for $n-f$ values from line 1 imply that node i sees

$> (n+f)/2+1+(n-5f)-f \geq n-2f+1$. Observe that $n-2f+1 > n-4f$. Consequently, no fault-free node would update variable t to value other than z due to condition at line 5, a contradiction. Therefore, Alg-AOSOV can only output some value y such that $\#(z) - \#(y) \geq \frac{n+f}{2}+1$. Hence, $(\frac{n+f}{2}+1)$-Plurality is satisfied.

In each of the three case above, we show that Alg-OSOV satisfies $(\frac{n+f}{2}+1)$-Plurality. This implies that Alg-OSOV also satisfies $(\frac{n+f}{2}+1)$-Majority. □

VI. Related Work

Fault-tolerant *consensus* has received significant attention over the past three decades [7], [29] since the seminal work by Lamport, Shostak, and Pease [37], [24]. The problem has been studied extensively, including various model assumptions, communication mechanisms, correctness conditions, time/communication complexity, and early-stopping property. Among these consensus problems, the Multi-Valued Consensus (MVC) problem is closely related to the voting problem, which considers the case when input may take more than two values [42], [28]. Most works on MVC considered variants of weak validity conditions (e.g., [36], [19], [27], [32], [41], [35]), which are similar to Weak property discussed in Section I-B.[2] Gil Neiger proved that $n \geq \max\{k,3\}f+1$ is necessary to solve multi-valued consensus with strong validity [33]. These results solve Byzantine voting with either Strong or Weak property; however, they do not apply to Byzantine voting with other voting properties. Santoro and Widmayer considered "consensus with strong majority" [38], which is different from "voting with Majority property" in this paper. In [38], they considered link failures, and only $\lceil n/2 \rceil + 1$ nodes need to reach agreement; whereas we consider node failures and all fault-free nodes need to agree on the same output.

Significant effort has been devoted to fault-tolerant leader election problem, e.g., [20], [30], including electing a leader in the presence of Byzantine nodes [22], [8]. The correctness conditions of these works are different from ours. Abraham et al. studied the problem from a game theoretical perspective in both synchronous and asynchronous systems [1]. Only rational nodes were considered in [1]. In contrast, we consider Byzantine nodes, which are not rational in the game theoretical point of view. BAR model [26], [13], [2] was proposed for different problems, where nodes are categorized as Byzantine, Altruistic, and Rational. They did not consider voting properties studied in our work. Researchers in economic literature presented results on fault-free voting (or election), e.g., [3], [17], [16], [4], [21]. These works did not consider node faults.

Recently, Plurality consensus has been studied in the GOSSIP model [9], [10]. They had different problem formulation and they did not consider faults [9], [10]. Our work is inspired by [12], which presented early results on Byzantine voting problem in synchronous systems. We extend their work to both synchronous and asynchronous systems. We present more general impossibility results, and voting algorithms that achieved two new relaxed voting properties.

Voting is also widely used in practical safety-critical systems [25], [34]. Parhami presented early results in optimal voting algorithms [34]. Reference [25] presented a thorough survey on the topic. These works focused on how to use centralized voting mechanism to mask component failures; however, they did not consider the case when the nodes participating in the voting may become faulty. As discussed in Section I-B, the two proposed relaxed voting properties share some similarities with the works in [25], [34]. Thus, we believe our voting algorithms are useful in safety-critical systems. We briefly discuss the applications in Section VII-B.

VII. Discussion and Summary

A. Comparison of Voting Properties

In some cases, the l-Majority/Plurality is identical to Weak property, i.e., given inputs proposed by fault-free nodes, they impose the same constraints on the output. Consider the case when $f=1$ and $n=4$. In this scenario, 2-Majority/Plurality is identical to Weak property as discussed in [40]. Our Byzantine voting algorithm Alg-OV achieves f-Majority/Plurality. In the example outline in Section I-B, 1-Majority/Plurality is different from Weak property. Consider the case when all nodes are fault-free with inputs $0,0,0$ and 1. In this case, Alg-OV outputs 0; whereas Weak property allows an algorithm to output 1, since no node is faulty.

In both synchronous and asynchronous systems, we have developed algorithms that achieve both relaxed voting properties. Hence, it is natural to ask the question: *why do we need both l-Majority/Plurality and c-Verified property?* There are mainly two reasons. First, there is a subtle difference between the two properties in some edge cases. Again consider the case when $f=1$ and $n=4$. Recall that Alg-OV achieves 1-Majority/Plurality and 3-Verified property in this case. Suppose three nodes are fault-free and have inputs $0,0,1$, and faulty node has an arbitrary input. 1-Majority/Plurality requires that output must be an input at a fault-free node; whereas 3-Verified property imposes no constraints on output. Second, there exist some inherent trade-offs between message complexity and the parameter choices. For example, consider a system of $mf+1$ nodes, where $m >> 3$. Then, we can reduce the message complexity by increasing the error bound for l-Majority/Plurality. For example, to achieve $(f+x)$-Majority/Plurality, we only need $n-x$ nodes to execute Alg-OV and broadcast their output as long as $n-x \geq 3f+1$. This follows from Theorem 2 and the observation that we ignore inputs from x nodes. In contrast, c-Verified property does not have such straightforward trade-off. Study of such trade-offs is left as future work.

B. Application in Safety-Critical Systems

For safety-critical systems where a centralized voting mechanism [25], [34] is not feasible, and nodes may become Byzantine faulty, algorithms presented in this work can be

[2]These "weak" validity conditions have several different names in the literature, such as unanimity, obligation, or non-intrusion (e.g., [39], [32]).

used to mask faults and propagate the voting output to fault-free nodes. Moreover, we have to ensure that there are enough number of nodes to achieve the desired results. For example, Theorem 4 states that $> \frac{n}{2}$ fault-free nodes need to have the same input v for any correct voting algorithm to output value v in synchronous systems. Therefore, to ensure that a safety-critical system works correctly in the scenario when up to f nodes may become faulty, and up to f' fault-free nodes may observe and propose bad input (e.g., due to bad sensors), we must have $n - f - f' > \frac{n}{2}$. That is, we must have $n \geq \max\{3f+1, 2f+2f'\}$ to obtain the desired results.

C. Summary and Future Work

We proved general impossibility results for the Byzantine voting problem, and proposed voting algorithms that are (asymptotically) optimal with respect to two relaxed voting properties in synchronous and asynchronous systems. There are two main open questions: (i) trade-off between message complexity and error bound of the voting algorithms, and (ii) applications in safety-critical systems, mainly on the quantitative assessment of the proposed algorithms.

References

[1] I. Abraham, D. Dolev, and J. Y. Halpern. "Distributed Protocols for Leader Election: A Game-Theoretic Perspective," *Distributed Computing: 27th International Symposium, DISC 2013*

[2] A. Aiyer, L. Alvisi, A. Clement, M. Dahlin, J. Martin, and C. Porth. "BAR tolerance for cooperative services," In *Symposium on Operating Systems Principles (SOSP)*, Oct. 2005.

[3] K. Arrow. "Social Choice and Individual Values," Cowles Foundation Monographs Series. Yale University Press, 1963.

[4] K. J. Arrow. "A Difficulty in the Concept of Social Welfare," *The Journal of Political Economy*, 58(4):328–346, 1950.

[5] H. Attiya, A. Bar-Noy, D. Dolev, D. Peleg, and R. Reischuk. "Renaming in an asynchronous environment," *Journal of the ACM*, July 1990.

[6] H. Attiya and F. Ellen. "Impossibility Results for Distributed Computing," Morgan & Claypool, June 2014.

[7] H. Attiya and J. Welch. "Distributed Computing: Fundamentals, Simulations, and Advanced Topics," Wiley Series on Parallel and Distributed Computing, 2004.

[8] J. Augustine, G. Pandurangan, and P. Robinson. "Fast Byzantine Leader Election in Dynamic Networks," *Distributed Computing: 29th International Symposium, DISC 2015*

[9] L. Becchetti, A. Clementi, E. Natale, F. Pasquale, and R. Silvestri. "Plurality consensus in the gossip model," In *Proceedings of the Twenty-Sixth Annual ACM-SIAM Symposium on Discrete Algorithms*, SODA '15, pages 371–390. SIAM, 2015.

[10] L. Becchetti, A. Clementi, E. Natale, F. Pasquale, R. Silvestri, and L. Trevisan. "Simple dynamics for plurality consensus," In *Proceedings of the 26th ACM Symposium on Parallelism in Algorithms and Architectures*, SPAA '14, pages 247–256, New York, NY, USA, 2014. ACM.

[11] M. Ben-Or. "Another advantage of free choice (extended abstract): Completely asynchronous agreement protocols," In *Proceedings of the Symposium on Principles of Distributed Computing*, PODC 1983.

[12] H. Chauhan and V. K. Garg. "Democratic Elections in Faulty Distributed Systems," *Distributed Computing and Networking: 14th International Conference, ICDCN 2013, Mumbai, India, January 3-6, 2013.*

[13] A. Clement, H. Li, J. Napper, J.-P. Martin, L. Alvisi, and M. Dahlin. "BAR primer," In *38th Annual IEEE/IFIP International Conference on Dependable Systms and Networks (DSN)*, June 2008.

[14] R. de Prisco, D. Malkhi, and M. Reiter. "On k-set consensus problems in asynchronous systems," *IEEE Trans. Parallel Distrib. Syst.*, 12(1):7–21, Jan. 2001.

[15] D. Dolev, N. A. Lynch, S. S. Pinter, E. W. Stark, and W. E. Weihl. "Reaching approximate agreement in the presence of faults,". *J. ACM*, 33:499–516, May 1986.

[16] D. Easley and J. Kleinberg. "Networks, Crowds, and Markets: Reasoning about a Highly Connected World," Cambridge, 2010.

[17] R. Farquharson. "Theory of voting,". Blackwell, 1969.

[18] M. J. Fischer, N. A. Lynch, and M. S. Paterson. "Impossibility of distributed consensus with one faulty process," *J. ACM*, 32:374–382, April 1985.

[19] M. Fitzi and M. Hirt. "Optimally efficient multi-valued Byzantine agreement," In *Proceedings of the Twenty-fifth Annual ACM Symposium on Principles of Distributed Computing.*

[20] R. Ingram, T. Radeva, P. Shields, S. Viqar, J. E. Walter, and J. L. Welch. "A leader election algorithm for dynamic networks with causal clocks," *Distributed Computing*, 26(2):75–97, 2013.

[21] S. Ishikawa and K. Nakamura. "The strategy-proof social choice functions," *Journal of Mathematical Economics*, 6(3):283 – 295, 1979.

[22] B. M. Kapron, D. Kempe, V. King, J. Saia, and V. Sanwalani. "Fast asynchronous Byzantine agreement and leader election with full information," *ACM Trans. Algorithms*, 6(4), 2010.

[23] L. Lamport. "The part-time parliament," *ACM Trans. Comput. Syst.*, 16(2):133–169, May 1998.

[24] L. Lamport, R. Shostak, and M. Pease. "The Byzantine generals problem," *ACM Trans. Program. Lang. Syst.*, 4(3):382–401, July 1982.

[25] G. Latif-Shabgahi, J. M. Bass, and S. Bennett. "A taxonomy for software voting algorithms used in safety-critical systems," *IEEE Transactions on Reliability*, 53(3):319–328, Sept 2004.

[26] H. Li, A. Clement, E. Wong, J. Napper, I. Roy, L. Alvisi, and M. Dahlin. "BAR gossip," In *Proceedings of the 2006 USENIX Operating Systems Design and Implementation (OSDI)*, Nov. 2006.

[27] G. Liang and N. Vaidya. "Error-free multi-valued consensus with Byzantine failures," In *Proceedings of the 30th Annual ACM SIGACT-SIGOPS Symposium on Principles of Distributed Computing.*

[28] N. A. Lynch, M. Fischer, and R. Fowler. "Simple and efficient Byzantine generals algorithm," In *"Proceedings - Symposium on Reliability in Distributed Software and Database Systems"*, pages 46–52. IEEE, 1982.

[29] N. A. Lynch. "Distributed Algorithms," Morgan Kaufmann, 1996.

[30] D. Malkhi, F. Oprea, and L. Zhou. "Omega meets Paxos: Leader election and stability without eventual timely links," In P. Fraigniaud, editor, *DISC*, volume 3724 of *Lecture Notes in Computer Science*, 2005.

[31] H. Mendes, C. Tasson, and M. Herlihy. "Brief announcement: The topology of asynchronous Byzantine colorless tasks." In *The 27th International Symposium on Distributed Computing (DISC)*, 2013.

[32] A. Mostefaoui and M. Raynal. "Signature-free asynchronous byzantine systems: From multivalued to binary consensus with $t < n/3$, $o(n^2)$ messages, and constant time," In *Structural Information and Communication Complexity*, volume 9439, pages 194–208. 2015.

[33] G. Neiger. "Distributed consensus revisited," *Inf. Process. Lett.*, 49(4):195–201, Feb. 1994.

[34] B. Parhami. "Optimal algorithms for exact, inexact, and approval voting," In *Fault-Tolerant Computing, 1992. FTCS-22. Digest of Papers., Twenty-Second International Symposium on*, pages 404–411, July 1992.

[35] A. Patra. "Error-free Multi-valued Broadcast and Byzantine Agreement with Optimal Communication Complexity," *Principles of Distributed Systems: 15th International Conference, OPODIS 2011*

[36] A. Patra and C. P. Rangan. "Communication Optimal Multi-valued Asynchronous Broadcast Protocol," *Progress in Cryptology – LATINCRYPT 2010: First International Conference on Cryptology and Information Security in Latin America, August 8-11, 2010*

[37] M. Pease, R. Shostak, and L. Lamport. "Reaching agreement in the presence of faults," *J. ACM*, 27(2):228–234, Apr. 1980.

[38] N. Santoro and P. Widmayer. "Agreement in synchronous networks with ubiquitous faults," *Theor. Comput. Sci.*, 384(2-3):232–249, Oct. 2007.

[39] Y. J. Song and R. Renesse. "Bosco: One-Step Byzantine Asynchronous Consensus," *Distributed Computing: 22nd International Symposium, DISC 2008, Arcachon, France, September 22-24, 2008.*

[40] L. Tseng. "Election in the presence of byzantine faults," Technical report, University of Illinois at Urbana-Champaign, 2016.

[41] L. Tseng and N. H. Vaidya. "Fault-tolerant consensus in directed graphs," In *Proceedings of the 2015 ACM Symposium on Principles of Distributed Computing*, PODC 2015.

[42] R. Turpin and B. A. Coan. "Extending binary Byzantine agreement to multivalued Byzantine agreement." In *Information Processing Letters*, volume 18, pages 73–76. 1984.