Fault-tolerant Consensus in Anonymous Dynamic Network

Qinzi Zhang[1] and Lewis Tseng[2]*
[1]Boston University, Boston, USA
[2]Clark University, Worcester, USA
E-mails: [1]qinziz@bu.edu, [2]lewistseng@acm.org

***Abstract*—This paper studies the feasibility of reaching consensus in an anonymous dynamic network. In our model, n *anonymous* nodes proceed in synchronous rounds. We adopt a hybrid fault model in which up to f nodes may suffer crash or Byzantine faults, and the *dynamic* message adversary chooses a communication graph for each round.**

We introduce a *stability* property of the dynamic network – (T, D)-*dynaDegree* for $T \geq 1$ and $n-1 \geq D \geq 1$ – which requires that for every T consecutive rounds, any fault-free node must have incoming directed links from at least D distinct neighbors. These links might occur in different rounds during a T-round interval. $(1, n-1)$-dynaDegree means that the graph is a complete graph in every round. $(1, 1)$-dynaDegree means that each node has at least one incoming neighbor in every round, but the set of incoming neighbor(s) at each node may change arbitrarily between rounds.

We show that exact consensus is impossible even with $(1, n-2)$-dynaDegree. For an arbitrary T, we show that for crash-tolerant approximate consensus, $(T, \lfloor n/2 \rfloor)$-dynaDegree and $n > 2f$ are together necessary and sufficient, whereas for Byzantine approximate consensus, $(T, \lfloor (n+3f)/2 \rfloor)$-dynaDegree and $n > 5f$ are together necessary and sufficient.

***Index Terms*—Consensus, Approximate consensus, Byzantine, Impossibility, Message adversary, Dynamic network**

I. Introduction

A dynamic network is a natural model for mobile devices equipped with wireless communication capability. Node mobility and unpredictable wireless signal (e.g., due to interference and attenuation) make the systems of mobile devices inherently dynamic. For example, nodes may join, leave, and move around, and communication links between nodes may appear and disappear over time.

Recent works have studied consensus and other distributed tasks in various formulations of dynamic networks. Following [10], [17], [22]–[24], we consider a network that does *not* eventually stops changing. That is, there exists a dynamic message adversary that controls and chooses the set of communication links continually. More concretely, we study computability of fault-tolerant consensus in the anonymous dynamic network model [11], [12], [28], [29] when nodes may crash or become Byzantine.

A. Anonymous Dynamic Network Model

We consider a fixed set of n nodes that proceed in *synchronous* rounds and communicate by a *broadcast* primitive. For example, such a broadcast primitive might be implemented by a medium access control (MAC) protocol in a wireless network. In each round, the communication graph is chosen by the dynamic message adversary. We do *not* assume the existence of a neighbor-discovery mechanism or an acknowledgement from a MAC layer; thus, nodes do not know the set of nodes that received their messages.

In every round t, the adversary first chooses the directed links that are "reliable" for round t. In other words, the links that are *not* chosen will drop any messages that were sent via them in round t. The adversary may use nodes' internal states at the beginning of the round and the algorithm specification to make the choice. We consider *deterministic* algorithms in which nodes update internal states and generate message following the specification deterministically.

The nodes do not know which links were chosen by the adversary. Each message is then delivered to the sender's outgoing neighbors, as defined by the links chosen by the adversary. After messages are received from incoming neighbors, the nodes transition to new states, and enter round $t+1$. Nodes are assumed to know n, the size of the network, and f, the upper bound on the number of node faults.

Compared to prior works on dynamic networks, our model are different from two following perspectives:

- *Anonymity*: Nodes are assumed to be anonymous [3], [4], [16], [21], [28], [29], [35]. That is, nodes are assumed to be identical and nodes do *not* have a unique identity. Instead, nodes may use "local" communication ports or rely on the underlying communication layer (e.g., MAC layer) to distinguish messages received from different incoming neighbors.
- *Hybrid faults*: In addition to the message adversary, up to f nodes may crash or have Byzantine behavior.

We are interested in the formulation, because most mobile devices are fragile and it is important to tolerate node faults. Moreover, in practice, it it not always straightforward to bootstrap a large-scale dynamic system so that all the nodes have unique and authenticated identity. For example, even though typical MAC protocols assume unique MAC address, it is a good engineering and security practice *not* to rely on this information in the upper layer.[1] Plus, MAC spoofing

*This material is based upon work partially supported by the National Science Foundation under Grant CNS-2238020.

[1]There is also a privacy aspect. For example, recent versions of Android and iOS support MAC randomization to prevent from device tracking.

is a well-known attack, which will add another layer of complexity in handling identities when designing Byzantine-tolerant algorithms.

This paper aims to answer the following question:

When is consensus solvable in our anonymous dynamic network model?

In this paper, we focus on the case when the nodes are able to communicate directly with each other, i.e., a single-hop network. The case of multi-hop communication is left as an interesting future work. Our model is motivated by the applications in drones, robots, and connected vehicles. In this applications, a team of fixed number of nodes are configured to communicate with each other, via dynamic wireless networks, in order to solve certain tasks (e.g., search, dynamic speed configuration, and flocking). In these applications, consensus is a key enabling primitive.

B. A Stability Property

Towards our goal, we introduce a *stability* property called "(T, D)*-dynaDegree*" for $T \geq 1$ and $n-1 \geq D \geq 1$, which quantifies the number of distinct incoming neighbors for each fault-free node over any T-round interval.[2] T is assumed to be finite, and both T and D are *unknown* to nodes. We only use these parameters for analysis. We do not assume the graph contains self-loop; hence the parameter D is upper bounded by $n-1$.

More concretely, (T, D)-dynaDegree requires that for every T *consecutive* rounds in a given execution, any fault-free node must have incoming links from at least D *distinct* neighbors. These directed links might occur in different rounds during a T-round interval. $(1, n-1)$-dynaDegree means that the graph is a complete graph in every round. $(1, 1)$-dynaDegree means that each node has at least one incoming neighbor in every round, but the incoming neighbor(s) may change arbitrarily between rounds. Figure 1 presents an example execution in a 3-node network that satisfies $(2, 1)$-dynaDegree, but does not satisfy $(1, 1)$-dynaDegree.

C. Contributions

We characterise the feasibility of solving consensus in our model using the stability property. In particular, we have the following contributions:

- We identify how (T, D)-dynaDegree is related to a recently identified impossibility result by Gafni and Losa [18], which then implies that exact consensus is impossible with $(1, n-2)$-dynaDegree even when no node may crash ($f = 0$). In other words, there exists an execution such that the network satisfies the stability property $(1, n-2)$-dynaDegree, yet nodes are not able to agree on exactly the same output.
- We identify that for crash-tolerant approximate consensus, $(T, \lfloor n/2 \rfloor)$-dynaDegree and $n > 2f$ are together necessary and sufficient.
- For Byzantine approximate consensus, $(T, \lfloor (n + 3f)/2 \rfloor)$-dynaDegree and $n > 5f$ are together necessary and sufficient.

Section IV presents our crash-tolerant approximate consensus algorithm. Section V presents our Byzantine approximate consensus algorithm. The impossibility results are presented in Section VI, which imply the necessity of the identified conditions. We conclude and discuss extensions in Section VII.

II. Preliminaries

A. Our Model: Anonymous Dynamic Network

We consider a *synchronous* message-passing system consisting of n *anonymous* nodes. Nodes only know n and do *not* have unique identities. For presentation and analysis purpose, we denote the set of nodes as the set of IDs, i.e., $\{1, \ldots, n\}$. For brevity, we often denote it by $[n]$.

Node Faults. We assume that at most f nodes may become faulty. We consider both crash and Byzantine faults. In the former model, a faulty node may crash and stop execution at any point of time. The latter model assumes faulty nodes may have an arbitrary faulty behavior, including sending different messages to different nodes. The set of faulty nodes is denoted by $\mathcal{B}$. Nodes that are not faulty are called *fault-free*. The set of fault-free nodes is denoted by $\mathcal{H}$.

Communication and Message Adversary. The underlying communication network is modeled as a synchronous dynamic network represented as a dynamic graph $G = (V, E)$, where V is a static set of nodes $[n]$, and $\mathcal{E} : \mathbb{N} \rightarrow \{(u, v) \mid (u, v) \in E\}$ is a function mapping a round number $t \in \mathbb{N}$ to a set of *directed* links $\mathcal{E}(t)$. For $(u, v) \in \mathcal{E}(t)$, v is said to be u's outgoing neighbor and u is said to be v's incoming neighbor in round t.

In round t, only messages sent over $\mathcal{E}(t)$ are delivered. All other messages are lost. We consider a dynamic message adversary that chooses $\mathcal{E}(t)$ in every round t. Note that there are different ways of modeling a dynamic network, e.g., using temporal graphs [10], [17]. We adopt the definition from [22]–[24].

We do not assume the existence of self-loop in E; however, nodes have the ability to send a message to itself. Such a message delivery *cannot* be disrupted by the message adversary. In other words, a message sent to oneself is always delivered reliably.

Most prior works assume that each node sends only one message and each message is of size at most $O(\log n)$ bits [10], [17], [22]–[24]. We adopt the same assumption of limited bandwidth of each edge. Section VII briefly discusses when each link has different bandwidth constraints.

Anonymity and Port Number. Nodes execute the same code, because they are identical and the only difference is a potentially distinct input given to each node. They

[2]"dynaDegree" stands for dynamic degree.

communicate with each other via a broadcast primitive. The delivery of messages are determined by the edge set chosen by the message adversary in each round t, namely $\mathcal{E}(t)$.

Following [3], we assume that each node has a "local" label for each incoming neighbor, i.e., a *unique port number* for each incoming link. The labels are *local* in the sense that two different nodes may use two different ports to identify messages received from the same node; therefore, it is not possible to use such information to assign global unique identities to all nodes in our model, without using consensus.

The nodes, however, can use ports to distinguish the sender for each received message locally. Since the ports are assumed to be static throughout the execution of the algorithm, upon receipt of two incoming messages m_1 and m_2, a node i is able to identify that these two messages are from two different senders by using the port numbers. In addition, a node i has the ability to keep track of all the past messages received from a specific port.

Formally, for node i, a port numbering is a bijection function $P_i : \{v \in V\} \rightarrow \{1, 2, \ldots, n\}$. Recall that we assume each node knows n; hence, P_i is well-defined. For two different nodes i and j, P_i may be different from P_j. Depending on the graph G and the message adversary, nodes might never receive messages from a specific port.

We assume that the underlying communication layer is authenticated in the sense that a Byzantine sender cannot tamper with the port numbering at fault-free nodes. As a result, Byzantine sender is *not* able to send bogus messages on behalf of other nodes.

B. A Stability Property: (T, D)-dynaDegree

We characterize graphs using the following property:

Definition 1 (**(T, D)-dynaDegree**)**.** *A dynamic graph $G = (V, E)$ satisfies (T, D)-dynaDegree for a finite $T \geq 1$ and $n - 1 \geq D \geq 1$ if for all $t \in \mathbb{N}$, in the static graph $G_t := (V, \cup_{i=t}^{t+T-1} \mathcal{E}(t))$, the number of distinct incoming neighbors "aggregated" over any T-round interval at any fault-free node i is at least D.*

On a high-level, G is the "base" communication graph that defines the capability for sending messages when all the links are reliable. In each round t, the message adversary chooses the set of reliable links, defined as $\mathcal{E}(t)$ – links in $\mathcal{E}(t)$ deliver message reliably. Nodes do not know G, nor $\mathcal{E}$.

The property (T, D)-dynaDegree counts the number of distinct incoming neighbors D over any T-round period. G_t is a static graph whose set of links does not change over time. The set of links in G_t is a union of $\mathcal{E}(t), \mathcal{E}(t + 1), \cdots, \mathcal{E}(t + T - 1)$. Any fault-free nodes in G_t must have $\geq D$ distinct incoming neighbors. Nodes do *not* know T nor D. These parameters are only used for analysis.

By definition, the incoming neighbors required by the T-interval dynamic degree does *not* need to be fault-free. For example, the message adversary may choose links to deliver messages from Byzantine neighbors.

Example. Consider Figure 1. $G(V, E)$ has $V = \{1, 2, 3\}$ and $E = \{(1, 2), (1, 3), (2, 1), (2, 3), (3, 1), (3, 2)\}$. The example satisfies $(2, 1)$-dynaDegree because in any 2-round interval, each node has at least 1 incoming neighbor. It does not satisfy $(1, 1)$-dynaDegree, because in odd rounds, the graph is disconnected.

Comparison with Prior Stability Properties. Prior papers identified several different stability properties on dynamic graphs for various distributed tasks. We compare (T, D)-dynaDegree with most relevant ones:

- *T-interval connectivity* [22]: it requires that for every T consecutive rounds, there exists a stable connected spanning subgraph. The set of dynamic links are assumed to be *bi-directional*. In other words, the graph is strongly connected in every T-round internal.
- *Rooted spanning tree* [10], [17], [38]: it requires that in every round t, the graph contains a *directed* rooted spanning tree, i.e., there exists at least one "coordinator" that can reach every other node (potentially via a multi-hop route) in the graph for round t.

Our condition is different from T-interval connectivity because of our assumption of directed links. (T, D)-dynaDegree is different from rooted spanning tree, because we allow rounds in which $G(V, \mathcal{E}(t))$ does not have a "root."

In order to solve consensus, the corresponding (T, D)-dynaDegree requires G_t to have a root node over a T-round interval. Hence, it is tempting to view such an interval in our model as one "mega-round" in the prior model [10], [17], [38]. However, the "dynamics" of how nodes change their states is different in our model. Generally speaking, our model captures a more fine-grained interaction between nodes that have different "views." This is because in a T-round interval, some nodes may update frequently while the other nodes only update once. Such a behavior is not captured when using the "mega-round" formulation. This perspective will become more clear when we discuss our algorithms.

C. Exact Consensus and Approximate Consensus

We study both exact consensus and approximate consensus problems [5], [13], [30], [32]. The former task requires that the fault-free nodes agree on exactly the same output, whereas the latter one only requires that the fault-free nodes to agree on roughly the same output. Formally, we have

Definition 2. Binary exact consensus algorithms need to satisfy the following three conditions:

(i) *Termination*: Each fault-free node outputs a value;
(ii) *Validity*: The output of each fault-free node equals to some binary input given to a non-Byzantine node; and
(iii) *Agreement*: Fault-free nodes have an identical output.

Definition 3. Approximate consensus algorithms need to satisfy the following three conditions:

(i) *Termination*: Each fault-free node outputs a value;

(a) When t is odd, $\mathcal{E}(t)$ is empty (b) When t is even, $\mathcal{E}(t) = \{(1, 2), (2, 1), (2, 3), (3, 2)\}$

Fig. 1: **Illustration of an example message adversary.** Figure 1a shows that during odd rounds, the message adversary removes all the links, whereas Figure 1b shows that during even rounds, the adversary removes two links $(1, 3)$ and $(3, 1)$.

(ii) *Validity*: The outputs of all fault-free nodes are within the convex hull of the non-Byzantine inputs; and
(iii) ϵ-*agreement*: The outputs of all fault-free nodes are within ϵ of each other's output.

Following prior work [17], [30], [36], we assume that the range of initial inputs of all nodes is bounded. Without loss of generality, we can scale the inputs to $[0, 1]$ as long as they are bounded by scaling ϵ down by the same factor. Formally, for approximate consensus algorithms, we assume that every $i \in \mathcal{H}$ has input $x_i \in [0, 1]$.

D. Challenges in Anonymous Dynamic Networks

Prior fault-tolerant consensus algorithms do not work in our model. There are mainly three categories of algorithms in synchronous or asynchronous message-passing networks: (i) algorithms that assume reliable message delivery, e.g., [13], [15], [34], [36]; (ii) algorithms that relax termination (namely asymptotic consensus), e.g., [10], [17], [25], [40]; and (iii) algorithms that piggyback the entire history (namely the full-information model) [38]. There are also some algorithms [23], [24] that do not tolerate node faults.

The analysis and design of these prior algorithms do not directly apply to our model because of three main challenges: (i) we cannot implement some well-known primitives such as reliable broadcast [1], [9], because nodes are anonymous, and T and D are unknown; (ii) nodes are not able to piggyback extra information because of the anonymity and limited bandwidth assumption; and (iii) due to anonymity, it is not easy to break a tie.

III. Related Work

Distributed tasks with fault-tolerance (of both node and link faults) have been widely studied [5], [30]. We focus on the closely related work on dynamic networks in this section. Early works [2], [6], [19], [37] focus on networks that eventually stop changing. These algorithms usually guarantee progress or liveness only after the network stabilizes (i.e., when network stops changing).

Subsequently, various works investigate networks with continual dynamic changes. Kuhn et al. propose the idea of T-interval connectivity and study problems on election, counting, consensus, token dissemination, and clock synchronization [20], [22]–[24]. The links in their model are assumed to be bi-directional and nodes are assumed to be correct. Bolomi et al. [8] investigate reliable broadcast primitive in dynamic networks with locally bounded Byzantine adversary. The nodes are assumed to have an identity.

Di Luna et al. have a series of works on anonymous dynamic networks which investigate problems like counting, leader election, and arbitrary function computation [11], [12], [26]–[29]. There is also a line of works on investigating message adversary for consensus in directed dynamic graphs [7], [10], [17], [38], [39]. These work do not assume node faults and link bandwidth is assumed to be unlimited. Therefore, their techniques are quite different from ours.

IV. Crash-Tolerant Approximate Consensus

This section considers the case when nodes may crash. We present a simple algorithm, *DAC*, which stands for Dynamic Approximate Consensus and achieves approximate consensus if $n \geq 2f + 1$ and the graph satisfies $(T, \lfloor n/2 \rfloor)$-dynaDegree. Section VI will prove that these conditions together are necessary. DAC shows that they are sufficient.

DAC is inspired by prior algorithms [13], [30], [36] in which nodes proceed in phases and nodes collect messages from a certain phase to update its local state value and proceed to the next phase. DAC has two key changes:

- The capability of "jumping" to a future phase when receiving a state with a larger phase index; and
- Each node uses a bit vector R_i to keep track of received messages from the same phase. This is possible because of our assumption of port numbering (cf. Section II).

Change (i) avoids sending repetitive state values (state values from prior phases) to handle message loss under limited bandwidth, while preserving the same convergence rate. Change (ii) allows node i to know when it is safe to proceed to the next phase. DAC is presented in Algorithm 1.

Each node i initializes its local state v_i to a given input x_i, and then broadcasts its state v_i in every round. It only stores two states – $v_{min,i}$ and $v_{max,i}$ – the smallest and largest phase-p states observed so far, respectively. Each node i in phase p_i has two ways to proceed to a higher phase:

1) (line 5 – 8): upon receiving a message from phase $q > p_i$, node i directly copies the received state and "jumps" to phase q; or
2) (line 12 – 15): upon receiving $\lfloor \frac{n}{2} \rfloor + 1$ phase-p_i states from different nodes, node i updates its state and proceeds to phase $p + 1$.

Algorithm 1 DAC: Steps at each node i in round t. Node i outputs v_i when $p_i = p_{end}$ (identified in (2))

```
Initialization:
    $v_i, v_{\min,i}, v_{\max,i} \leftarrow x_i$                    ▷ $x_i$ is the input
    $p_i \leftarrow 0$                                              ▷ phase index
    $R_i \leftarrow$ zero vector of length $n$
    $R_i[i] \leftarrow 1$

 1: for $t \leftarrow 0$ to $\infty$ do
 2:     broadcast $\langle i, v_i, p_i \rangle$ to all
 3:     $M_i \leftarrow$ messages received in round $r$
 4:     for each $\langle j, v_j, p_j \rangle$ received from port $j$ in $M_i$ do
 5:         if $p_j > p_i$ then
 6:             $v_i \leftarrow v_j$
 7:             $p_i \leftarrow p_j$
 8:             RESET()
 9:         else if $p_j = p_i$ and $R_i[j] = 0$ then
10:             $R_i[j] \leftarrow 1$
11:             STORE($v_j$)
12:             if $|R_i| \geq \lfloor \frac{n}{2} \rfloor + 1$ then
13:                 $v_i \leftarrow \frac{1}{2}(v_{\min,i} + v_{\max,i})$
14:                 $p_i \leftarrow p_i + 1$
15:                 RESET()
16:         if $p_i = p_{end}$ then                                 ▷ $p_{end}$ in (2)
17:             output $v_i$

18: function RESET()
19:     $R_i[i] \leftarrow 1$;  and $R_i[j] \leftarrow 0, \forall j \neq i$
20:     $v_{\min,i}, v_{\max,i} \leftarrow v_i$

21: function STORE($v_j$)
22:     if $v_j < v_{\min,i}$ then
23:         $v_{\min,i} \leftarrow v_j$
24:     else if $v_j > v_{\max,i}$ then
25:         $v_{\max,i} \leftarrow v_j$
```

Since a node might not receive enough phase-p_i messages in a round, it uses an n-bit bit vector to keep track of the senders. Recall that even though our model does *not* assume node identity, each node can still use receiving ports to distinguish messages received. For brevity, denote the number of ones in vector R_i by $|R_i|$.[3] Every fault-free node repeats this process until it proceeds to the termination phase p_{end}, which will be defined later in Equation (2).

A. Correctness of DAC

Technical Challenge. Intuitively, the algorithm is correct, because a "future" state value is better, in terms of convergence, than the current state value (i.e., the v_i's from the current phase p). In other words, "jump" should not affect correctness. However, because nodes do not use state values from the same phase to update their state values, we cannot directly apply prior proofs (e.g., [1], [13], [30], [34], [36]), which rely on the fact that a pair of nodes receive at least *one common value* for each phase (via a typical quorum intersection argument).

In DAC, nodes may move or jump to a phase, because they do not use the same updating rule. The key challenge is to devise the right setup so that we can use induction to derive a bound on the convergence rate. Our induction-based proof is useful for handling the case when nodes may not receive common values (due to message loss). Intuitively, we need to find a way to replace a common value by a common interval (that contain state values across different phases) for proving convergence.

Algorithm DAC satisfies termination because after T rounds, each node must receive either $\lfloor n/2 \rfloor + 1$ messages in the same phase (including message received from itself) or a message with a higher phase owing to $(T, \lfloor n/2 \rfloor)$-dynaDegree. Validity is also straightforward because of our updating rules and the assumption of non-Byzantine behavior. Hence, we focus on the ϵ-agreement below.

[3]One practical optimization is to use an $(n-1)$-bit bit vector. In our design, $R_i[i]$ is always 1, since each node always has its own state value.

Notations. We introduce two useful notations.

Definition 4. Let S be a finite multiset. Define the *cardinality* $|S|$ as number of elements in S counting multiplicity, the *range* of S as $\text{range}(S) = \max(S) - \min(S)$, and the *interval* of S as $\text{interval}(S) = [\min(S), \max(S)]$.

Definition 5. Define $V^{(p)}$ as a multiset of phase-p states of all nodes that have not crashed yet.

For a faulty node which crashed before phase p, its phase-p state is empty and hence is excluded in $V^{(p)}$. Due to the "jump" feature of DAC, some fault-free nodes may skip a particular phase p. To simplify our analysis, we introduce the following definition:

Definition 6. If a node i jumps from some phase p to phase $q > p$, then we define its phase-p' state value $v_i^{p'}$ of skipped phases ($p < p' < q$) as v_i^q.

Denote $n_p = |V^{(p)}|$. Definition 6 and the assumption that $n \geq 2f + 1$ imply that $n_p \geq n - f \geq \lfloor \frac{n}{2} \rfloor + 1$ for all phase p. Without loss of generality, we order $V^{(p)}$ *chronologically* such that the skipped state values are ordered last (breaking tie arbitrarily). In other words, $V^{(p)} = \{v_1^p, \ldots, v_{n_p}^p\}$, where v_k^p is the phase-p state of the k-th node that proceeded to phase p. For nodes that skip state p, their values appear last in $V^{(p)}$.

With $V^{(p)}$ defined, we can introduce the notion of convergence rate.

Definition 7 (Convergence Rate)**.** Consider an algorithm $\mathcal{A}$ in which each node i maintains a state value v_i. Then we say $\mathcal{A}$ has convergence rate ρ, for some $\rho \in [0, 1]$, if $\text{range}(V^{(p+1)}) \leq \rho \cdot \text{range}(V^{(p)})$.

Finally, we need two more definitions to help our proof. Define $V_k^{(p)} = \{v_1^p, \ldots, v_k^p\}$, i.e., the first k elements in $V^{(p)}$. Define $W^{(p)}$ as $V^{(p)}$ sorted in ascending order of values, i.e., $W^{(p)} = \{w_1^p, \ldots, w_{n_p}^p\}$ such that $w_1^p \leq \ldots \leq w_{n_p}^p$. Note that $W^{(p)}$ is defined with respect to the values, instead of chronological order.

Convergence Proof. For ϵ-agreement, we first prove the following key lemma to identify the convergence rate. Roughly speaking, the convergence rate identifies the ratio that the range of fault-free nodes decreases in each phase. The base case is similar to the "common value" technique in [1], [14], [34]. The difference lies in the inductive step where we need to consider nodes that skip phases due to the "jump" updating rule. Later in Section V, we generalize the technique to handle Byzantine nodes, where the proof naturally becomes more complicated.

Lemma 1. *For each p $(0 \leq p \leq p_{end}-1)$ and $k \in [n_{p+1}]$,*

$$V_k^{(p+1)} \subseteq \left[\frac{w_1^p + w^p_{\lfloor \frac{n}{2} \rfloor + 1}}{2}, \frac{w^p_{n_p - \lfloor \frac{n}{2} \rfloor} + w^p_{n_p}}{2}\right]. \quad (1)$$

Proof. We prove the lemma by induction on k. First, we prove an important claim.

Claim 2. *For every $p \geq 0$, if a node in phase p updates to phase $p+1$ by receiving $\lfloor n/2 \rfloor + 1$ phase-p states, then its new state value v in phase $p+1$ satisfies*

$$v \in \left[\frac{w_1^p + w^p_{\lfloor n/2 \rfloor + 1}}{2}, \frac{w^p_{n_p - \lfloor n/2 \rfloor} + w^p_{n_p}}{2}\right].$$

Proof of Claim 2. In each phase, the state value v_i of each node remains unchanged until the node updates to the next phase. Moreover, line 9 ensures that each state is received at most once by a receiver in each phase. Therefore, if some node in phase p receives $\lfloor n/2 \rfloor + 1$ phase-p states (including from itself), then the smallest $\lfloor n/2 \rfloor + 1$ possible states it can receive are $w_1^p, \ldots, w^p_{\lfloor n/2 \rfloor + 1}$. Similarly, the maximum possible states are $w^p_{n_p}, \ldots, w^p_{n_p - \lfloor n/2 \rfloor}$. In conclusion, the new state in phase $p+1$ falls in the interval in Claim 2. □

Claim 2 proves the base case that Equation (1) holds for fixed $k = 1$ and for every $p \geq 0$ because $V_1^{(p+1)}$ only consists of the state value of one node, which must update to phase $p+1$ by receiving $\lfloor n/2 \rfloor + 1$ phase-p states.

In the induction case, assume Equation (1) holds for every $p \geq 0$ and for some k, and we want to prove Equation (1) for every $p \geq 0$ and for $k+1$. Note that the new node proceeds to phase $p+1$ by either receiving $\lfloor n/2 \rfloor + 1$ phase-p state values (including one from itself) or copying a future state. In the former case, Claim 2 implies the induction statement, whereas in the latter case, the range of $V_{k+1}^{(p+1)}$ is unchanged and therefore Equation (1) again holds for $k+1$. □

Remark 1. By definition, $n \geq n_p$. This implies that $n - \lfloor n/2 \rfloor \geq n_p - \lfloor n/2 \rfloor$, which leads to $\lfloor n/2 \rfloor + 1 \geq n_p - \lfloor n/2 \rfloor$. Since w^p's are ordered in the ascending order, we have $w^p_{\lfloor n/2 \rfloor + 1} \geq w^p_{n_p - \lfloor n/2 \rfloor}$, and thus

$$\begin{aligned} \text{range}(V^{(p+1)}) &\leq \frac{w^p_{n_p - \lfloor n/2 \rfloor} + w^p_{n_p}}{2} - \frac{w_1^p + w^p_{\lfloor n/2 \rfloor + 1}}{2} \\ &\leq \frac{w^p_{n_p} - w_1^p}{2} = \frac{1}{2} \cdot \text{range}(V^{(p)}). \end{aligned}$$

In other words, Algorithm 1 converges with rate $\frac{1}{2}$.

This implies the following theorem, which identifies the phase p_{end} in which node i is able to output v_i.

Theorem 3. *Algorithm DAC satisfies ϵ-agreement after phase p_{end}, where*

$$p_{\text{end}} = \log_{\frac{1}{2}}(\epsilon) \quad (2)$$

Interestingly, the lower bound from [17] shows that $1/2$ is the optimal rate for any fault-tolerant approximate consensus algorithms, even in static graphs. Hence, DAC achieves the optimal convergence rate and optimal resilience even in the static graph with only node crash faults.

V. Byzantine Approximate Consensus

This section considers up to f Byzantine nodes (denoted as set $\mathcal{B}$), and the rest of the nodes (denoted as set $\mathcal{H}$) are fault-free and always follow the algorithm specification. We present an approximate consensus algorithm, *DBAC*, which stands for Dynamic Byzantine Approximate Consensus and is correct if $n \geq 5f+1$ and $G(V,E)$ satisfies $(T, \lfloor (n+3f)/2 \rfloor)$-dynaDegree.

Algorithm DBAC and Algorithm DAC share a similar structure, but DBAC has different update rules to cope with Byzantine faults. Plus, nodes do *not* skip phases in DBAC. The pseudo-code is presented in Algorithm 2.

Each node starts with phase 0 and initializes its local state value v_i to the given input x_i. Then it broadcasts its current local state value in every round. For each node in phase p_i, upon receiving $\lfloor \frac{n+3f}{2} \rfloor + 1$ state values from phase p_i or higher, it updates its local state value v_i to the average of the $(f+1)$-st lowest state value and the $(f+1)$-st highest state value that have been received so far and then proceeds to phase $p+1$. To achieve the goal, node i uses $R_{i,\text{low}}$ and $R_{i,\text{high}}$ – lists that store the $f+1$ lowest and $f+1$ highest received states in phase p or higher, respectively. Recall that $|R_i|$ denotes the number of ones in R_i, whereas $|R_{i,\text{low}}|$ and $|R_{i,\text{high}}|$ denote the cardinality (i.e., the number of elements) of $R_{i,\text{low}}$ and $R_{i,\text{high}}$, respectively.

Our update rule ensures that the new state value falls in the range of fault-free state values despite of the existence of Byzantine messages. Moreover, since at most f nodes are Byzantine faulty and the graph is assumed to satisfy $(T, \lfloor (n+3f)/2 \rfloor)$-dynaDegree, this step is always non-blocking. (Recall that a node can receive a message from itself as well.) Every node repeats this process until phase p_{end}, whose value will be determined later in Equation (6).

DBAC is inspired by the iterative Byzantine approximate consensus algorithm (BAC) [14], which update states using states from the same phase. BAC relies on reliable channels; hence, is not feasible in our dynamic network model. DBAC can update states using messages from different phases (as shown in the STORE$(-)$ function below). These differences allow us to tolerate the nature of dynamic network; however, using messages from different phases make the correctness proof more complicated than prior analysis.

Algorithm 2 DBAC: Steps at each node i in round t. Node i outputs v_i when $p_i = p_{end}$ (identified in (6))

```
Initialization:
    $v_i \leftarrow x_i$                                        ▷ $x_i$ is the input
    $p_i \leftarrow 0$                                          ▷ phase index
    $R_i \leftarrow$ zero vector of length $n$
    $R_i[i] \leftarrow 1$
    $R_{i,\text{low}}, R_{i,\text{high}} \leftarrow \{\}$

1:  for $t \leftarrow 0$ to $\infty$ do
2:      broadcast $\langle i, v_i, p_i \rangle$ to all
3:      $M_i \leftarrow$ messages received in round $r$
4:      for each $\langle j, v_j, p_j \rangle$ from port $j$ in $M_i$ do
5:          if $p_j \geq p_i$ and $R_i[j] = 0$ then
6:              $R_i[j] \leftarrow 1$
7:              STORE($v_j$)
8:          if $|R_i| \geq \lfloor \frac{n+3f}{2} \rfloor + 1$ then
9:              $v_i \leftarrow \frac{1}{2}(\max(R_{i,\text{low}}) + \min(R_{i,\text{high}}))$
10:             $p_i \leftarrow p_i + 1$
11:             RESET()
12:     if $p_i = p_{end}$ then
13:         output $v_i$

14: function RESET()
15:     $R_i[j] \leftarrow 0, \forall j \neq i$
16:     $R_{i,\text{low}}, R_{i,\text{high}} \leftarrow \{\}$

17: function STORE($v_j$)
18:     if $|R_{i,\text{low}}| \leq f + 1$ then
19:         $R_{i,\text{low}} \leftarrow R_{i,\text{low}} \cup \{v_j\}$
20:     else if $v_j < \max(R_{i,\text{low}})$ then
21:         replace max value in $R_{i,\text{low}}$ with $v_j$
22:     if $|R_{i,\text{high}}| \leq f + 1$ then
23:         $R_{i,\text{high}} \leftarrow R_{i,\text{high}} \cup \{v_j\}$
24:     else if $v_j > \min(R_{i,\text{high}})$ then
25:         replace min value in $R_{i,\text{high}}$ with $v_j$
```

There exists a Byzantine approximate consensus algorithm [1] that achieves an optimal resilience $n \geq 3f + 1$ in a static graph with only Byzantine nodes; however, it uses a stronger primitive, reliable broadcast [9], and a technique of witness (of certain state values). Because of the anonymity assumption, such techniques are not possible in our model.

A. Correctness of DBAC

Theorem 4. *Algorithm DBAC satisfies termination.*

Proof. We prove termination by induction on phase $p \geq 0$. Formally, we define the induction statement as: every fault-free node proceeds to phase p for $1 \leq p \leq p_{end}$ within finite number of rounds. The base case holds because all nodes are initially in phase 0. Now suppose all fault-free nodes proceed to phase p within a finite number of rounds. Then after all fault-free nodes are in phase p or higher, by assumption of $(T, \lfloor (n+3f)/2 \rfloor)$-dynaDegree, every fault-free node receive at least $\lfloor (n+3f)/2 \rfloor + 1$ state values from fault-free nodes in phase p or higher within T rounds (including one from itself). Hence, every fault-free node proceeds to the next phase according to line 8 – 11, which proves the induction. □

We define $V^{(p)}$ and $W^{(p)}$ in a similar way as we did in the previous section. The difference is that we only consider "*fault-free* nodes." Recall that in the case of crash faults, $V^{(p)}$ and $W^{(p)}$ might include nodes that crash later in phases after phase p. In the Byzantine case, we exclude any Byzantine state values, as the values are not well-defined.

We then sort $V^{(p)} = \{v_1^p, \ldots, v_{|V^{(p)}|}^p\}$ chronologically, i.e., in the increasing order of round index in which the state value is calculated (breaking ties arbitrarily). Since we have already proved that DBAC terminates, $|V^{(p)}| = h$ for all $p \geq 0$, where h is the number of fault-free nodes and $h = |\mathcal{H}| \geq n - f$. For easiness of calculation, we also introduce the following notations:

Definition 8. Define $W^{(p)} = \{w_1^p, \ldots, w_h^p\}$ as $V^{(p)}$ ordered by values, i.e., $w_1^p \leq \ldots \leq w_h^p$.

Definition 9. Define $U = \{u_1, \ldots, u_b\}$ as a multiset of *arbitrary* values from Byzantine nodes, where b is the number of Byzantine nodes in the execution and $b = |\mathcal{B}| \leq f$.

Definition 10. For round t and phase p ($0 \leq p \leq p_{\text{end}}$), define $k(t, p)$ as the number of fault-free nodes that are in phase p or higher at the start of round t.

Moreover, define $V_t^{(p)} = \{v_1^p, \ldots, v_{k(t,p)}^p\}$, i.e., the first $k(t, p)$ elements in the multiset $V^{(p)}$. If $k(t, p) = 0$, we define $V_t^{(p)} = \emptyset$. In other words, $V_t^{(p)}$ is the multiset of phase-p states of fault-free nodes whose phases are $\geq p$ at the start of round t.

Remark 2. Observe the properties below for $k(t, p)$:

1) For fixed $p \geq 0$, $k(t, p)$ is non-decreasing with respect to t, i.e., $t \leq t'$ implies
$$k(t, p) \leq k(t', p) \text{ and thus } V_t^{(p)} \subseteq V_{t'}^{(p)}.$$
2) For fixed $t \geq 0$, $k(t, p)$ is non-increasing with respect to p, i.e., $p \leq q$ implies
$$k(t, p) \geq k(t, q).[4]$$
3) When $t = 0$, $k(0, 0) = h$ and $k(0, p) = 0$ for all $p > 0$ because all nodes are initially in phase 0. Consequently, $V_0^{(0)} = V^{(0)}$ and $V_0^{(p)} = \emptyset$ for $p > 0$.
4) By termination, every fault-free node updates to phase p_{end} within finite time. Therefore, there exists a finite t_{end} that is the last round in which a fault-free node updates to p_{end}. Moreover, $k(t_{\text{end}}, p) = h$ and $V_{t_{\text{end}}}^{(p)} = V^{(p)}$ for all p.

We are now ready to prove the key lemma that bounds the range of fault-free values. Recall that $\text{interval}(V) = [\min(V), \max(V)]$ and $\text{range}(V) = |\max(V) - \min(V)|$.

[4]Although it still holds that $V_t^{(p)} \supseteq V_t^{(q)}$, the proof is not immediate. This identity turns out to be the key to the proof of Lemma 5.

Lemma 5. *For every round $t \geq 0$,*

$$\text{interval}(V_t^{(q)}) \subseteq \text{interval}(V_t^{(p)}),\ \forall 0 \leq p \leq q. \tag{3}$$

In other words, Lemma 5 suggests that in every round, higher-phase states are within lower-phase states.

Proof. We prove the lemma by induction on t.

In the base case when $t = 0$, recall Remark 2 that $V_0^{(0)} = V^{(0)}$ and $V_0^{(p)} = \emptyset$ for all $p > 0$, and so (3) trivially holds.

For the induction case, assume (3) holds for rounds t, and we want to prove for $t+1$. The key is to show that

$$\text{interval}(V_{t+1}^{(p+1)}) \subseteq \text{interval}(V_t^{(p)}),\ \forall\, p \geq 0. \tag{4}$$

Recall that $V_t^{(p)} \subseteq V_{t+1}^{(p)}$ by Remark 2. Together with (4), we have $\text{interval}(V_{t+1}^{(p+1)}) \subseteq \text{interval}(V_{t+1}^{(p)})$, which proves the induction case. The rest of the proof aims to prove (4).

If no node updates from phase p to $p+1$ in round t, then $V_{t+1}^{(p+1)} = V_t^{(p+1)}$ and (4) follows from the induction assumption. Otherwise, consider some node i that updates from phase p to $p+1$ in round t. Its new state in $V_{t+1}^{(p+1)}$ is of form $v = \frac{1}{2}(\max(R_{i,\text{low}}) + \min(R_{i,\text{high}}))$. For both $R_{i,\text{low}}$ and $R_{i,\text{high}}$, they consist of $f+1$ messages each of which either comes from a Byzantine node or is in $V_t^{(q)}$ for some $q \geq p$. Since there are at most f Byzantine nodes, there exist $q, q' \geq p$ and $u \in V_t^{(q)}, w \in V_t^{(q')}$ such that

$$u \leq \max(R_{i,\text{low}}) \leq \min(R_{i,\text{high}}) \leq w.$$

By induction assumption, $V_t^{(q)}, V_t^{(q')} \subseteq V_t^{(p)}$. Consequently, $u \leq v \leq w$ and $v \in \text{interval}(V_t^{(p)})$. This proves (4). ☐

Lemma 5 implies the validity of Algorithm DBAC upon substituting $t = t_{\text{end}}$, $p = 0$ and $q = p_{\text{end}}$ into Equation (3), which together with Remark 2 implies that

$$V^{(p_{\text{end}})} \subseteq \text{interval}(V^{(0)}).$$

DBAC: ϵ-agreement. We next present the proof for convergence. In addition to the effect of Byzantine values, we also need to consider the case when a node uses values from different phases when updating. This is more complicated to analyze than the case of DAC, since in our prior analysis, a node simply jumps to a future state that trivially satisfies the induction statement. Also, we cannot apply prior proofs for the traditional Byzantine fault model with reliable channel (e.g., [1], [14], [31], [34]) either. This is again because a pair of fault-free nodes may not use a common value to update their future states. A technical contribution is to identify a setup to use induction to prove the convergence.

We need to prove that the new state value at a fault-free node falls in a smaller interval as it updates to a higher phase. In the base case, each node in phase p receives non-Byzantine messages from a fixed multiset $V^{(p)}$. By the classical common value analysis and quorum intersection argument [14], [31], all nodes in the base case must receive at least one common value from a fault-free node.

In the more general inductive step, this technique no longer works, because each node can also receive messages from higher phase(s). We need to show that all fault-free nodes must share some common information. Even though each pair of fault-free nodes may *not* receive a common value, we show that each fault-free node must receive at least one non-Byzantine message in a "*common multiset*" (a generalized concept of common value). We then bound the range of this common multiset using a_k^p and A_k^p, defined below. The common multiset allows us to derive the desired convergence rate.

Definition 11. For each p, define a_k^p and A_k^p recursively as:

$$a_{k+1}^p = (a_k^p + w_1^p)/2,\ \ A_{k+1}^p = (A_k^p + w_h^p)/2,$$

with initial values $a_0^p = A_0^p = w_{2f+1}^p$.

Notice two useful properties. First, since $w_1^p \leq w_{2f+1}^p \leq w_h^p$, we have

$$w_1^p \leq a_k^p \leq w_{2f+1}^p \leq A_k^p \leq w_h^p.$$

Moreover, using geometric series, their explicit formulas are:

$$a_k^p = 2^{-k} w_{2f+1}^p + \sum_{i=1}^{k} 2^{-i} w_1^p = w_1^p + 2^{-k}(w_{2f+1}^p - w_1^p).$$

Similarly, $A_k^p = w_h^p + 2^{-k}(w_{2f+1}^p - w_h^p)$.

We are now ready to present the full proof below, and the illustration of common multiset is presented in Figure 2. Recall that v_k^{p+1} denotes the phase-$(p+1)$ state of the k-th fault-free node that updates to phase $p+1$.

Lemma 6. *Suppose $n \geq 5f+1$. Then for every $k \in [h]$,*

$$v_k^{p+1} \in [a_k^p, A_k^p],\ \forall p \geq 0. \tag{5}$$

Proof. We prove the theorem by induction on k.

Base Case: In base case, fix $k = 1$ and consider phase $p \geq 0$. Assume that node i^* is the first fault-free node that proceeds to phase $p+1$ (breaking ties arbitrarily), and denote by $R_{\text{low}}, R_{\text{high}}$ the recording lists of node i^* in phase p.

By construction, all fault-free states received by i^* must be in phase p. In addition, at most f received state values are Byzantine. Therefore, $\max(R_{\text{low}}) \geq w_1^p$ because $\max(R_{\text{low}})$ reaches its minimum possible value in the worst case when i^* receives these following $\lfloor (n+3f)/2 \rfloor + 1$ states:

$$u_1 \leq \ldots \leq u_f \leq w_1^p \leq \ldots \leq w_{\lfloor (n+3f)/2 \rfloor + 1 - f}^p.$$

Similarly, $\min(R_{\text{high}}) \geq w_{\lfloor (n+3f)/2 \rfloor + 1 - 2f}^p$. Also, since $n \geq 5f+1$, $w_{\lfloor (n+3f)/2 \rfloor + 1 - 2f}^p \geq w_{2f+1}^p$. Therefore,

$$v_1^{p+1} = \frac{\max(R_{\text{low}}) + \min(R_{\text{high}})}{2} \geq \frac{w_1^p + w_{2f+1}^p}{2} = a_1^p.$$

Symmetrically, $\max(R_{\text{low}}) \leq w_{2f+1}^p$ and $\min(R_{\text{high}}) \leq w_{\lfloor (n+3f)/2 \rfloor + 1 - 2f}^p$. Recall that $h = |\mathcal{H}| \geq n - f$ and $n \geq 5f+1$, so $w_{\lfloor (n+3f)/2 \rfloor + 1 - 2f}^p \leq w_h^p$. Thus,

$$v_1^{p+1} \leq \frac{w_{2f+1}^p + w_h^p}{2} = A_1^p.$$

In conclusion, $v_1^{p+1} \in [a_1^p, A_1^p]$, for all $p \geq 0$.

Induction Case: In induction case, assume Equation (5) is true for all $i \in [k]$, and we want to prove for $k+1$. Consider an arbitrary phase p and assume that node j^* is the $(k+1)$-st fault-free node that proceeds to phase $p+1$ in round t. Denote $R_{\text{low}}, R_{\text{high}}$ as recording lists of node j^* in phase p, and denote the received state values of node j^* as $r_1 \leq \ldots \leq r_{\lfloor (n+3f)/2 \rfloor + 1}$. Then $\max(R_{\text{low}}) = r_{f+1}$ and $\min(R_{\text{high}}) = r_{\lfloor (n+3f)/2 \rfloor + 1 - f}$.

Note that every received state value r must come from one of the three possible sources: (i) a Byzantine node; (ii) $r \in V_t^{(p)}$; or (iii) $r \in V_t^{(q)}$ for some $q > p$. Let's consider the latter two cases.

First case: Suppose $r \in V_t^{(q)}$ for some $q > p$. Then

$$r \in \text{interval}(V_t^{(q)}) \overset{(i)}{\subseteq} \text{interval}(V_t^{(p+1)}) \overset{(ii)}{\subseteq} [a_k^p, A_k^p].$$

Here (i) follows from Lemma 5 and (ii) follows from the induction assumption.

Second case: Suppose $r \in V_k^{(p)}$. Recall that $a_k^p \leq w_{2f+1}^p \leq A_k^p$, so we can partition $\text{interval}(V^{(p)})$ into three parts as in Figure 2: $V_1 = [w_1^p, a_k^p)$, $V_2 = [a_k^p, A_k^p]$, and $V_3 = (A_k^p, w_h^p]$.

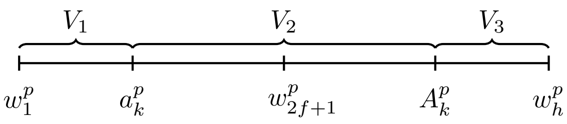

Fig. 2: Partition of $\text{interval}(V^{(p)})$ into V_1, V_2 and V_3.

By definition of w_{2f+1}^p, it is the $(2f+1)$-st largest state in $V^{(p)}$. Since $a_k^p \leq w_{2f+1}^p$, $r < a_k^p$ implies $r < w_{2f+1}^p$ and thus at most $2f$ state values from $V^{(p)}$ fall in V_1.

In conclusion, among the $\lfloor (n+3f)/2 \rfloor + 1$ received state values, at most f are Byzantine, and the rest are fault-free and greater than or equal to w_1^p. Moreover, among the fault-free state values, at most $2f$ are less than a_k^p. Hence, $r_{f+1} \geq w_1^p$ and $r_{3f+1} \geq a_k^p$. Finally, since $n \geq 5f+1$, $r_{\lfloor (n+3f)/2 \rfloor + 1 - f} \geq r_{3f+1}$ and thus

$$v_{k+1}^{p+1} = \frac{r_{f+1} + r_{\lfloor (n+3f)/2 \rfloor + 1 - f}}{2} \geq \frac{w_1^p + a_k^p}{2} = a_{k+1}^p.$$

Symmetrically, at most $h - (2f+1)$ fault-free states from $V^{(p)}$ fall in V_3 (i.e., greater than A_k^p). Therefore, $r_{\lfloor (n+3f)/2 \rfloor + 1 - f} \leq w_h^p$ and $r_{f+1} \leq A_k^p$,[5] and thus $v_{k+1}^{p+1} \leq A_{k+1}^p$. This proves Equation (5) for $k+1$. □

Theorem 7. *Algorithm DBAC satisfies ϵ-agreement.*

Proof. Lemma 6 implies that $\text{interval}(V^{(p+1)}) \subseteq [a_n^p, A_n^p]$. Furthermore, by definition and geometric series, we have

$$a_n^p = w_1^p + 2^{-n}(w_{2f+1}^p - w_1^p), \quad \text{and}$$
$$A_n^p = w_h^p + 2^{-n}(w_{2f+1}^p - w_h^p).$$

and $w_1^p = \min V^{(p)}$ and $w_h^p = \max V^{(p)}$. Hence, $A_n^p - a_n^p$ gives an upper bound of convergence rate, which is $1 - 2^{-n}$ because $\text{range}(V^{(p+1)}) \leq (1 - 2^{-n}) \, \text{range}(V^{(p)})$.

In conclusion, DBAC satisfies ϵ-agreement at p_{end}, where

$$p_{\text{end}} = \frac{\log \epsilon}{\log(1 - 2^{-n})}, \tag{6}$$

because for all $p \geq p_{\text{end}}$, $\text{range}(V^{(p)}) \leq (1 - 2^{-n})^{p_{\text{end}}} \leq \epsilon$. □

VI. Impossibility Results

A. Exact Consensus

We first show that $(1, n-2)$-dynaDegree is not sufficient for solving binary exact consensus, even when all nodes are fault-free. Gafni and Losa prove the following theorem for a complete graph in a recent paper [18]:[6]

Theorem 8 (from [18]). *Consider a synchronous model where in every round, each node might fail to receive one of the messages sent to it. It is impossible to achieve deterministic binary exact consensus, even when all nodes are fault-free.*

This theorem implies the following corollary, since by the definition of $(1, n-2)$-dynaDegree, the message adversary can force any node to drop any single message sent to it.

Corollary 1. *It is impossible to achieve deterministic binary exact consensus in the anonymous dynamic network with $(1, n-2)$-dynaDegree, even when all nodes are fault-free.*

B. Crash-tolerant Approximate Consensus

Theorem 9. $(T, \lfloor \frac{n}{2} \rfloor)$*-dynaDegree and $n \geq 2f+1$ are together necessary for solving deterministic crash-tolerant approximate consensus.*

Proof Sketch. The proof consists of two parts. First, we show that it is impossible to achieve deterministic approximate consensus in an anonymous dynamic network with $(1, \lfloor \frac{n}{2} \rfloor - 1)$-dynaDegree, even when all nodes are fault-free. The proof for the first part is by contradiction. Assume that there exists a deterministic approximate consensus algorithm in a dynamic graph with $(1, \lfloor \frac{n}{2} \rfloor - 1)$-dynaDegree. However, to satisfy termination, a node i must be able to make decision after communicating with only $\lfloor \frac{n}{2} \rfloor$ nodes (including i itself). Therefore, it is possible for the message adversary to pick $\mathcal{E}(t)$ in a way that there are two non-overlapping groups of nodes that do not communicate with each other, making ϵ-agreement impossible when these two group of nodes have different inputs.

Second, we show that there exists a finite T' such that it is impossible to achieve deterministic approximate consensus in an anonymous dynamic network with $(T', n-1)$-dynaDegree and $n \leq 2f$. The reason that we cannot prove

[5] At most $h - (3f+1)$ states are greater than A_k^p, so the $((n-f) - [h - (3f+1)])$-th state is less than or equal to A_k^p. Since $n \geq 5f+1$ and $h \leq n$, $(n-f) - [h - (3f+1)] \geq f+1$. This implies $r_{f+1} \leq r_{(n-f)-[h-(3f+1)]} \leq A_k^p$.

[6] As noted in [18], the following theorem is different from, although similar to, the main result from the seminal paper by Santoro and Widmaye [33]. The model in [33] considers a synchronous system in which one node fails to send some of its messages per round.

for a general T in this case is that it is trivial to design an algorithm that works for a fixed number of T, as each node can simply repeat the same process for T rounds. However, we argue that it is impossible to do so with an unknown T.

Assume that there exists a deterministic approximate consensus algorithm $\mathbb{A}$ in a dynamic graph with $n \leq 2f$ and $(T, n-1)$-dynaDegree for a fixed T. Now, we need to find a T' so that $\mathbb{A}$ is incorrect, deriving a contradiction.

Observe that to satisfy termination after f nodes crash before the execution of the algorithm, a node i must be able to make decision after communicating with only $\leq f$ nodes, since $n \leq 2f$. Without loss of generality, assume that in this scenario, namely Scenario 1, $\mathbb{A}$ takes R rounds.

Next we show that by choosing $T' = R+1$, $\mathbb{A}$ is incorrect. Consider the graph with $(R+1, n-1)$-dynaDegree. The message adversary can then pick $\mathcal{E}(t)$ for $1 \leq t \leq R$ so that there are two non-overlapping groups of nodes that do not communicate with each other. This is possible because (i) this scenario is indistinguishable from Scenario 1, so nodes must output in R rounds; (ii) within R rounds, nodes only communicate with $\leq f$ nodes in $\mathbb{A}$; and (iii) $n \leq 2f$.

Finally, consider the execution where these two non-overlapping groups are given different input value 0 and 1, respectively. Since each group makes a decision without communicating with another group, ϵ-agreement is violated in $\mathbb{A}$, a contradiction. □

C. Byzantine Approximate Consensus

Theorem 10. *$(1, \lfloor \frac{n+3f}{2} \rfloor)$-dynaDegree and $n \geq 5f$ are together necessary for solving deterministic Byzantine approximate consensus.*

Proof. The proof also consists of two parts. First, we show that it is impossible to achieve deterministic approximate consensus in the anonymous dynamic network with $(1, \lfloor \frac{n+3f}{2} \rfloor - 1)$-dynaDegree and $n \geq 3f+1$.[7] Assume that there exists a deterministic Byzantine approximate consensus algorithm $\mathbb{A}$.

Observe that to satisfy termination, a node i must be able to make decision after communicating with only $\lfloor \frac{n+3f}{2} \rfloor$ nodes, including i itself. Consider the following scenario:

- Divide nodes into two groups: group A contains node $i_1, i_2, \ldots, i_{\lfloor \frac{n+3f}{2} \rfloor}$, and group B contains $i_{\lfloor \frac{n-3f}{2} \rfloor+1}, \ldots, i_n$. Note that each group has size $\lfloor \frac{n+3f}{2} \rfloor$, and there are $3f$ nodes in the intersection of the two groups.
- Nodes $i_{\lfloor \frac{n-f}{2} \rfloor+1}, \ldots i_{\lfloor \frac{n+f}{2} \rfloor}$ are Byzantine faulty.
- The message adversary picks $\mathcal{E}(t)$ in a way that nodes in group A receive only messages from group A and nodes in group B receive only messages from group B.
- Nodes $i_1, \ldots, i_{\lfloor \frac{n-f}{2} \rfloor}$ have input 0.
- Nodes $i_{\lfloor \frac{n+f}{2} \rfloor+1}, \ldots, i_n$ have input 1.
- Byzantine nodes have the following behavior: they behave to group A, as if they had input 0; and behave to group B, as if they had input 1. This is possible because of the anonymity assumption. Because the port numbering is potentially different at each node, Byzantine nodes have the ability to equivocate without being caught. In other words, useful primitives like reliable broadcast is impossible.

Now, we make the following observations:

- From the perspective of group A, only nodes $i_{\lfloor \frac{n+f}{2} \rfloor+1}, \ldots, i_{\lfloor \frac{n+3f}{2} \rfloor}$ have input 1. The rest have input 0. Note that the number of nodes with input 1 in this case is exactly f.
- From the perspective of group B, only nodes $i_{\lfloor \frac{n-3f}{2} \rfloor+1}, \ldots, i_{\lfloor \frac{n-f}{2} \rfloor}$ have input 0. The rest have input 1. Note that the number of nodes with input 0 in this case is exactly f.

To satisfies validity, the first observation forces nodes in group A to output 0, because there are only f nodes with input 1 and all of them could be Byzantine faulty. If group A outputs 1 in this scenario, then it is straightforward to construct an indistinguishable scenario such that only nodes $i_{\lfloor \frac{n+f}{2} \rfloor+1}, \ldots, i_{\lfloor \frac{n+3f}{2} \rfloor}$ are Byzantine faulty (and nodes $i_{\lfloor \frac{n-f}{2} \rfloor+1}, \ldots i_{\lfloor \frac{n+f}{2} \rfloor}$ are fault-free), causing a violation of validity.

Similarly, group B must output 1, violating the ϵ-agreement property.

The second part of proving that there exists a finite T' such that it is impossible to achieve deterministic Byzantine approximate consensus in the anonymous dynamic network with $n \leq 5f$ and $(T', n-1)$-dynaDegree is similar to the case of crash faults. We omit it here for lack of space. □

VII. Conclusion and Discussion

In conclusion, we study the feasibility of fault-tolerant consensus in anonymous dynamic networks. We identify the necessary and sufficient conditions for solving crash-tolerant and Byzantine approximate consensus.

There are many interesting open problems in our model:

- We assume that nodes do not know the base graph G. Does knowing the graph help?
- In practical applications, nodes might only know the IDs for a small set of other nodes. Does this knowledge help in increasing resilience or reducing the requirement for dynamic degree?
- What is the optimal convergence rate for Byzantine approximate consensus algorithms?
- Observe that both our algorithms complete in $T \cdot p_{end}$ rounds in the worst case. For practical applications, it is useful to assume a probabilistic message adversary that picks $\mathcal{E}(t)$ randomly and investigate algorithms achieving the optimal expected number of rounds.
- With unlimited bandwidth, one can indeed simulate the algorithm in [13] by piggybacking the entire history of each node's past messages when sending the current

[7]The necessity of $n \leq 3f$ is from prior work [5], [30].

state value. This achieves convergence rate of $1/2$. In fact, DBAC can improve the convergence rate by piggybacking a limited set of old messages, under limited bandwidth. It is interesting to identify the trade-off between bandwidth and convergence rate.

References

[1] I. Abraham, Y. Amit, and D. Dolev. Optimal resilience asynchronous approximate agreement. volume 3544, pages 229–239, 12 2004.

[2] Y. Afek, B. Awerbuch, and E. Gafni. Applying static network protocols to dynamic networks. In *28th Annual Symposium on Foundations of Computer Science*, pages 358–370. IEEE Computer Society, 1987.

[3] D. Angluin. Local and global properties in networks of processors (extended abstract). In *Proceedings of the Twelfth Annual ACM Symposium on Theory of Computing*, STOC '80.

[4] D. Angluin, J. Aspnes, D. Eisenstat, and E. Ruppert. On the power of anonymous one-way communication. In *Principles of Distributed Systems, 9th International Conference, OPODIS 2005*.

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

[6] B. Awerbuch, B. Patt-Shamir, D. Peleg, and M. E. Saks. Adapting to asynchronous dynamic networks (extended abstract). In *Proceedings of the 24th Annual ACM Symposium on Theory of Computing, May 4-6, 1992, Victoria, British Columbia, Canada*.

[7] M. Biely, P. Robinson, U. Schmid, M. Schwarz, and K. Winkler. Gracefully degrading consensus and *k*-set agreement in directed dynamic networks. *Theor. Comput. Sci.*, 726:41–77, 2018.

[8] S. Bonomi, G. Farina, and S. Tixeuil. Reliable broadcast in dynamic networks with locally bounded byzantine failures. *Stabilization, Safety, and Security of Distributed Systems - 20th International Symposium, SSS 2018, Tokyo, Japan, November 4-7, 2018, Proceedings*.

[9] G. Bracha. Asynchronous Byzantine agreement protocols. *Information and Computation*, 75(2):130–143, 1987.

[10] B. Charron-Bost, M. Függer, and T. Nowak. Approximate consensus in highly dynamic networks: The role of averaging algorithms. In *Automata, Languages, and Programming - 42nd International Colloquium, ICALP 2015, Kyoto, Japan, July 6-10, 2015, Proceedings, Part II*.

[11] G. Di Luna and R. Baldoni. Non Trivial Computations in Anonymous Dynamic Networks. In E. Anceaume, C. Cachin, and M. Potop-Butucaru, editors, *19th International Conference on Principles of Distributed Systems (OPODIS 2015)*, volume 46 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 33:1–33:16, Dagstuhl, Germany, 2016. Schloss Dagstuhl – Leibniz-Zentrum für Informatik.

[12] G. A. Di Luna and R. Baldoni. Brief announcement: Investigating the cost of anonymity on dynamic networks. In *Proceedings of the 2015 ACM Symposium on Principles of Distributed Computing*, PODC '15, page 339–341, New York, NY, USA, 2015. Association for Computing Machinery.

[13] 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(3):499–516, 1986.

[14] 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.

[15] A. D. Fekete. Asymptotically optimal algorithms for approximate agreement. In *Proceedings of the fifth annual ACM symposium on Principles of distributed computing*, PODC '86, pages 73–87, New York, NY, USA, 1986. ACM.

[16] F. E. Fich and E. Ruppert. Hundreds of impossibility results for distributed computing. *Distributed Comput.*, 16(2-3):121–163, 2003.

[17] M. Függer, T. Nowak, and M. Schwarz. Tight bounds for asymptotic and approximate consensus. *J. ACM*, 68(6):46:1–46:35, 2021.

[18] E. Gafni and G. Losa. Invited paper: Time is not a healer, but it sure makes hindsight 20:20. In *Stabilization, Safety, and Security of Distributed Systems - 25th International Symposium, SSS 2023, Jersey City, NJ, USA, October 2-4, 2023, Proceedings*.

[19] R. Ingram, P. Shields, J. E. Walter, and J. L. Welch. An asynchronous leader election algorithm for dynamic networks. In *23rd IEEE International Symposium on Parallel and Distributed Processing, IPDPS 2009, Rome, Italy, May 23-29, 2009*, pages 1–12. IEEE, 2009.

[20] F. Kuhn, T. Locher, and R. Oshman. Gradient clock synchronization in dynamic networks. In *SPAA 2009: Proceedings of the 21st Annual ACM Symposium on Parallelism in Algorithms and Architectures, Calgary, Alberta, Canada, August 11-13, 2009*.

[21] F. Kuhn, N. A. Lynch, and C. C. Newport. The abstract MAC layer. In *Distributed Computing, 23rd International Symposium, DISC 2009, Elche, Spain, September 23-25, 2009. Proceedings*.

[22] F. Kuhn, N. A. Lynch, and R. Oshman. Distributed computation in dynamic networks. In L. J. Schulman, editor, *Proceedings of the 42nd ACM Symposium on Theory of Computing, STOC 2010, Cambridge, Massachusetts, USA, 5-8 June 2010*, pages 513–522. ACM, 2010.

[23] F. Kuhn, Y. Moses, and R. Oshman. Coordinated consensus in dynamic networks. In *Proceedings of the 30th Annual ACM Symposium on Principles of Distributed Computing, PODC 2011, San Jose, CA, USA, June 6-8, 2011*.

[24] F. Kuhn and R. Oshman. Dynamic networks: models and algorithms. *SIGACT News*, 42(1):82–96, 2011.

[25] H. LeBlanc, H. Zhang, X. Koutsoukos, and S. Sundaram. Resilient asymptotic consensus in robust networks. *IEEE Journal on Selected Areas in Communications: Special Issue on In-Network Computation*, 31:766–781, April 2013.

[26] G. A. D. Luna, R. Baldoni, S. Bonomi, and I. Chatzigiannakis. Conscious and unconscious counting on anonymous dynamic networks. In *Distributed Computing and Networking - 15th International Conference, ICDCN 2014, Coimbatore, India, January 4-7, 2014. Proceedings*.

[27] G. A. D. Luna, R. Baldoni, S. Bonomi, and I. Chatzigiannakis. Counting in anonymous dynamic networks under worst-case adversary. In *IEEE 34th International Conference on Distributed Computing Systems, ICDCS 2014, Madrid, Spain, June 30 - July 3, 2014*, pages 338–347. IEEE Computer Society, 2014.

[28] G. A. D. Luna and G. Viglietta. Brief announcement: Efficient computation in congested anonymous dynamic networks. In *Proceedings of the 2023 ACM Symposium on Principles of Distributed Computing, PODC 2023, Orlando, FL, USA, June 19-23, 2023*.

[29] G. A. D. Luna and G. Viglietta. Optimal computation in leaderless and multi-leader disconnected anonymous dynamic networks. In *37th International Symposium on Distributed Computing, DISC 2023, October 10-12, 2023, L'Aquila, Italy*.

[30] N. Lynch. *Distributed Algorithms*. Morgan Kaufman, 1996.

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

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

[33] N. Santoro and P. Widmayer. *Time is not a healer*. In Proceedings of the 6th Annual Symposium on Theoretical Aspects of Computer Science on STACS 89, Feb. 1, 1989, pp. 304–313.

[34] D. Sakavalas and L. Tseng. *Network Topology and Fault-Tolerant Consensus*, volume 9. 05 2019.

[35] L. Tseng and Q. Zhang. Brief announcement: Computability and anonymous storage-efficient consensus with an abstract mac layer. In *PODC '22: ACM Symposium on Principles of Distributed Computing, Italy, 2022*. ACM, 2022.

[36] N. H. Vaidya, L. Tseng, and G. Liang. Iterative approximate byzantine consensus in arbitrary directed graphs. In *ACM Symposium on Principles of Distributed Computing, PODC '12, Funchal, Madeira, Portugal, July 16-18, 2012*.

[37] J. E. Walter, J. L. Welch, and N. H. Vaidya. A mutual exclusion algorithm for ad hoc mobile networks. *Wirel. Networks*, 7(6):585–600, 2001.

[38] K. Winkler, A. Paz, H. R. Galeana, S. Schmid, and U. Schmid. The time complexity of consensus under oblivious message adversaries. In *14th Innovations in Theoretical Computer Science Conference, ITCS 2023, January 10-13, 2023, MIT, Cambridge, Massachusetts, USA*.

[39] K. Winkler, M. Schwarz, and U. Schmid. Consensus in rooted dynamic networks with short-lived stability. *Distributed Comput.*, 32(5):443–458, 2019.

[40] H. Zhang and S. Sundaram. Robustness of complex networks with implications for consensus and contagion. In *Proceedings of CDC 2012, the 51st IEEE Conference on Decision and Control*, 2012.