Time-Optimal and Energy-Efficient Deterministic Consensus

Shachar Meir
Weizmann Institute of Science, Rehovot, Israel

Hugo Mirault
Augusta University, GA, USA

David Peleg
Weizmann Institute of Science, Rehovot, Israel

Peter Robinson
Augusta University, GA, USA

Abstract

We study fault-tolerant consensus in a variant of the synchronous message passing model, where, in each round, every node can choose to be awake or asleep. This is known as the sleeping model (Chatterjee, Gmyr, Pandurangan PODC 2020) and defines the awake complexity (also called *energy complexity*), which measures the maximum number of rounds that any node is awake throughout the execution. Only awake nodes can send and receive messages in a given round and all messages sent to sleeping nodes are lost. We present new deterministic consensus algorithms that tolerate up to $f < n$ crash failures, where n is the number of nodes. Our algorithms match the optimal time complexity lower bound of $f + 1$ rounds. For multi-value consensus, where the input values are chosen from some possibly large set, we achieve an energy complexity of $\mathcal{O}(\lceil f^2/n \rceil)$ rounds, whereas for binary consensus, we show an algorithm to achieve $\mathcal{O}(\lceil f/\sqrt{n} \rceil)$ energy complexity.

2012 ACM Subject Classification Theory of computation → Distributed algorithms

Keywords and phrases Distributed computing, Crash faults, Consensus, Energy complexity, Sleeping model

Digital Object Identifier 10.4230/LIPIcs.OPODIS.2025.15

Related Version *Previous Brief Annoucement*: `https://dl.acm.org/doi/abs/10.1145/3732772.3733554` [23]

Funding Hugo Mirault and Peter Robinson were supported in part by National Science Foundation (NSF) Grant No. CCF-2402836.

Eligibility The paper is eligible for the best student paper award.

1 Introduction

Reaching agreement in a network of distributed processes in the presence of faults is a fundamental problem in distributed computing with numerous applications, including blockchain networks and synchronization problems; e.g., see [28, 1, 21]. In the consensus problem [27, 18], there is a set of players (or nodes) that communicate with each other by sending point-to-point messages. Each player holds an initial value, and the goal is that all of them decide on the same value eventually.

While the problem can be solved in just a single round of communication assuming that all players operate correctly, achieving consensus becomes significantly more challenging when taking into account that some of the players may unexpectedly fail throughout the execution. The seminal result of [14] showed that consensus is impossible with a deterministic algorithm in asynchronous networks, when even just a single player may crash. Moreover, [9] proved

that $f+1$ rounds are indeed necessary in the synchronous setting, when considering up to f crash failures, and a simple algorithm matches this lower bound on the time complexity.

Even though the time complexity of deterministic consensus has been resolved, the actual number of rounds that a player needs to actively participate in the protocol is less clear. This motivates studying consensus in the sleeping model, where, in every round, each player chooses to either be asleep or awake, and the *energy complexity* of the algorithm is defined as the maximum number of awake rounds over all players in the worst case. The sleeping model was first introduced in [6] for studying the (average) energy complexity[1] of computing a maximal independent set and has since received significant attention recently; e.g., see [10, 16, 15, 17, 3]. The sleeping model can be interpreted as a simplified variant of the class of models studied in [4, 5], where the main focus is on wireless networks.

In [26], Pass and Shi introduced the *sleepy model of consensus* in the context of Blockchain networks with Byzantine players, where the players exhibit sporadic participation: A player can be either online (alert) or offline (asleep), and it is possible that only a fraction of the players participate in the computation at any given point in time. A player's current participation status may change as determined by the adversary. Moreover, once an asleep player becomes online, it is guaranteed to receive all pending messages that were addressed to it while sleeping. The sleepy model has become a standard model for studying blockchain protocols under dynamic participation and has been studied extensively since its inception; e.g., see [12, 22, 24, 25, 11, 7, 13]. For instance, a constant-latency protocol in this setting is given in [24], and the constant factor was further reduced in [22]. A variant of the sleepy model where the set of players is infinite is considered in [20]. A generalized version of the sleepy model was studied in [8]. Note that the assumptions of the sleepy model are different from the sleeping model of [6] considered here, which assumes that the sleeping schedule of the players is entirely under the control of the algorithm and all messages sent to a currently sleeping player are lost.

Our Contributions

We establish the first energy efficient algorithms for consensus under crash faults in the sleeping model, while still obtaining optimal time complexity. In more detail, we present the following new deterministic algorithms:

- Multi-value consensus can be solved with an energy complexity of $\mathcal{O}(\lceil \frac{f^2}{n} \rceil)$ rounds, an optimal time complexity of $f+1$, while sending at most $\mathcal{O}(f^3+nf)$ messages.
- Binary consensus can be solved with an energy complexity of $\mathcal{O}(\lceil \frac{f}{\sqrt{n}} \rceil)$ rounds, an optimal time complexity of $f+1$, and with a message complexity of $\mathcal{O}(nf)$.

We point out that none of the prior works in the sleeping model considers faulty nodes, and thus our results also shed some light on the impact of faults on the energy complexity.

2 Model and Preliminaries

We consider the standard synchronous message-passing model of distributed computing, where n players $p_0, \ldots, p_{n-1}$ are connected in a clique network. Each player can choose to either *sleep* or be *awake* in each round, which is known as the *sleeping model*. If a player p is asleep in some round, then any message sent to p in this round is lost; in particular, p

[1] The terms "energy complexity" and "awake complexity" are used interchangeably in the literature. In this work, we use energy complexity.

does *not* receive these messages upon its next wake-up. In this work, we extend the above model with *crash faults*, which means that an adversary can crash up to f players throughout the execution, for some parameter $f < n$. As we only consider deterministic algorithms, we assume that the adversary is omniscient. If a player p *crashes* in round r, then only an arbitrary subset of messages sent by p are guaranteed to be delivered. Moreover, p does not perform any computation in any future round after crashing. Any player who does not crash in rounds $1, \ldots, r-1$ is said to be *alive in round* r If a player does not crash until it has decided, we say that it is *non-faulty*.

It is possible in the sleeping model that there are certain rounds in which no players are awake, e.g., because the adversary crashed the few player that tried to wake up. Consequently, any algorithm that solves consensus in optimal time needs to implement an awake schedule that ensures sufficient progress towards a common decision in each round.

Complexity measures. We quantify the performance of our algorithms in terms of the *message complexity*, i.e., the total number of messages sent in the worst case, and the *time complexity*, which captures the worst-case number of rounds until all players have terminated. In the sleeping model, we consider an additional measure:

- the *(worst case) energy complexity* is the maximum number of rounds (taken over all players) that a player is awake in the worst case execution;

Consensus. Every player starts with an input value that is assigned from a set D of integers. In Section 3, we assume that D is of size at most polynomial in n, whereas in Section 4, we restrict D to be $\{0, 1\}$. An algorithm solves consensus if, in any execution and for any input assignment, the following properties hold:

- *Termination*: Every non-faulty player eventually decides on some value.
- *Agreement*: If some player decides on v in an execution α, then every other player who decides must also decide on v.
- *Validity*: Any decision value must be the input value of some (possibly faulty) player.

3 Multi-value Consensus

In this section, we give an algorithm for the setting, where the input values of the players are chosen from an arbitrary subset of the integers. We present the detailed pseudo-code in Algorithm 2.

3.1 Description of the Algorithm

At the beginning of round 1, we fix f static committees $C_1, \ldots, C_f$, each containing $f+1$ members. The players use their IDs to determine which committee to join. That is, each player locally executes a function $\mathsf{JoinComm}(x, y, n)$, which constructs x committees of size y, using all n players, such that each player's ID is in at most $\lceil \frac{f(f+1)}{n} \rceil$ committees. Note that the procedure's third parameter, i.e, the number of players used to construct the committees, will be utilized carefully in Section 4. For now, the reader should focus on the choice of the first two parameters.

In Phase 1, which consists of round 1, all players awake simultaneously and send their initial values to committee C_1. Each player $p \in C_1$ keeps track of the maximum received value in a variable Y that is initialized to p's input. Phase 2 consists of $f-1$ rounds, where all members of committee C_r are awake in rounds r and $r+1$. In round r, they receive the values sent by committee C_{r-1} and each member computes its new maximum value locally.

Algorithm 1 Procedure JoinComm(a, b, c) assigns the first c players to a committees, each of size b.

Parameters : a : number of committees
b : size of committees
c : number of players to use (starting from the first one)

$C_1, \cdots, C_a \leftarrow \emptyset$
foreach $1 \leq i \leq a \cdot b$ **do**
  $j \leftarrow i \mod c$
  $k \leftarrow \lceil i/b \rceil$
  Add p_j in set C_k
Return $C_1, \cdots, C_a$

Algorithm 2 Multi Value Consensus. Code for player p.

`// Phase 1 (round 1)`
`Player` p `wakes up in round 1:`
$C_1, \cdots, C_{f+1} \leftarrow$ JoinComm$(f, f+1)$
$Y \leftarrow X$
Send Y to every player in C_1
upon *receiving values* Y' **do:**
  $Y \leftarrow \max(Y \cup (\bigcup Y'))$

`// Phase 2 (rounds` $2, \ldots, f$`)`
For each round $r \in [2, f]$ **do**:
if $p \in C_{r-1}$ **then**
  `Wake up in round` r
  Send Y to every player in C_r
if $p \in C_r$ **then**
  `Wake up in round` r
  upon *receiving values* Y' **do:**
    $Y \leftarrow \max(Y \cup (\bigcup Y'))$

`// Phase 3 (round` $f+1$`)`
`Player` p `wakes up in round` $f+1$
if $p \in C_f$ **then**
  Broadcast Y to all
upon *receiving values* Y' **do:**
  $Y \leftarrow \max(Y \cup (\bigcup Y'))$
Decide on Y at the end of round $f+1$

In round $r+1$, each player in C_r forwards this value to the next committee C_{r+1}. Finally, Phase 3 consists of round $f+1$, where everyone wakes up, and the committee members of C_f broadcast their values, and everyone decides on the maximum value that they have seen. See Figure 1 for a high-level overview of this process.

▶ **Theorem 1.** *Suppose that each player starts with an input value from a set of integers D. Algorithm 2 solves consensus in $f+1$ rounds, sends $\mathcal{O}(f^3 + nf)$ messages of size $\mathcal{O}(\log |D|)$ bits, and achieves an energy complexity of $\mathcal{O}(\lceil f^2/n \rceil)$ rounds, if at most f players crash during the execution.*

3.2 Proof of Theorem 1

We first show that the algorithm correctly solves consensus. Every correct player decides on its variable Y after $f+1$ rounds, which ensures the claimed bound on the time complexity. By the code of the algorithm, Y is only updated to the input values of other players, and thus we obtain the following.

▶ **Lemma 2.** *Algorithm 2 satisfies validity and terminates in $f+1$ rounds.*

To show that the algorithm satisfies agreement, we need to prove several technical lemmas. Consider the set D of values decided by the players after round $f+1$ in some given execution, and let $\delta = \max(D)$. We say that *a player p has value v* if p's variable $Y = v$ at that point.

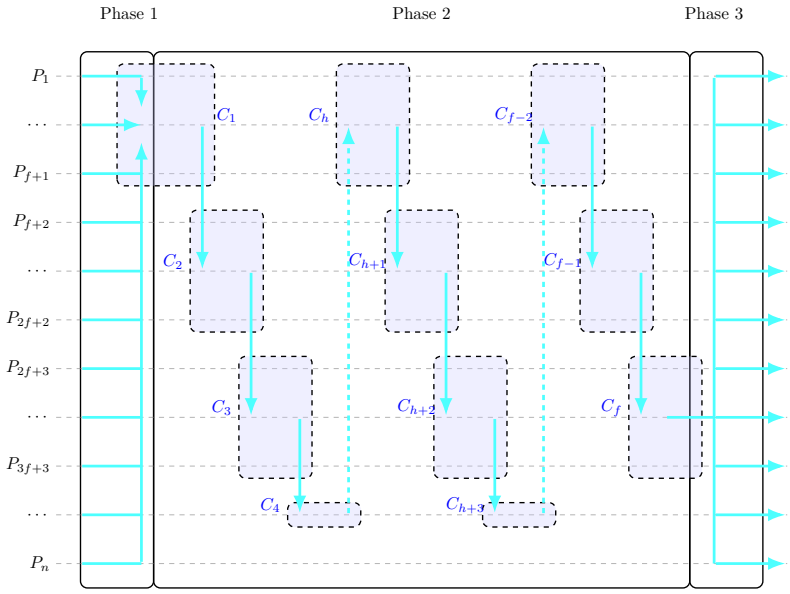

Figure 1 High-level Overview of Algorithm 2.

▶ **Lemma 3.** *Suppose that there exists some round $r \in [1, f+1]$ with the property that there is an awake player p, who is alive in round r and sends a message with value $v_p > \delta$. Then no player decides on δ.*

Proof. Consider any round r that satisfies the premise of the claim. It follows that p successfully sends its message to all destinations. In the case that $r = f+1$, this means that $p \in C_f$, and every player receives p's value in round $f+1$, which is strictly greater than δ. Hence, none of the players will decide on δ. On the other hand, if $r \leq f$, then every alive member $p' \in C_r$ receives the message from p and updates its variable Y such that $Y > \delta$. Since not all members of C_r can crash, it follows that there is some alive player in C_r, who is awake and alive in round $r+1$ and who sends a value strictly greater than δ. Extending this argument inductively for all rounds $r, \ldots, f+1$, completes the proof of the lemma. ◀

▶ **Lemma 4.** *Consider any round $r \in [1, f]$ and let $m \geq \delta$ be the maximum value held by any player in C_r at the start of round $r+1$. Then, every alive player has a value of at most m at the beginning of round $r+1$.*

Proof. The statement trivially holds for any alive player in C_r. Now, consider some alive player $p \notin C_r$ and assume, towards a contradiction, that p's value is $v > m$ when it starts round $r+1$. First, consider the case where v was p's input value. This means that p sent v to every player in C_1 initially. Since p did not crash, every $p' \in C_1$ must have set Y to some value $v_{p'} \geq v$ at the end or round 1 and subsequently sends $v_{p'}$ to everyone in C_2 in round 2. Recall that every committee has at least one member who does not crash. Thus, it follows by induction that every player $q \in C_{r'}$ $(r' \geq 2)$ will receive a message with a value no smaller than v in round r' from some alive player in $C_{r'-1}$ and update its own value accordingly. Consequently, q has set Y to a value strictly greater than m at the start of round $r'+1$. In particular, this statement holds for $r' = r$, which contradicts the assumption that m was the maximum value among players in C_r. Now suppose that v was not p's input value, which means that p must have been a member of some committee $C_{r'}$ and adopted v as its value in some round $r' \leq r$. Since p did not crash in round $r'+1$ and recalling that $v > \delta$, applying Lemma 3 yields a contradiction to the assumption that some player decided on δ. ◀

▶ **Lemma 5.** *Algorithm 2 satisfies agreement.*

Proof. Consider the earliest round $r \in [1, f+1]$ in which no player crashes. First, assume that $r = 1$. Then, it follows from Lemma 3 that no player has an input greater than δ. Thus, it follows from validity (see Lem. 2) that δ was the maximum input value, and hence every player in C_1 adopts δ in round 1. It follows that all alive members of every committee will have value δ, and thus every player decides on δ eventually. On the other hand, if $1 < r < f+1$, then Lemma 3 and the fact that someone decides on δ tell us that every alive player in C_{r-1} must have sent a value of at most δ in round r. Moreover, Lemma 4 ensures that all alive players (whether they are sleeping or awake) have a value of at most δ at the start of round $r+1$; let V denote this set of values. If $\max V < \delta$, then Lemma 4 implies us that every alive player has a value strictly less than δ. Since values are updated using the maximum rule, this contradicts the assumption that some player decided on δ, and hence it follows that some alive player $p \in C_r$ has the value δ, which is the maximum among all committee members. And, applying Lemma 4 guarantees that no greater values are held by players not in C_r. Since p does not crash, a simple inductive argument shows that δ is the only possible decision value. ◀

It remains to prove the claimed bound on the energy and message complexity: The first and the last phases require every player to be awake for a constant number of rounds. The second phase requires the players in each committee to be awake only in the two rounds during which their respective committee needs to receive or send messages. Since each player belongs to at most $\lceil \frac{(f+1)\cdot f}{n} \rceil$ committees, the energy complexity is $\mathcal{O}(\lceil f^2/n \rceil)$. During Phases 1 and 3, the algorithm sends $\mathcal{O}(f \cdot n)$ messages. In each round r of Phase 2, all players in committee C_{r-1} send a message to every player in committee C_r, which amounts to $\mathcal{O}(f^2)$ messages for each of the $f-1$ rounds, and thus the total number of messages is $\mathcal{O}(f^3 + f \cdot n)$.

4 Binary Consensus

In this section, we show how to improve over the energy complexity bounds obtained in Section 3, by restricting the domain of the input values to a single bit, while keeping the optimal time complexity of $f+1$ rounds. Recall that, in Algorithm 2, the energy complexity was dominated by the number of committees that each player is a member of, which in turn depends on the size of the individual committees, i.e., $f+1$. Here, we explore the natural strategy of reducing the committee size to achieve a lower energy complexity. While conceptually simple, having smaller committees entails that we can no longer rely on each committee having at least one alive member, requiring a player to spread its value among multiple committees instead. Moreover, it might create an opening for the adversary to strategically crash players, one at each round, such that eventually a round is reached where all the members of a committee (or in general all the players that should receive messages) have already crashed. This allows the adversary to abstain from crashing during that particular round without any player receiving new information, which makes it impossible for the algorithm to terminate after $f+1$ rounds. We compensate for this lack of information spreading across committees by using players aware of decisive information (here, players holding 1) as propagating units for further committees. Assume players may choose 1 over 0 when they have the choice (i.e., they know at least 1 player holds each of those values as their initial value). This situation allows a player to know that everyone may decide 1 when he receives the value, and/or he will crash before the end of the procedure.

Algorithm 3 Binary Consensus. Code for player p, who has input value X.

```
   // Phase 1 (round 1)
1  player p wakes up in round 1:
2  h ← min{f, n' − √n' + 1}
   // h = first round of Phase 3 or 4,
   // depending on the relation between
   // f and n'
3  Locally compute first batch of committees for
     Phases 1 and 2:
     C_1, ··· , C_{h−1} ←JoinComm(h − 1, √n', n')
4  Locally compute second batch of committees:
     C_h, ··· , C_f ←JoinComm(f − h + 1, f + 1, n)
5  Z ← 0; Y ← 0; T ← 0
6  if X = 1 then
7      Y ← 1; T ← ⌈(f+1)/√n'⌉
8      Send 1 to every player in committee C_1
9  if p received value 1 and Y = 0 then
10     Y ← 1; T ← ⌈(f+1)/√n'⌉

   // Phase 2 (rounds 2, . . . , h − 1)
11 For each round r ∈ [2, h − 1] do:
12 if T > 0 then
13     player p wakes up in round r
14     Send 1 to every player in C_r
15     T ← T − 1
16 if p ∈ C_r then
17     player p wakes up in round r
18     if p received value 1 and Y = 0 then
19         Y ← 1; T ← ⌈(f+1)/√n'⌉

   // Phase 3 (rounds h, . . . , f − 1)
20 For each round r ∈ [n' − √n' + 1, f − 1] do:
   // r = first round of the phase
21 if Y = 1 and r = n' − √n' + 1 then
22     player p wakes up in round r
23     Send 1 to every player in C_r.
24 if T > 0 then
25     player p wakes up in round r
26     Send 1 to every player in C_r.
27     T ← T − 1
28 if p ∈ C_r then
29     player p wakes up in round r
30     if p received value 1 Z = 0 then
31         Z ← 1 ;T ← 1

   // Phase 4 (rounds f, f + 1)
   // round f
32 Player p always wakes up in round f
33 if Y = 1 or Z = 1 then
34     Send 1 to every player in C_f
35 if p ∈ C_f and p received value 1 then
36     Y ← 1
   // round f + 1
37 Player p always wakes up in round f+1
38 if p ∈ C_f and Y = 1 then
39     Send 1 to all
40 if received value 1 then
41     Decide 1 at the end of round f + 1
42 else
43     Decide 0 at the end of round f + 1
```

4.1 Description of the Algorithm

We now give a high-level overview of our algorithm; see Algorithm 3 for the detailed pseudo code.

The algorithm is divided into four phases, and employs committees of players, of size roughly $\sqrt{n}$. During the first phase, players holding 1 as their initial value send it to the first committee. During the second phase, players who received the value 1 become active for $\mathcal{O}(f/\sqrt{n})$ rounds, and propagate this value in each round to the currently awake committee. Depending on the relation between f and $\lfloor\sqrt{n}\rfloor^2 - \lfloor\sqrt{n}\rfloor + 1$, a third phase might happen. In this third phase, players behave the same as in phase 2, except that the committees are of size $f+1$ and the active players propagate the value 1 only if they received it during the last round. The fourth phase consists of two last rounds. In round f, every player aware of the value 1 sends it to a committee of size $f+1$, and in the last round, each member of that committee broadcasts this value to all players if it received it on the previous round. This strategy ensures that, assuming an adversary crashes at most f players, if a player decides on 1 in the last round, then there must exist a non-faulty player holding 1 in some earlier round $< f+1$. This strategy also ensures a worst case energy complexity of $O(\lceil f/\sqrt{n}\rceil)$. A more detailed description follows next.

During initialization, we use Procedure JoinComm (see Algorithm 1) to fix f different static committees $C_1, \ldots, C_f$. Let $n' = \lfloor\sqrt{n}\rfloor^2$ be the largest square number smaller or equal to n and let $h = \min\{n' - \sqrt{n'} + 1, f\}$. The size of committees depends on the relation

between f and n'. Specifically, we divide the committees into two parts: $C_1, \ldots, C_{h-1}$, where each committee is of size $\sqrt{n'}$, and $C_h, \ldots, C_f$, where each committee is of size $f+1$. Note that if $f \leq n' - \sqrt{n'} + 1$, then $h = f$ and $C_{h-1} = C_{f-1}$ (in other words, all the committees except for C_f are of size $\sqrt{n'}$). If $f > n' - \sqrt{n'} + 1$, then there could be more than one committee of size $f+1$. To simplify the notation, we also define $C_{f+1} = P$. A crucial property of being a member of committee C_r is that every $p \in C_r$ will wake up (at least) in round r, and thus we say that round r is the *awake round of committee* C_r. Each player has two local variables Y, Z that are initialized with 0, in which we store the player's current decision estimate (each variable is used in exactly one of two parts in the algorithm explained later). Since our algorithm restricts itself to sending messages with bit value 1, called 1-*valued message*, we only ever update Y and Z by setting $Y \leftarrow 1$ or $Z \leftarrow 1$, if they were 0 before receiving such a message.

Conceptually, the algorithm consists of four phases:

Phase 1 consists only of round 1, in which every player p wakes up and, if p has input value $X = 1$, then it sends a message with a single bit 1 to all members of committee C_1. Player p becomes *active*, in the sense that p remains awake for the next $T = \left\lceil \frac{f+1}{\sqrt{n'}} \right\rceil$ rounds, where T is a counter that is decremented by 1 in each subsequent round. Any member $q \in C_1$ who has $Y = 0$ and receives such a message becomes *active* as well.

In Phase 2 (rounds $2, \ldots, h-1$), we propagate these 1-valued messages across all other committees as follows. If a player $p' \in C_r$ receives the value 1 for the first time upon waking up in round r (i.e., p' never received 1 before nor had 1 as their initial value), then p' will become *active*, and remain awake for the next $\lceil \frac{f+1}{\sqrt{n'}} \rceil$ rounds, during which p' sends bit 1 to the next corresponding committees whose members are awake in the respective round. Note that the length of Phase 2 depends on h (or in other words the relation between f and n'). If f is relatively small ($f \leq n' - \sqrt{n'} + 1$), then $h = f$ and upon completing Phase 2 in round $h - 1 = f - 1$, the players proceed directly to Phase 4 (described below). Otherwise, the players proceed to Phase 3.

In Phase 3 (rounds $h, \ldots, f-1$), the size of the committees are $f+1$ and we propagate 1-valued message in a similar way to Algorithm 2. During round h (the first round of Phase 3), every player awakes and sends a 1-valued message to every player in C_h, if it has $Y = 1$. Every player $p \in C_h$ that receives a 1-valued message during round h will awake during round $h+1$ and send a 1-valued message to C_{h+1}. In subsequent rounds of Phase 3, every player $p \in C_r$ that receives a 1-valued message for the first time in Phase 3 (it may have received a 1-value message in previous phases) during round r will awake in round $r+1$ and send a 1-valued message to C_{r+1}.

In Phase 4 (rounds $f, f+1$), all players wake up. In round f, every player that has $Y = 1$ or $Z = 1$ sends a 1-valued message to every player in C_f. Every player in C_f that receives a 1-valued message sets $Y = 1$. In round $f+1$, a player p broadcast 1 if $p \in C_f$ and $Y = 1$. A player decides 1 if it receives a 1-valued message during round $f+1$ and decides 0 otherwise.

Figure 2 schematically illustrates the execution of Algorithm 3.

The main result of this section, proved hereafter, is the following.

▶ **Theorem 6.** *Suppose that each player starts with a one-bit input value and that at most $f < n$ players crash. There is a deterministic algorithm that solves consensus in $f+1$ rounds, sends $\mathcal{O}(nf)$ messages of constant size. Moreover, the algorithm achieves $\mathcal{O}(\lceil f/\sqrt{n} \rceil)$ energy complexity.*

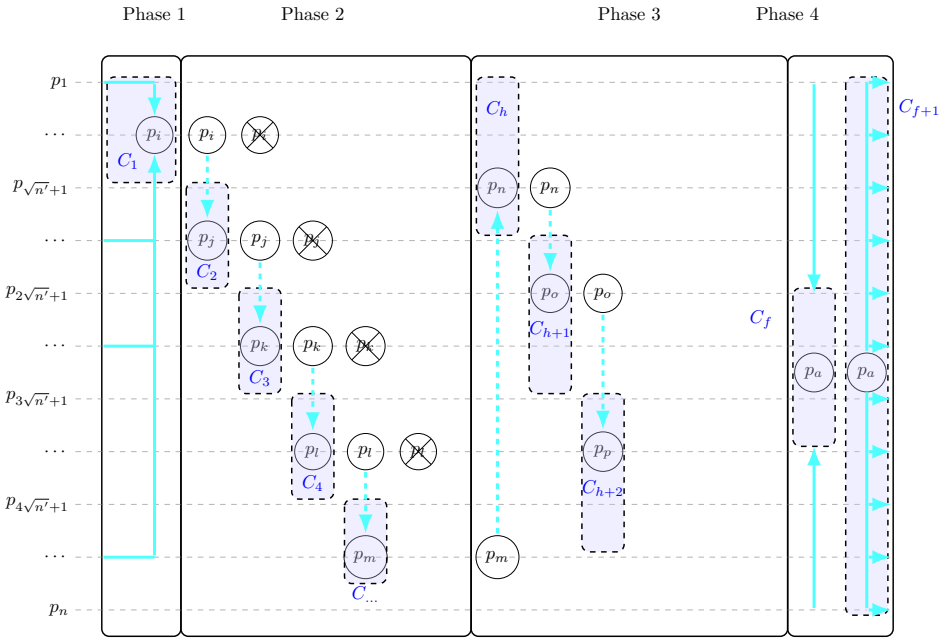

Figure 2 An Example of the Execution of Binary Consensus (Algorithm 3):
Phase 1: Every player with input value $X = 1$ sends a message to p_i, then crashes.
Phase 2: p_i is only able to send a message to $p_j \in C_2$ as it crashes. The same happens in each of the rounds of Phase 2, i.e., the player sending 1 can only send its message to one member of the next committee.
Phase 3: Assume that $f > n' - \sqrt{n'}$. All committees are now of size $f + 1$. Player p_m, who received the message in the last round of Phase 2, tries to send a message to everyone in C_h but crashes and the message is only received by p_n; in the remaining rounds of Phase 3, this process continues analogously.
Phase 4: Every alive player with $Y = 1$ try to send messages in C_f in round f, they may crash before fully propagating 1 to C_f. $p_a \in C_f$ receives a message. In round $f + 1$, p_a try to broadcast 1, and the adversary here can not crash p_a without exceeding the threshold f. Consensus is reached.

4.2 Notation and Terminology

Before proving Theorem 6, we need to define some crucial terms for the analysis of the algorithm. Recall that $h = \min\{n' - \sqrt{n'} + 1, f\}$. If $f \leq n' - \sqrt{n'}$, then Phase 3 is not executed, and $h = f$ is the first round of the last phase of the execution (denoted Phase 4 in the algorithm, for rounds $f, f + 1$). Otherwise, $h = n' - \sqrt{n'} + 1$ is the round that starts Phase 3.
Consider some round r. We say that a *player p is triggered in round r*, if p receives a 1-valued message in round r that causes it to set its timer (and start broadcasting 1 as long as its timer is not 0), or $r \in \{1, h, f + 1\}$ and p wakes up in r to send a 1-valued message. The next observation is immediate from the description of the algorithm.

▶ **Observation 7.** *Let m be a specific 1-valued message sent by p. There exists a round r such that one of the following is true:*

- *m is sent as a consequence of p being triggered in round r by receiving a 1-valued message from some player p', in some round following r, or*
- *$r \in \{1, h, f + 1\}$ and p wakes up during round r by the algorithm's description and sends m (since it has $Z = 1$, $Y = 1$ or $X = 1$, depending on the round).*

We refer to the round r guaranteed by Observation 7 as the *trigger round of* m. Note that if $r \in \{h, f+1\}$ and p wakes up by the algorithm's description during r (for having $Y = 1$), then p must have been triggered in some prior round $r' < r$, in which case the trigger round of m is r'. We say that p was *triggered by* p' to send m in the former case, as well as in the latter case if $r \in \{h, f+1\}$. We say that p was *triggered by the algorithm* before round 1 to send m in the latter case if $r = 1$. For convenience, we define $C_{f+1} = V$ (i.e., the set of all players).

▶ **Observation 8.** *The following hold:*

(1) *During each round $r \in [h+1, f+1]$, a player sends a 1-valued message only if it was triggered in the previous round, and that can happen at most once.*

(2) *A player can be triggered to send messages during rounds $[1, h]$ only once.*

Proof. Part (1) follows since a player can only be triggered if $Z = 0$, and when it is triggered it sets $Z = 1$. Part (2) holds since a player can only be triggered if $Y = 0$, upon which Y is set to 1. ◀

4.3 Proof of Theorem 6

To prove the correctness of the algorithm, we need to show that it satisfies the validity, termination and agreement requirements. To start the proof, we make the following key observation on the structure of committees. Note that this observation is the reason we use the first n' players for the committees of Phase 2 rather then all n players.

▶ **Observation 9.** *$C_i = C_{i+\sqrt{n'}}$ for every $i \in [1, h-1-\sqrt{n'}]$.*

Proof. This follows from the round robin construction of the committees in Algorithm 1 and the fact that $\sqrt{n'}$ is an integer; i.e, we can form $\sqrt{n'}$ committees of size $\sqrt{n'}$. Then, committee number $\sqrt{n'}+1$ will be the same as committee number 1 and so on. ◀

Next, we show that if enough players ($f+1$) receive a 1-valued message prior to round $f+1$, then agreement on 1 is guaranteed.

▷ Claim 10. If a non-faulty player receives a 1-valued message before round $f+1$, then all non-faulty players decide 1 at the end of the execution.

Proof. Let p be a non-faulty player that receives a 1-valued message for the first time during round $r \leq f$. If $r = f$, then $p \in C_r$ and during round f, p sets $Y \leftarrow 1$. Hence, in round $f+1$, p broadcasts 1 to all players, resulting in every non-faulty processor deciding 1. Otherwise ($r < f$), p awakes in round f and sends 1 to every player in C_f. Since $|C_f| = f+1$, at least one recipient is non-faulty. Following the same reasoning of the previous case, the claim follows. ◁

A direct implication of Claim 10 is the following corollary.

▶ **Corollary 11.** *If $f+1$ players receive a 1-valued message before round $f+1$, then all non-faulty players decide 1 at the end of the execution.*

Proof. At least one of those $f+1$ players is non-faulty, and thus every non-faulty player decides 1 by Claim 10. ◀

Since we defined a player to be non-faulty if it does not crash before deciding, it suffices to show validity, termination, and agreement for non-faulty players.

▶ **Lemma 12.** *Algorithm 2 satisfies validity and terminates in $f+1$ rounds.*

Proof. Every non-faulty player decides on Y in round $f+1$, hence termination is guaranteed in $f+1$ rounds.

Next, we argue validity: In the case where every player starts with 0, no message is sent throughout the entire execution, causing all players who decide to decide on their input value. Now consider the case where everyone starts with 1. Since $n \geq f+1$, then by Corollary 11, every non-faulty player decides 1, and validity follows. ◀

We now focus on the agreement property: The cases where every player starts with the same input value are already covered by Lemma 12, and henceforth we can assume that there are players starting with 0 as well as with 1.

We now prove the main correctness claim, namely, that the agreement requirement is satisfied by the algorithm. Assume towards contradiction that there exist an execution EX and two non-faulty players p^0, p^1 that decide $0, 1$ respectively in EX. By the algorithm's description, p^1 must have received a 1-valued message during round $f+1$ (not before, otherwise by Claim 10, p^0 would decide 1). Denote by $\hat{p}_\ell$ the player that sent p^1 the 1 valued message during round $f+1$ (if there are more than one, then choose the one that was triggered to do so first, and if there are still more than one, choose one arbitrarily). Note that $\hat{p}_\ell$ must have been triggered (to send that message) by some player $\hat{p}_{\ell-1}$ during some prior round $r_{\ell-1}$ (the same rules for picking $\hat{p}_{\ell-1}$ apply). In general, $\hat{p}_{j+1}$ is either triggered by $\hat{p}_j$ during round r_j or is triggered by the algorithm before round 1. We obtain a sequence of players $\hat{p}_1, \ldots, \hat{p}_{\ell+1}$, where $\hat{p}_1$ is some player that has input 1 (and is triggered by the algorithm before round 1), $\hat{p}_{\ell+1} = p^1$, and $r_\ell = f+1$. For ease of notation we will refer to $\hat{p}_j$ sending 1 to $\hat{p}_{j+1}$ as $\hat{p}_j$ *triggering* $\hat{p}_{j+1}$.

▶ **Observation 13.** *For every round $r \in [h+1, f+1]$, $r = r_j$ for some $j \in \ell$.*

Proof. We prove the observation inductively on the round number r.

Base ($r = f+1$). $r = f+1 = r_\ell$ by definition.

Step ($r < f+1$). By the induction hypothesis, we know that for $r+1 = r_j$ for some $j \leq \ell$. Recall that $\hat{p}_j$ is triggered by $\hat{p}_{j-1}$ during round $r_{j-1} < r_j = r+1$ ($r_{j-1} \leq r$) and that $\hat{p}_j$ triggers $\hat{p}_{j+1}$ during round $r_j = r+1$. If $r > r_{j-1}$, then $\hat{p}_j$ does not crash during round r and sends 1 to every player in C_r. By Corollary 11, every non-faulty player decides 1, since $|C_r| \geq f+1$. ◀

▶ **Observation 14.** *In execution EX, a player that is triggered in round $r < h$ cannot be triggered during round $r' \geq h$.*

Proof. Assume towards contradiction that p is triggered during both rounds $r < h$ and $r' \geq h$. Hence, p does not crash during round h, and since it was triggered during round $r < h$, it has $Y = 1$ during round h, which means it sends 1 to C_h. If $h = f+1$, then p^0 receives and decides 1; otherwise, by Corollary 11 ($|C_h| = f+1$), it follows again that p^0 decides 1, which is a contradiction. ◀

▷ Claim 15. Every player in the sequence $\hat{p}_1, \ldots, \hat{p}_\ell$ is unique.

Proof. Assume towards a contradiction that $\hat{p}_j = \hat{p}_{j'}$ for some indices $j < j'$. Consider the following cases. If $h \leq r_j$ (Phase 3), then $\hat{p}_j = \hat{p}_{j'}$ sends a message during round $r_{j'}$, which means that it did not crash during round r_j and sent 1 to every player in C_{r_j+1}. In Phase 3, the committee size is $|C_{r_j}| = f+1$. Therefore, by Corollary 11, all non-faulty players will decide 1, in contradiction.

If $r_j < h$ (Phase 1 or 2), then $\hat{p}_j$ is triggered during round r_{j-1} and $\hat{p}_{j'}$ is triggered during round $r_{j'-1}$. Note that $r_{j-1} < r_j \leq r_{j'-1}$. It cannot be that $r_{j'-1} \geq h$ because otherwise $\hat{p}_j = \hat{p}_{j'}$ is triggered in both $r_{j-1} < h$ and $r_{j'-1} \geq h$ violating Observation 14. Also, it cannot be that $r_{j'-1} < h$, because a player can be triggered only once during rounds $[1, h]$. Hence, $r_{j'-1}$ cannot exist. ◁

▷ Claim 16. Every player that sets its timer $T > 0$ during execution EX must have crashed before it sets $T \leftarrow 0$.

Proof. Let p be a player that sets its timer during round $r \leq f$. Consider two cases.

If $r \geq h$ and $h < f + 1$ (Phase 3), then p sets its timer to 1. In the following round, $r + 1$, p sends 1 to C_{r+1}. If $r + 1 \leq f$ then $|C_{r+1}| = f + 1$ and by Corollary 11, every non-faulty player decides 1. If $r + 1 = f + 1$, then $p \in C_f$, $|C_{r+1}| = n$ and every non-faulty player receives 1 and thus decides 1. In both cases all non-faulty players decide 1, contradiction.

If $r < h$ (Phase 1 or 2), then p sets its timer to $\lceil (f + 1)/\sqrt{n'} \rceil$. If p does not crash before its timer reaches 0, then it sends 1 to the committees awake during the $\lceil \frac{f+1}{\sqrt{n'}} \rceil$ rounds following r. Since the size of every such committee is at least $\sqrt{n'}$ and by Observation 9, p sends 1 to at least $\min\{f + 1, n'\}$ different players. If $f \leq n' - \sqrt{n}$ or p sends 1 during round h, then p sends 1 to at least $f + 1$ players, and by Corollary 11, every non-faulty player decides 1, contradiction.

The remaining, more difficult case is when $f > n' - \sqrt{n'}$ and p does not send 1 during round h. In this case, p sends 1 to all of the first $n' < f + 1$ players and $h = n' - \sqrt{n'} + 1$. All of those players must crash before round $h + 1$ (otherwise, every player in C_h receives 1 and Corollary 11 applies). Recall that for every round $r \in [h + 1, f + 1]$, $r = r_j$ for some $j \in [1, \ell]$ (Observation 13) and there is a unique player $\hat{p}_j$ that sends 1 during round r (by Claim 15). The player $\hat{p}_j$ cannot have been triggered during any round $r' < h$ by Observation 14, so $\hat{p}_j \notin \{p_1, \ldots, p_{n'}\}$ (as they were triggered during a round in $[1, h - 1]$). If $\hat{p}_j$ is non-faulty, then its timer reaches 0 and every non-faulty player decides 1 (as explained above). Hence, the set of players $\{p_1, p_2, \ldots, p_{n'}\} \cup \{\hat{p}_j \mid r_j \in [h + 1, f + 1]\}$ contains exactly $n' + f - h + 1 = f + \sqrt{n'} \geq f + 1$ unique players that must crash, contradiction. ◁

▶ **Observation 17.** *In every round r such that $r_j < r < r_{j+1}$ for some $j \in [1, \ell]$, every player in the committee C_r receives a 1-valued message.*

Proof. By definition, during round r_{j+1} the player $\hat{p}_{j+1}$ sends 1 to $\hat{p}_{j+2}$ and during round r_j, $\hat{p}_{j+1}$ is triggered by $\hat{p}_j$. Hence, in all rounds $r_j < r < r_{j+1}$, $\hat{p}_{j+1}$ does not crash (otherwise it will not be able to send during round r_{j+1}) and must have a non-zero timer, or in other words send a message, by Claim 16. Hence, every players in C_r receives 1 from $\hat{p}_{j+1}$. ◀

A *failure mapping* for execution EX is a mapping $\varphi : [0, f] \to V$ that maps every round $j \in [0, f]$ to a player $\varphi(j)$ such that

(P1) For $j \in [1, f]$, $\varphi(j) \in C_j$ and $\varphi(j)$ has crashed in EX.

(P2) $\varphi(0) = \hat{p}_1$.

(P3) φ is an injection, i.e., the players $\varphi(j)$ are distinct.

Note that the existence of a failure mapping for execution EX leads to contradiction, since the combination of these properties means that every round $j \in [0, f]$ is associated with a unique player that crashes in EX (otherwise, p^0 decides 1), which is $f + 1$ unique players, in contradiction to the number of faults being f.

We now describe a procedure Constr that, given an execution EX, attempts constructing a corresponding failure mapping φ for EX. Procedure Constr(EX) operates as follows:

1. Let $R = \{r_j \mid j \in [1, \ell - 1]\}$ and $P = \{\hat{p}_j \mid j \in [1, \ell]\}$.
2. $\varphi(0) = \hat{p}_1$ (satisfies property (P2)).
3. For $r_j \in R$, $\varphi(r_j) = \hat{p}_{j+1}$.
4. For $r \in [1, \ldots, f] \setminus R$, we define $\varphi(r) = p \in C_r$, where p is chosen such that there is no $r' < r$ with $\varphi(r') = p$.

Note that a-priori, Procedure Constr(EX) might fail to produce a failure mapping for EX. This might happen if at some round $r \notin R$, there is no player in C_r that is not already used in a previous round (since the players $\hat{p}_j$ for $j \in [1, \ell]$ are m unique players by claim 15). Note that $r < h$.

▷ Claim 18. If Procedure Constr(EX) is completed successfully, then Properties (P1), (P2), (P3) are satisfied.

Proof. Property (P1) is satisfied by the construction since for every round $r_j \in R$, by definition, $\hat{p}_{j+1} \in C_{r_j}$ receives a 1-valued message, and, for every round $r \in [1, f] \setminus R$, every player $p \in C_r$ receives a 1-valued message, by Observation 17.

Property (P2) is satisfied since $\varphi(0) = \hat{p}_1$ by construction.

Property (P3) is satisfied because of the following argument. Let $r < r'$ be two rounds. If $r' \in \{1, \ldots, f\} \setminus R$, then by construction $\varphi(r) \neq \varphi(r')$. Otherwise ($r' \in R$), if $r \in R$ then $\varphi(r) \neq \varphi(r')$ by Claim 15. If $r \in \{1, \ldots, f\} \setminus R$ and $r' = r_j$ for some $j \in [1, \ell - 1]$ then $\varphi(r') = p_{j+1}$. $p_{j+1} \notin C_r$ since (1) $r < h$ (2) $\hat{p}_{j+1}$ receives 1 and sets its timer during round r_j (3) if $r_j \geq h$, then during round h, $\hat{p}_{j+1}$ sends 1 to C_h in contradiction. Otherwise $r_j < h$, but a player can set its timer only once during round $[1, h - 1]$, a contradiction. Hence, $p_{j+1} \neq \varphi(r) \in C_r$. ◁

To complete the proof, we argue that this does not happen, i.e., the procedure always succeeds (hence leading to contradiction as discussed earlier).

▷ Claim 19. Procedure Constr(EX) is always successful.

Proof. Assume towards contradiction that the construction is unsuccessful. Hence, there exists some round $r \notin R$, such that $r \leq \min\{f, n' - \sqrt{n'}\} = h - 1$, where there is no player in C_r that is not already used in a previous round. Recall that $|C_r| = \sqrt{n'}$. For every $p \in C_r$ there exists $r' < r$ such that $\varphi(r') = p$. Let $p'_1, p'_2, \ldots p'_{\sqrt{n'}} \in C_r$ be the players in C_r in order of round labeling, and let $r(p'_j) < r$ be the round such that $\varphi(r(p'_j)) = p'_j$ for $j \in [1, \sqrt{n'}]$. It might be the case that $p'_1 = \hat{p}_1$ in which case $r(p'_1) = 0$. Hence, $r(p'_2) \geq 1$ and by Observation 9, $r(p'_{j+1}) \geq r(p'_j) + \sqrt{n'}$. Therefore, by induction we get that $r \geq r(p'_{\sqrt{n'}}) + \sqrt{n'} \geq 1 + (\sqrt{n'} - 1)\sqrt{n'} = 1 + n' - \sqrt{n'} > \min\{f, n' - \sqrt{n'}\} = h - 1$ in contradiction. ◁

The last claim implies that the existence of execution EX leads to the existence of a failure mapping for it, in contradiction with the assumption of at most f failures. This yields the following.

▶ **Lemma 20.** *Algorithm 3 (Binary Consensus) achieves agreement.*

▶ **Lemma 21.** *Algorithm 3 sends $\mathcal{O}(nf)$ messages of constant size and has an energy complexity of $\mathcal{O}(\lceil f/\sqrt{n} \rceil)$.*

Proof. We start by proving the message complexity. Note that if $f \leq \sqrt{n}$, then we can simply use Algorithm 2, which, in that case, already achieves awake complexity $O(1)$, round complexity $f + 1$ and message complexity $\mathcal{O}(fn)$; note that the players know f and $\sqrt{n}$ and hence can locally determine which algorithm to execute. Hence, in the remainder of the analysis, we can assume that $f > \sqrt{n}$.

(Phase 1): Every player with input value 1 may send a message to committee C_1. Therefore $\mathcal{O}(n \cdot \sqrt{n'}) = \mathcal{O}(nf)$ message might be sent since $|C_1| = \sqrt{n'}$.

(Phase 2): Every player p can be triggered only once during rounds $[1, h]$ (Observation 8), resulting in p sending messages to the current round's committee for $\lceil \frac{f+1}{\sqrt{n'}} \rceil$ rounds, and the size of a committee in phase 2 is $\sqrt{n'}$. Hence, the message complexity of this phase is $\mathcal{O}(nf + n\sqrt{n}) = \mathcal{O}(nf)$.

(Phase 3): In Phase 3, every player may send messages in round r to all players in C_{r+1} at most once. Since every committee in Phase 3 is of size $f + 1$, this accounts for at most $\mathcal{O}(nf)$ messages.

(Phase 4): In round f, every player may send a message to every player in C_f. In the final round, up to $f + 1$ players may broadcast, which again yields $\mathcal{O}(nf)$ messages.

Next, we analyze the energy complexity:

(Phase 1 and 4): The first and last phase consist of a constant number of rounds, which amounts to $\mathcal{O}(1)$ awake rounds per node.

(Phase 2): Recall that a player is awake in round r if its timer is not 0 or it belongs to C_r. During the rounds $1 \leq r < h$, the timer of a player p can be set to $\lceil (f + 1)/\sqrt{n'} \rceil = \mathcal{O}(f/\sqrt{n})$ only once. Therefore, every player is awake in at most

$$O\left(\frac{\min\{f, n' - \sqrt{n'}\} \cdot \sqrt{n'}}{n'} + \left\lceil \frac{f+1}{\sqrt{n'}} \right\rceil\right) = \mathcal{O}(f/\sqrt{n})$$

rounds.

(Phase 3): During rounds $r \geq h$, the timer of a player p can be set to 1 once, and it belongs to.

$$O\left(\frac{\max\{0, f - (n' - \sqrt{n'})\} \cdot (f+1)}{n}\right) = \mathcal{O}(f\sqrt{n}/n) = \mathcal{O}(f/\sqrt{n})$$

committees, where the first equality follows since

$$f - (n' - \sqrt{n'}) \leq n - \sqrt{n'}(\sqrt{n'} - 1) \leq n - (\sqrt{n} - 1)(\sqrt{n} - 2) = 3\sqrt{n} - 2$$

Hence, the awake complexity is $\mathcal{O}(\lceil f/\sqrt{n} \rceil)$. ◀

5 Conclusion

In this work, we have initiated the study of fault-tolerant and energy-efficient algorithms in synchronous networks, by showing that the $f + 1$ barrier on the time complexity of deterministic consensus algorithms does not apply to energy complexity. An intriguing question raised by our work is how close the energy complexity bound of $\Theta(f/\sqrt{n})$ is to the lower bound and, more specifically, whether the energy complexity must depend on the number of faults f. Note that it is possible to solve certain graph problems such as vertex coloring on a ring in just $O(1)$ energy [2], even though they are known to have a super-constant time complexity [19], which means that we cannot rule out an $O(1)$-energy algorithm for consensus.

References

1 Hagit Attiya and Jennifer Welch. *Distributed Computing: Fundamentals, Simulations and Advanced Topics (2nd ed.)*. John Wiley & Sons, Inc., April 2004.

2 Alkida Balliu, Pierre Fraigniaud, Dennis Olivetti, and Mikaël Rabie. Solving sequential greedy problems distributedly with sub-logarithmic energy cost. In Alkida Balliu and Fabian Kuhn, editors, *Proceedings of the ACM Symposium on Principles of Distributed Computing, PODC 2025, Hotel Las Brisas Huatulco, Huatulco, Mexico, June 16-20, 2025*, pages 417–427. ACM, 2025. doi:10.1145/3732772.3733499.

3 Leonid Barenboim and Tzalik Maimon. Deterministic logarithmic completeness in the distributed sleeping model. In Seth Gilbert, editor, *DISC 2021,*, volume 209 of *LIPIcs*, pages 10:1–10:19. LIPICS, 2021. doi:10.4230/LIPICS.DISC.2021.10.

4 Yi-Jun Chang, Varsha Dani, Thomas P. Hayes, Qizheng He, Wenzheng Li, and Seth Pettie. The energy complexity of broadcast. In Calvin Newport and Idit Keidar, editors, *PODC 2018*, pages 95–104. ACM, 2018. doi:10.1145/3212734.3212774.

5 Yi-Jun Chang, Varsha Dani, Thomas P. Hayes, and Seth Pettie. The energy complexity of BFS in radio networks. In Yuval Emek and Christian Cachin, editors, *PODC 2020*, pages 273–282. ACM, 2020. doi:10.1145/3382734.3405713.

6 Soumyottam Chatterjee, Robert Gmyr, and Gopal Pandurangan. Sleeping is efficient: Mis in o (1)-rounds node-averaged awake complexity. In *ACM PODC 2020*, pages 99–108, 2020. doi:10.1145/3382734.3405718.

7 Francesco D'Amato, Giuliano Losa, and Luca Zanolini. Asynchrony-resilient sleepy total-order broadcast protocols. In *Proceedings of the 43rd ACM Symposium on Principles of Distributed Computing*, pages 247–256, 2024. doi:10.1145/3662158.3662779.

8 Francesco D'Amato and Luca Zanolini. Recent latest message driven ghost: Balancing dynamic availability with asynchrony resilience. In *2024 IEEE 37th Computer Security Foundations Symposium (CSF)*, pages 127–142. IEEE, 2024. doi:10.1109/CSF61375.2024.00001.

9 Danny Dolev and H. Raymond Strong. Authenticated algorithms for byzantine agreement. *SIAM J. Comput.*, 12(4):656–666, 1983. doi:10.1137/0212045.

10 Fabien Dufoulon, William K. Moses Jr., and Gopal Pandurangan. Distributed MIS in o(log log n) awake complexity. In *PODC 2023, Orlando, FL, USA, June 19-23, 2023*, pages 135–145. ACM, 2023. doi:10.1145/3583668.3594574.

11 Francesco D'Amato, Roberto Saltini, Thanh-Hai Tran, and Luca Zanolini. Tob-svd: Total-order broadcast with single-vote decisions in the sleepy model. In *2025 IEEE 45th International Conference on Distributed Computing Systems (ICDCS)*, pages 1033–1043. IEEE, 2025. doi:10.1109/ICDCS63083.2025.00104.

12 Yuval Efron, Joachim Neu, and Toniann Pitassi. Fully-fluctuating participation in sleepy consensus. In *7th Conference on Advances in Financial Technologies (AFT 2025)*, pages 17–1. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2025. doi:10.4230/LIPIcs.AFT.2025.17.

13 Yuval Efron, Joachim Neu, Ling Ren, and Ertem Nusret Tas. Optimal good-case latency for sleepy consensus. *arXiv preprint arXiv:2510.06023*, 2025.

14 Michael J. Fischer, Nancy A. Lynch, and M. S. Paterson. Impossibility of distributed consensus with one faulty process. *Journal of the ACM*, 32(2):374–382, April 1985. doi:10.1145/3149.214121.

15 Mohsen Ghaffari and Julian Portmann. Distributed MIS with low energy and time complexities. In *PODC 2023, Orlando, FL, USA, June 19-23, 2023*, pages 146–156. ACM, 2023. doi:10.1145/3583668.3594587.

16 Mohsen Ghaffari and Anton Trygub. A near-optimal low-energy deterministic distributed SSSP with ramifications on congestion and APSP. In Ran Gelles, Dennis Olivetti, and Petr Kuznetsov, editors, *PODC 2024, Nantes, France, June 17-21, 2024*, pages 401–411. ACM, 2024. doi:10.1145/3662158.3662812.

17 Khalid Hourani, Gopal Pandurangan, and Peter Robinson. Awake-efficient distributed algorithms for maximal independent set. In *42nd IEEE International Conference on Distributed*

Computing Systems, ICDCS 2022, Bologna, Italy, July 10-13, 2022, pages 1338–1339. IEEE, 2022. doi:10.1109/ICDCS54860.2022.00153.

18 Leslie Lamport, Robert Shostak, and Marshall Pease. The byzantine generals problem. *ACM Transactions on Programming Languages and Systems*, 4(3):382–401, 1982. doi:10.1145/357172.357176.

19 Nathan Linial. Locality in distributed graph algorithms. *SIAM Journal on computing*, 21(1):193–201, 1992. doi:10.1137/0221015.

20 Giuliano Losa and Eli Gafni. Consensus in the unknown-participation message-adversary model. *arXiv preprint arXiv:2301.04817*, 2023. doi:10.48550/arXiv.2301.04817.

21 Nancy Lynch. *Distributed Algorithms*. Morgan Kaufman Publishers, Inc., San Francisco, USA, 1996.

22 Dahlia Malkhi, Atsuki Momose, and Ling Ren. Towards practical sleepy bft. In *Proceedings of the 2023 ACM SIGSAC Conference on Computer and Communications Security*, pages 490–503, 2023. doi:10.1145/3576915.3623073.

23 Hugo Mirault and Peter Robinson. Brief announcement: Towards energy-efficient distributed agreement. In *Proceedings of the ACM Symposium on Principles of Distributed Computing*, PODC '25, pages 428–431, New York, NY, USA, 2025. Association for Computing Machinery. doi:10.1145/3732772.3733554.

24 Atsuki Momose and Ling Ren. Constant latency in sleepy consensus. In *Proceedings of the 2022 ACM SIGSAC Conference on Computer and Communications Security*, pages 2295–2308, 2022. doi:10.1145/3548606.3559347.

25 Joachim Neu, Javier Nieto, and Ling Ren. On the limits of consensus under dynamic availability and reconfiguration. *arXiv preprint arXiv:2510.03625*, 2025.

26 Rafael Pass and Elaine Shi. The sleepy model of consensus. In *International conference on the theory and application of cryptology and information security*, pages 380–409. Springer, 2017. doi:10.1007/978-3-319-70697-9_14.

27 Marshall Pease, Robert Shostak, and Leslie Lamport. Reaching agreement in the presence of faults. *Journal of the ACM (JACM)*, 27(2):228–234, 1980. doi:10.1145/322186.322188.

28 Jie Xu, Cong Wang, and Xiaohua Jia. A survey of blockchain consensus protocols. *ACM Comput. Surv.*, 55(13s):278:1–278:35, 2023. doi:10.1145/3579845.