Synchronization Modulo k in Dynamic Networks

Louis Penet de Monterno[1(✉)], Bernadette Charron-Bost[2], and Stephan Merz[3]

[1] École polytechnique, IP Paris, 91128 Palaiseau, France
penetdemonterno@lix.polytechnique.fr
[2] DI ENS, CNRS, École Normale Supérieure, 45 rue d'Ulm, 75005 Paris, France
[3] Université de Lorraine, CNRS, Inria, LORIA, 54000 Nancy, France

Abstract. We define the mod k-synchronization problem as a weakening of the Firing Squad problem, where all nodes fire not at the same round, but at rounds that are all equal modulo k. We propose an algorithm that achieves mod k-synchronization in any dynamic network where there exist – possibly several – fixed spanning stars within each period of Δ consecutive rounds. In other words, we require that there always exists a temporal path of length at most Δ between some fixed node γ and every other node. As opposed to the perfect synchronization achieved in the Firing Squad problem, mod k-synchronization thus does not require any strong connectivity property in the network. In our algorithm, all the nodes "know" Δ, but they ignore what nodes are the centers of the spanning stars. We also prove that if the bound Δ for guaranteeing fixed spanning stars exists but is unknown to the agents, then mod k-synchronization is impossible.

All nodes in our algorithm fire in less that $6kn + 4k$ rounds after all nodes become active, but unfortunately uses unbounded counters. We then propose a refinement of this algorithm so that it becomes finite state while maintaining the same time complexity. The correctness of our first algorithm has been formally established in the proof assistant Isabelle.

1 Introduction

Distributed algorithms are often designed in a synchronous computing model, in which computation is divided into *communication-closed rounds*: any message sent at some round can be received only at that round. In this model, it is usually assumed that each run of an algorithm is started by all nodes simultaneously, i.e., at the same round, or even at round one. For instance, most synchronous consensus algorithms (e.g., [8,13,14]), as well as many distributed algorithms for dynamic networks (e.g., [10,11]) require synchronous starts.

This assumption makes the sequential composition of two distributed algorithms $A; B$ – in which each node starts executing B when it has completed the execution of A – quite problematic. Indeed, nodes start the algorithm B asynchronously when the algorithm A terminates asynchronously, and the properties of B are no more guaranteed in this context of asynchronous starts.

C. Johnen et al. (Eds.): SSS 2021, LNCS 13046, pp. 425–439, 2021.
https://doi.org/10.1007/978-3-030-91081-5_28

This leads to the problem of simulating synchronous starts, classically referred to as the *firing squad problem*: Each node is initially *passive* and then becomes *active* at an unpredictable round. The goal is to guarantee that the nodes, when all active, eventually synchronize by *firing* – i.e., entering a designated state for the first time – at the same round.

Unfortunately, the impossibility result in [4] demonstrates that the firing squad problem is not solvable without a strong connectivity property of the network, namely, there exists some positive integer Δ such that the communication graph within every period of Δ consecutive rounds is strongly connected and the bound on the delay Δ is "known"[1]. In many situations, this connectivity property is not guaranteed: as an example, in the dynamic graphs corresponding to the Heard-Of models for benign failures, a node that suffers permanent and complete send omissions is constantly a sink in the communication graph.

However, looking more closely at many distributed algorithms designed in the round-based model, we see that these algorithms actually do not require perfectly synchronous starts, and still work under the weaker condition that all the nodes start executing the algorithms in rounds with numbers that are equal modulo k, for some positive integer k. The corresponding synchronization problem, that we call mod k-*synchronization*, is formally specified as follows:

Termination. If all nodes are eventually active, then every node eventually fires.

mod k-simultaneity. If two nodes fire at round t and t', then $t' \equiv t \mod k$.

Indeed, let A be an algorithm organized into regular *phases* consisting of a fixed number k of consecutive rounds: the sending and transition functions of every node at round t are entirely determined by the value of t modulo k. Moreover, assume that A has been proved correct (with respect to some given specification) when all nodes start A synchronously (at round one), but with any dynamic graph in a family $\mathcal{G}$ that is stable under the addition of arbitrary finite prefixes. For instance, the *ThreePhaseCommit* algorithm for non-blocking atomic commitment [1], as well as the consensus algorithms in [9] or the *LastVoting* algorithm [6] – corresponding to the consensus core of *Paxos* – fulfill all the above requirements for phases of length $k = 3$ and $k = 4$, respectively, and the family $\mathcal{G}$ of dynamic graphs in which there exists an infinite number of "good" communication patterns (e.g., a sequence of $2k$ consecutive communication graphs in which a majority of nodes is heard by all nodes in each graph). The use of a mod k-synchronization algorithm prior to the algorithm A yields a new algorithm that executes exactly like A does, after a finite preliminary period during which every node becomes active and fires. The above property on the set of dynamic graphs $\mathcal{G}$ then guarantees this variant of A to be correct with asynchronous starts and dynamic graphs in $\mathcal{G}$.

Another typical example for which perfect synchronization can be weakened to synchronization modulo k is the development of the basic *rotating coordinator* strategy in the context of asynchronous starts. Roughly speaking, this

[1] in a sense that will be detailed in Sect. 3.5.

strategy consists in the following: if nodes have unique identifiers in $\{1, \ldots, n\}$, the coordinator at round t is the node whose identifier is t modulo n. For that, each node u maintains a local counter c_u whose current value is the number of rounds in which it has been active. At each round, the coordinator of u is the node with the identifier that is equal to the current value of c_u modulo n. Since there may be only one coordinator per round, such a selection rule requires synchronous starts. Clearly, with the use of a mod n-synchronization algorithm in a preliminary phase and a counter for each node that now counts the number of rounds elapsed since the node fired, the above scheme implements the rotating coordinator strategy from the first round where all nodes have fired.

A natural question is then whether synchronization modulo k may be achieved without strong connectivity. In this paper, we address this issue and show that this problem is solvable under the sole assumption of a fixed center γ with a fixed and "known" bound on the delay Δ, that is, every node receives a message from γ (possibly indirectly) in every period of Δ consecutive rounds. In fact, we exhibit an algorithm, denoted by $SynchMod_k$, that achieves synchronization modulo k in any dynamic graph with a fixed center and a delay at most equal to k. The case where $\Delta > k$ will be covered separately, in Sect. 3.5. Interestingly, our algorithm requires no node identifiers. In particular, nodes are not assumed to "know" what node is the center of the graph. Provided that the communication graph is centered with delay at most Δ, no other assumption is made on the dynamic graph.

The correctness proof of our algorithm relies on a series of preliminary lemmas that consider all the possible cases for the respective values of the variables in the algorithm. In order to increase our confidence in the correctness and remove any doubts on such combinatorial proofs, we have developed a formal proof of the correctness of our algorithm in the interactive theorem prover Isabelle/HOL [12].

2 Preliminaries

2.1 The Computational Model

We consider a networked system with a *fixed* set V of n nodes. We assume a round-based computational model in the spirit of the Heard-Of model [6], in which point-to-point communications are organized into *synchronized rounds*: each node sends messages to all nodes and receives messages sent by *some* of the nodes. Rounds are communication closed in the sense that no node receives messages in round t that are sent in a round different from t. The collection of communications (which nodes receive messages from which nodes) at each round t is modelled by a directed graph (digraph, for short) with a set of nodes equal to V. The digraph at round t is denoted by $\mathbb{G}(t) = (V, E_t)$, and is called the *communication graph at round* t. The set of u's incoming neighbors in the digraph $\mathbb{G}(t)$ is denoted by $In_u(t)$.

We assume a self-loop at each node in all these digraphs since every node can communicate with itself instantaneously. The sequence of such digraphs $\mathbb{G} = (\mathbb{G}(t))_{t \geq 1}$ is called a *dynamic graph* [3].

In round t ($t = 1, 2, \ldots$), each node u successively (a) broadcasts messages determined by its state at the beginning of round t, (b) receives *some* of the messages sent to it, and finally (c) performs an internal transition to a successor state. A *local algorithm* for a node is given by a *sending function* that determines the messages to be sent in step (a) and a *transition function* for state updates in step (c). An *algorithm* for the set of nodes V is a collection of local algorithms, one per node.

We also introduce the notion of *start schedules*, represented as collections $\mathbb{S} = (s_u)_{u \in V}$, where each s_u is a positive integer or is equal to ∞.

The execution of an algorithm A with the dynamic graph $\mathbb{G}$ and the start schedule $\mathbb{S}$ then proceeds as follows: Each node u is initially *passive*. If $s_u = \infty$, then the node u remains passive forever. Otherwise, s_u is a positive integer, and u becomes *active* at the beginning of round s_u, setting up its local variables. In round t ($t = 1, 2 \ldots$), a passive node sends only heartbeats, corresponding to *null* messages, and cannot change its state. An active node applies its sending function in A to its current state to generate the messages to be sent, then it receives the messages sent by its incoming neighbors in the directed graph $\mathbb{G}(t)$, and finally applies its transition function $\mathcal{T}_u$ in A to its current state and the list of messages it has just received (including the null messages from passive nodes), to compute its next state. Since each local algorithm is deterministic, an execution of the algorithm A is entirely determined by the initial state of the network, the dynamic graph $\mathbb{G}$, and the start schedule $\mathbb{S}$.

The states "passive" and "active" do not refer to any physical notion, and are relative to the algorithm under consideration: as an example, if two algorithms A and B are sequentially executed according to the order "A followed by B", then at some round, a node may be active w.r.t. A while it is passive w.r.t. B. In such a situation, the node is integrally part of the system and can send messages, but these messages are empty with respect to the semantics of the algorithm B.

2.2 Network Model and Start Model

Let us first recall the notion of *product* of two digraphs $G_1 = (V, E_1)$ and $G_2 = (V, G_2)$, denoted by $G_1 \circ G_2$ and defined as follows [5]: $G_1 \circ G_2$ has V as its set of nodes, and (u, v) is an edge if there exists $w \in V$ such that $(u, w) \in G_1$ and $(w, v) \in G_2$. For any dynamic graph $\mathbb{G}$ and any integer $t' > t \geq 1$, we let $\mathbb{G}(t : t') = \mathbb{G}(t) \circ \mathbb{G}(t+1) \circ \cdots \circ \mathbb{G}(t')$. By extension, we let $\mathbb{G}(t : t) = \mathbb{G}(t)$.

The set of incoming neighbors of u in $\mathbb{G}(t : t')$ is noted as $In_u(t : t')$. The set $In_u(t : t)$ is simply noted $In_u(t)$.

Each edge (u, v) in the digraph $\mathbb{G}(t : t')$ corresponds to a $u \triangleright v$ *path in the interval* $[t, t']$, i.e., a finite sequence of nodes $u = w_{t-1}, w_t, \ldots, w_{t'} = v$ such that each pair (w_i, w_{i+1}) is an edge of $\mathbb{G}(t + i)$. This path is said to be *active* if each node $w_{t-1}, w_t, \ldots, w_{t'}$ is active in rounds $t - 1, t, \ldots t'$, respectively.

A *network model* is any non-empty set of dynamic graphs. We will focus on those network models $\mathcal{G}^*_\Delta$ of dynamic graphs $\mathbb{G}$ where each digraph $\mathbb{G}(t : t+\Delta-1)$ contains a *fixed* star graph, namely,

$$\exists \gamma \in V, \forall t \in \mathbb{N}, \forall u \in V, \gamma \in In_u(t : t + \Delta - 1).$$

The dynamic graph $\mathbb{G}$ is said to be *centered at* the node γ with delay Δ, and γ is called a Δ-*center of* $\mathbb{G}$.

The network model $\mathcal{G}^*_\Delta$ contains some dynamic graphs which are partitionned during less than Δ consecutive rounds. If the network model containing dynamic graphs which are rooted in each round is denoted by $\mathcal{G}^{rooted}$, we can easily check that, because of self-loops, if a node γ is a root of each digraph $\mathbb{G}(t)$, then the dynamic graph $\mathbb{G}$ is centered at γ with delay $|V| - 1$. Then $\mathcal{G}^{rooted} \subseteq \mathcal{G}^*_{|V|-1}$. Similarly, if the network model containing dynamic graphs which are strongly connected in each round is denoted by $\mathcal{G}^{strong}$, we get $\mathcal{G}^{strong} \subseteq \mathcal{G}^{rooted} \subseteq \mathcal{G}^*_{|V|-1}$.

We also define a *start model* as a non-empty set of start schedules. A start schedule $\mathbb{S} = (s_u)_{u \in V}$ is *complete* if every s_u is finite, i.e., no node is passive forever. Synchronous starts correspond to complete start schedules where all s_u are finite and equal. The point of this paper is to simulate mod k-*synchronous starts* defined by $s_u \equiv s_v \mod k$ for every pair of nodes u and v, with any complete start schedule.

The algorithm we introduce in the next section requires the existence of a Δ-center. By comparison, the firing squad problem is solvable with some strong connectivity hypothesis. In other words, every node must be a Δ-center.

3 The Algorithm

In this section, we present simultaneously the pseudo-code of our algorithm, and its formal definition in the Isabelle framework. The correctness of the algorithm has been formally verified in Isabelle.[2] The proof that we present in this article closely follows our formal proof.

3.1 Pseudo-code and Formal Definition

The state of each node is represented by five variables whose initial value is given below.

```
record locState =
  x :: nat
  synch :: bool
  ready :: bool
  force :: nat      — force ∈ {0, 1, 2}
  level :: nat      — level ∈ {0, 1, 2}
definition initState where
  initState ≡ (| x = 0, synch = False, ready = False, force = 0, level = 0 |)
```

We define a datatype for messages sent between two nodes u and v: messages either carry a value of some type $'msg$, or are equal to *Null* if u is passive, or to *Void* if u is not an incoming neighbor of v.

```
datatype 'msg message = Content 'msg | Null | Void
```

[2] The complete Isabelle development is available at https://github.com/louisdm31/asynchronous_starts_HO_model/tree/master/proof/sync-mod.

Algorithm 1: The $SynchMod_k$ algorithm

Initialization:
  $c_u \in \mathbb{N}$, initially 0
  $synch_u \leftarrow false$
  $ready_u \leftarrow false$
  $force_u \in \{0, 1, 2\}$, initially 0
  $level_u \in \{0, 1, 2\}$, initially 0
At each round:
  send $\langle c_u, synch_u, force_u, ready_u \rangle$ to all
  receive incoming messages: let In^a be the set of nodes from which a non-null message is received.
  if *all received messages are non-null* **then**
    $synch_u \leftarrow \bigwedge_{v \in In^a} synch_v \wedge c_v \equiv c_u \mod k$
  end
  else
    $synch_u \leftarrow false$
  end
  $ready_u \leftarrow \bigwedge_{v \in In^a} ready_v$
  $force_u \leftarrow \max_{v \in In^a} force_v$
  $c_u \leftarrow 1 + \min_{\substack{v \in In^a \\ force_v = force_u}} c_v$
  if $c_u \equiv 0 \mod k$ **then**
    if $level_u = 0 \wedge synch_u$ **then**
      $level_u \leftarrow 1$
      if $force_u < 2$ **then**
        $force_u \leftarrow 1$
        $c_u \leftarrow 0$
      end
    end
    else if $level_u = 1 \wedge ready_u \wedge synch_u$ **then**
      $level_u \leftarrow 2$ `/* the node` u `fires */`
      $force_u \leftarrow 2$
      $c_u \leftarrow 0$
    end
    $synch_u \leftarrow true$
    $ready_u \leftarrow level_u > 0$
  end

3.2 Informal Description of the Algorithm

We fix some $k > 2$. In this algorithm, the nodes hold a *level* variable. When they become active, they move from passive state to level 0. They later move to level 1, then to level 2. Each time a node moves from some level to the next, this constitutes a *level-up event*. From now on, the level reached during this level-up event will be called the *strength* of this event. Reaching level 2 means firing. The

conditional statements at lines 20 and 27 of Algorithm 1 are executed when the node reaches level 1 and 2 respectively. The intuition of the algorithm can be summarized by two simple ideas.

Firstly, *each node keeps track of the most recent strongest level-up event.* Only the strongest level-up events are considered: if some node "knows" about a level-up event from level 1 to level 2, it will not record any level-up event from level 0 to level 1, nor any level-up event from passive state to level 0. Among the strongest level-up events, the nodes keep track of the age of the most recent one. For that purpose, they hold two variables c_u and $force_u$. At any round, node u knows that c_u rounds ago, some node reached a level equal to $force_u$ from the previous level (as proved in Lemma 6), and the node does not know any node which reached a level equal to $force_u$ (or higher) in any more recent round (as proved in Lemma 7). With lines 17 and 18, they update their c_u and $force_u$ variables using those of their incoming neighbors. The presence of self-loops implies that, in these lines, the minima and maxima are well-defined.

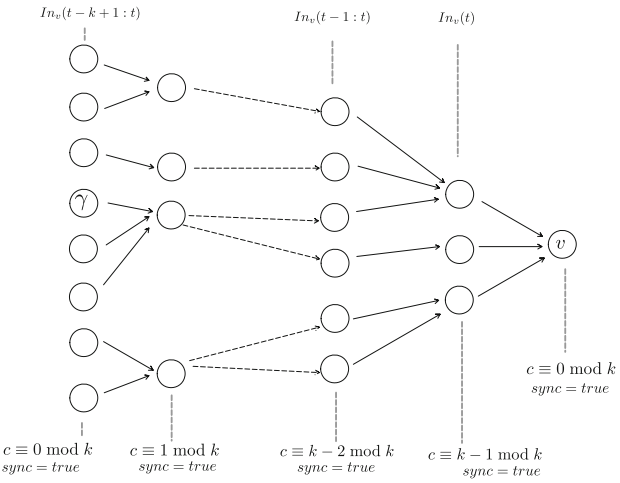

Fig. 1. Evolution of the incoming neighbors of u between round $t-k$ and t: case where every c_u is congruent to 0 in round $t-k$

Secondly, *a node may level up in round t only if its counter c_u is congruent to 0 and the counter of γ was also congruent to 0 k rounds ago.* Since the nodes do not "know" a fixed Δ-center, they conservatively level up only if all of their incoming neighbors $v \in In_u(t-k+1:t)$ were congruent to 0 k rounds ago. The assumption $\Delta \leq k$ guarantees that γ is one of these incoming neighbors. For that purpose, they use a Boolean variable *synch*. When the counter of some node v becomes congruent to 0 in some round $t-k$, it sets its $synch_v$ variable to *true* in line 32. During the next $k-1$ rounds, it will check whether the counters of its incoming neighbors are all congruent to its own counter (line 11). In case they are not, the node will set its $synch_u$ variable to *false*. This *false* value will

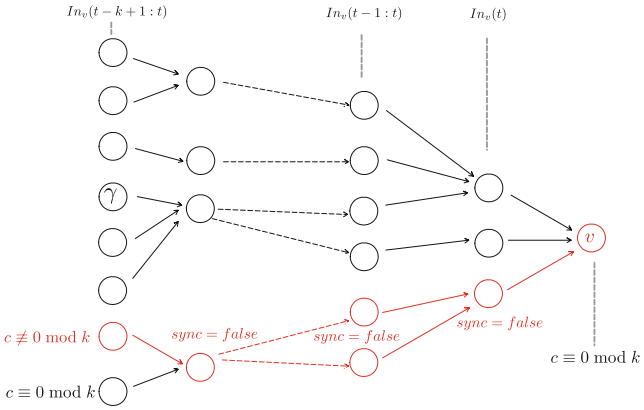

Fig. 2. Evolution of the incoming neighbors of u between round $t-k$ and t: case where some c_u are not congruent to 0 in round $t-k$

disseminate to its outgoing neighbors (also line 11). If, in round t, its $synch_u$ variable is still true, node u knows that no non-congruence was detected between round $t-k$ and round t. This means that every Δ-center was congruent with 0 in round $t-k$ (as proved in Lemma 3.c). In that case, a level-up event will take place (see Fig. 1). In contrast, if some node $v \in In_u(t-k+1:t)$ is not congruent with 0 in round $t-k$, then the line 11 guarantees that $synch_u$ will ultimately be *false* at the beginning of round t (see Fig. 2). In addition to *synch*, the *ready* variable makes sure that γ was already in level 1 k rounds ago (as proved in Lemma 4). Otherwise, the level-up event to level 2 is forbidden. Intuitively, the round t_γ in which γ reaches level 1 is used as a landmark for the mod k-synchronization: Lemma 9 shows that nodes fire in rounds which are congruent to t_γ modulo k.

3.3 Notation and Preliminary Lemmas

In the rest of this section, we fix an execution ρ of the $SynchMod_k$ algorithm for a complete activation schedule $\mathbb{S}$ and a Δ-centered dynamic graph $\mathbb{G} \in \mathcal{G}^*_\Delta$ with $\Delta \leq k$. Let $s^{\max} = \max_{u \in V} s(u)$ (note that $s^{\max} < \infty$) and let γ denote any Δ-center of $\mathbb{G}$.

If the node u is active in round t, the value of any u's variable x_u just before u executes line 19 at round t and at the very end of round t are denoted by $x_u^{pre}(t)$ and $x_u(t)$ respectively. By extension, $x_u(t)$ refers to the initial state if $t = s_u - 1$. In our formal proof, these values are encapsulated in a *rho* variable: for any round t, for any node u, *rho t u* returns either *Passive* or *Active s*, where *s :: locState* contains $c_u(t), synch_u(t) \ldots$. We now prove that this execution satisfies both properties of the mod k-synchronization problem.

definition *liveness* **where** — *termination*
liveness rho $\equiv \forall$ *u.* $\exists$ *t s. rho t u = Active s* $\wedge$ *level s = 2*
definition *safety* **where** — mod *k-simultaneity*
safety rho $\equiv \exists$ *c.* $\forall$ *u t s ss.*

rho t u = Active s ⟶ level s < 2 ⟶
rho (Suc t) u = Active ss ⟶ level ss = 2 ⟶ t mod k = c

We proved these propositions under the following assumptions:

assumes ∀ *u t. path In gamma u t k* — *gamma is a k-center*
and ∀ *u t. u ∈ In t u* — *the graph contains self-loops*
and *HORun (HOMachine k) rho In* — *rho is an execution*
and ∀ *p.* ∃ *t. rho t p ≠ Asleep* — *the schedule is complete*
and $k > 2$

The *HORun* term above is defined in [7] and characterizes executions of an algorithm. Since this definition was first written for synchronous starts, we adapted it to describe asynchronous starts.

We denote $In_u^a(t)$ the subset of nodes in $In_u(t)$ which are active in round $t-1$ in this execution. Some simple claims follow immediately from the definition of the transition function, regardless of the connectivity properties of $\mathbb{G}$. We consider some node $u \in V$ and some round t in which u is active (i.e., $t \geq s_u$).

Lemma 1.

(a) $level_u(t+1) \in \{level_u(t), level_u(t)+1\}$
(b) If $c_u(t) \neq 0$, *then* $force_u(t) = force_u^{pre}(t)$ *and* $c_u(t) = c_u^{pre}(t)$.
(c) $c_u(t) \equiv c_u^{pre}(t) \mod k$.
(d) If $synch_u^{pre}(t) = true$ *holds, then each node* $v \in In_u(t)$ *is active at round* $t-1$ *with:* $c_v^{pre}(t-1)+1 \equiv c_u^{pre}(t) \mod k$.
(e) If $c_u^{pre}(t) \not\equiv 1 \mod k$ *and* $synch_u^{pre}(t)$ *holds, then each node* $v \in In_u(t)$ *is active in round* $t-1$ *with* $synch_v^{pre}(t-1)$.
(f) If $c_u^{pre}(t) \not\equiv 1 \mod k$ *and* $synch_u^{pre}(t) = ready_u^{pre}(t) = true$, *then for every node* $v \in In_u^a(t)$, *it holds that* $ready_v^{pre}(t-1) = true$.
(g) For every $v \in In_u^a(t)$, *we have:*
$$force_v^{pre}(t-1) \leq force_v(t-1) \leq force_u^{pre}(t) \leq force_u(t).$$
(h) $\forall v \in In_u^a(t)$, $force_v^{pre}(t-1) = force_u^{pre}(t) \Rightarrow$
$$c_u^{pre}(t) \leq 1 + c_v(t-1) \leq 1 + c_v^{pre}(t-1).$$
(i) $level_u(t) \leq force_u(t)$.

Lemma 2. *No node can perform a level-up event action in round* $k-1$ *or earlier.*

We now show a few properties on the incoming neighbors of nodes that reach level 1 or 2. This situation is illustrated in Fig. 1.

Lemma 3. *Let* i *be an integer,* $0 \leq i < k$, *and let* u *and* v *be two nodes such that* $u \in In_v(t-k+i+1:t)$. *If* v *is active in round* t, *if* $c_v^{pre}(t) \equiv 0 \mod k$ *and* $synch_v^{pre}(t) = true$ *hold, then*

(a) $t \geq k$.
(b) u *is active in round* $t-k+i$.
(c) $c_u^{pre}(t-k+i) \equiv i \mod k$.
(d) If $ready_v^{pre}(t)$ *is true and* $i > 0$, *then* $ready_u^{pre}(t-k+i)$ *is true as well.*

Lemma 4. *If some node* u *reaches level 2 in round* t_u, *then* γ *is already in level 1 in round* t_u.

Lemma 5. *If γ reaches level 1 in round t_γ, no node can reach level 1 or 2 in any of the rounds $t_\gamma + 1, \ldots, t_\gamma + k - 1$.*

Lemma 6. *Let u be some node, and t be some round in which u is active. There exists some node w which reached a level equal to $force_u^{pre}(t)$ in round $t - c_u^{pre}(t)$. Moreover, an active $w \triangleright u$ path exists in the interval $[t - c_u^{pre}(t) + 1, t]$.*

We consider the set $Z = \{(f, t), \exists u \in V, level_u(t) = f \wedge level_u(t-1) \neq f\}$. This set is the finite set of level-up events. Using Lemma 6, any node u satisfies $z_u(t) = (force_u^{pre}(t), t - c_u^{pre}(t)) \in Z$ in every round $t \geq s_u$ in which u is active. We order Z lexicographically. The following two lemmas prove that $z_u(t)$ is the most recent strongest level-up event "known" by u in round t.

Lemma 7. *For every node u and v, if u leveled up in round t, then for every $i > 0$ such that there exists an active $u \triangleright v$ path in the interval $[t+1, t+i]$,*

$$\begin{aligned} &level_u(t) \leq force_v^{pre}(t+i) \\ \wedge\ &level_u(t) = force_v^{pre}(t+i) \Rightarrow c_v^{pre}(t+i) \leq i. \end{aligned}$$

Lemma 8. *If there exists an active $u \triangleright v$ path between two nodes u and v in the interval $[t+1, t']$, then $z_u(t) \leq z_v(t')$.*

Lemma 9. *If γ reached level 1 in some round t_γ, whereas some u reaches level 1 or 2 in some round $t_u \geq t_\gamma$, then $t_u \equiv t_\gamma \mod k$.*

Proof. By contradiction, we consider the earliest node u which levels up in some round $t_u \geq t_\gamma$ with $t_u \not\equiv t_\gamma \mod k$. By Lemma 2, $t_\gamma \geq k$. The Lemma 6 implies the existence of a node v which reached a level equal to $force_u^{pre}(t_u)$ in some round $t_v = t_u - c_u^{pre}(t_u)$.

In the case $force_u^{pre}(t_u) = 2$, from Lemma 4, we obtain $t_v \geq t_\gamma$.

In the case $force_u^{pre}(t_u) = 1$, Lemma 5 tells us that $t_u - t_\gamma \geq k$. Using self-loops and $\mathbb{G} \in \mathcal{G}_\Delta^*$ respectively, there exists a $\gamma \triangleright \gamma$ path in the interval $[t_\gamma + 1, t_u - k]$ and a $\gamma \triangleright u$ path in the interval $[t_u - k + 1, t_u]$. By concatenation, we obtain a $\gamma \triangleright u$ path in the interval $[t_\gamma + 1, t_u]$ Using Lemma 3.b, this path is active. From Lemma 7, $c_u^{pre}(t_u) \leq t_u - t_\gamma$. We also get $t_v \geq t_\gamma$.

The case $force_u^{pre}(t_u) = 0$ is impossible: we have $force_\gamma(t) \geq level_\gamma(t) \geq 1$ by Lemma 1.i. Using Lemma 1.g, we get $1 \leq force_\gamma(t) \leq force_{w_{t+1}}^{pre}(t+1) \leq \cdots \leq force_u^{pre}(t_u)$, where $w_t, w_{t+1}, \ldots, w_{t_u}$ is the $\gamma \triangleright u$ path constructed above.

In both possible cases, we have $t_v \geq t_\gamma$. By line 19, we have $c_u^{pre}(t) \equiv 0 \mod k$. Recalling $t_v = t_u - c_u^{pre}(t_u)$, we obtain $t_v \equiv t_u \not\equiv t_\gamma \mod k$. This contradicts the fact that u was the earliest such node. □

We say that the system is *monovalent* in round t if every node u is active and the values in the family $(c_u^{pre}(t))_{u \in V}$ are mutually congruent modulo k. Moreover, we denote $\bar{c}^{pre}(t)$ some integer which is congruent to every value $(c_u^{pre}(t))_{u \in V}$.

Lemma 10. *If the system is monovalent in round t, it is monovalent in any round $t + i$. Moreover, $\bar{c}^{pre}(t+i) \equiv \bar{c}^{pre}(t) + i \mod k$.*

Lemma 11. *If, in some round t, the system is monovalent, then every node u is in level 1 in round $t + 2k$ and in level 2 in round $t + 3k$.*

3.4 Correctness Proof

Lemma 12. *Under the assumption of a Δ-centered dynamic graph with $\Delta \leq k$, any execution of the SynchMod$_k$ algorithm satisfies the* mod k*-simultaneity property.*

Proof. We fix some node u, and we assume that u reaches level 2 in round t_u. From Lemma 4, we obtain $t_u \geq t_\gamma$, where t_γ is the round in which γ reaches level 1. By Lemma 9, $t_u \equiv t_\gamma \mod k$. That proves the mod k-simultaneity property. □

Lemma 13. *Under the assumptions of a complete activation schedule and of a Δ-centered dynamic graph with $\Delta \leq k$, any execution of the SynchMod$_k$ algorithm terminates.*

Proof. For every node u, the sequence $(z_u(t))_{t \geq s_u}$ belongs to the finite set Z. Moreover, by Lemma 8, this sequence is non-decreasing. Then it eventually stabilizes to some value z_u^{max}. Let z^{min} be $min\{z_u^{max}, u \in V\}$. We consider the round t^0 in which every node is active, and every sequence $(z_u(t))_{t \geq s_u}$ has stabilized to z_u^{max}. We consider the subset $V_{min} = \{u \in V, z_u^{max} = z^{min}\}$. We claim that $\forall t > t^0, \forall u \in V_{min}, In_u(t) \subseteq V_{min}$:

By contradiction, if in some round $t > t^0$, some $w \notin V_{min}$ belongs to $In_u(t)$, we would obtain $z_u^{max} = z^{min} < z_w(t-1) \leq z_u(t)$, using $u \in V_{min}$, $w \notin V_{min}$ and Lemma 8.

We apply Lemma 11 to the subsystem consisting of V_{min}. Since for all $t > t^0$ and $u \in V_{min}$, $In_u(t) \subseteq V_{min}$, this subsystem behaves like an independent system. Then, in round $t^0 + 3k$, every node in V_{min} is in level 2. By Lemma 1.i, every node $u \in V_{min}$ satisfies $force_u^{pre}(t^0 + 3k) = 2$. By definition of V_{min}, every node $u \in V$ has $force_u^{pre}(t^0 + 3k) = 2$. Now, we prove that in round $t^0 + 3k$, the entire system is monovalent:

Let us consider two nodes u_1 and u_2. By Lemma 6, we obtain two nodes w_1 and w_2 which reached level 2 in round $t^0 + 3k - c_{u_1}^{pre}(t^0 + 3k)$ and $t^0 + 3k - c_{u_2}^{pre}(t^0 + 3k)$ respectively. By Lemma 12, we obtain $c_{u_1}^{pre}(t^0 + 3k) \equiv c_{u_2}^{pre}(t^0 + 3k) \mod k$. That proves monovalence.

The termination property now follows from Lemma 11. □

The previous two lemmas yield the following correctness theorem:

Theorem 1. *Under the assumption of a Δ-centered dynamic graph with $\Delta \leq k$, and a complete activation schedule, the SynchMod$_k$ algorithm solves the* mod k*-synchronization problem for any integer k greater than 2.*

3.5 Solvability Results

We show that the mod k-synchronization problem is always solvable, regardless of the value of k, if the bound Δ on the delay is known: for each possible Δ, we can exhibit an algorithm which solves mod k-synchronization in any Δ-centeed dynamic graph.

Corollary 1. *For any positive integer k, the* mod k*-synchronization problem is solvable in each network model $\mathcal{G}^*_\Delta$ in any complete activation schedule.*

Proof. Depending on the relative values of k and Δ, we consider the following cases:

1. $k = 1$. The problem is trivially solvable in any network model, in particular $\mathcal{G}^*_\Delta$.
2. $\Delta \leq k$ and $k > 2$. By Theorem 1, the $SynchMod_k$ algorithm solves the mod k-synchronization problem in $\mathcal{G}^*_\Delta$ if $k > 2$.
3. $\Delta \leq k = 2$. Theorem 1 shows that the $SynchMod_4$ algorithm achieves mod 4-synchronization in $\mathcal{G}^*_2$, and hence achieves mod 2-synchronization in $\mathcal{G}^*_2$.
4. $\Delta > k$. We have $\Delta \leq \lceil \frac{\Delta}{k} \rceil \cdot k$. By Theorem 1, the mod $\lceil \frac{\Delta}{k} \rceil \cdot k$-synchronization problem is solvable in $\mathcal{G}^*_\Delta$ using $SynchMod_{\lceil \frac{\Delta}{k} \rceil \cdot k}$. The mod k-synchronization problem is also solvable in $\mathcal{G}^*_\Delta$, *a fortiori*. □

In contrast, we show that the mod k-synchronization problem is not solvable if the delay Δ is unknown to the nodes.

Theorem 2. *If $k > 1$, then the* mod k*-synchronization problem is not solvable in the network model $\bigcup_{i \in \mathbb{N}} \mathcal{G}^*_i$.*

Proof. By contradiction, assume that an algorithm A solves the problem in the above-mentioned network model. We consider any system and we fix two nodes u and v in this system. We denote I the digraph only containing self-loops. We denote C_u and C_v the digraphs only containing self-loops and a star centered in u and v respectively. We construct four executions of A:

1. Every node starts in round 1. The dynamic graph is equal to C_u at each round. This dynamic graph belongs to $\mathcal{G}^*_1$. Using the termination of A, u fires in some round f_u.
2. Every node starts in round 1. The dynamic graph is equal to C_v at each round. This dynamic graph belongs to $\mathcal{G}^*_1$. Using the termination of A, v fires in some round f_v.
3. Every node starts in round 1. During the first $f_u + f_v$ rounds, the communication graph is equal to I. In every subsequent round, the communication graph is equal to C_u. This dynamic graph belongs to $\mathcal{G}^*_{1+f_u+f_v}$.
4. the node u starts in round 1, whereas every other node starts in round 2. During the first $f_u + f_v$ rounds, the communication graph is equal to I. In every subsequent round, the communication graph is equal to C_u. This dynamic graph belongs to $\mathcal{G}^*_{1+f_u+f_v}$.

From the point of view of u, the third execution is indistinguishable from the first execution. Then u fires in round f_u in the third execution. From the point of view of v, the third execution is indistinguishable from the second execution during the first f_v rounds. Then v fires in round f_v in the third execution. Using the mod k-simultaneity of A in the third execution, we obtain:

$$f_u \equiv f_v \bmod k.$$

Similarly, u fires in round f_u and v fires in round $1+f_v$ in the forth execution. Using the mod k-simultaneity of A in the forth execution, we obtain:

$$f_u \equiv f_v + 1 \bmod k.$$

We obtain a contradiction if $k > 1$. □

4 Complexity Analysis

4.1 Time Complexity Analysis

Theorem 3. *There are at most $6kn + 4k$ rounds between the activation of all nodes and the firing of all nodes.*

Proof. We now bound the number of rounds between the activation of all nodes (noted s^{max}) and the firing of all nodes. Let t_γ be the round in which γ reaches level 1. First, we try to bound $t_\gamma - s^{max}$. We consider the non-decreasing series $(z_\gamma(t))_{t \geq s_\gamma}$. By Lemma 4, no node can reach level 2 before round t_γ. Then, for any $t \in \{s_\gamma, \ldots, t_\gamma\}$, we have $z_\gamma(t) \in Z^- = \{(f,t) \in Z, f < 2\}$. This set $Z^- \subseteq Z$ is the set of level-up events of strength 0 or 1. Since nodes can reach level 0 and 1 only once, the cardinality of Z^- is bounded by $2n$, where n is the total number of nodes. We can show that γ is in level 1 in round t if $(z_\gamma(t))_{t \in \mathbb{N}}$ remains stable between rounds $t - 3k$ and t. Then the worst case scenario happens if $z_\gamma(s_\gamma)$ starts with the lowest value of Z^-, and every $3k$ rounds, $z_\gamma(t)$ moves to the closest greater element of Z^-. Then $t_\gamma - s^{max}$ is bounded by $2n \times 3k = 6kn$.

Second, if γ is in level 1 in round t_γ, then every node u satisfies $z_u(t_\gamma + k) \geq z_\gamma(t_\gamma)$. By Lemma 9, the system is monovalent in round $t_\gamma + k$. By Lemma 11, every node is in level 2 in round $t_\gamma + 4k$. We finally obtain that there is at most $6kn + 4k$ rounds between the activation of all nodes and the firing of all nodes. □

4.2 Reducing Memory Usage

For all nodes u, for all rounds t, we have $(force_u^{pre}(t), t - c_u^{pre}(t)) \in Z$ by Lemma 6. Since Z is finite, $c_u^{pre}(t)$ tends to infinity as t tends to infinity. We present below a idea (inspired by [2]) which can alleviate this issue: in each execution of Algorithm 1, total memory usage increases forever, whereas in each execution of Algorithm 2, total memory usage grows during some arbitraryly-long initial period, and then drops and remains bounded forever. The idea is as follows:

As soon as $force_u(t) = 2$, the node u "knows" that some node v fired in round $t - c_u(t)$ (see Lemma 6). Then u may fire in any round $t' \equiv t - c_u(t) \bmod k$. At this point, the transition function can thus be simplified as in Algorithm 2. This simplified version uses a constant amount of memory.

Algorithm 2: The $OptSynchMod_k$ algorithm

```
1  Initialization:
2      initialize with SynchMod_k's initial state
3  At each round:
4      if force_u = 2 then
5          send ⟨c_u, true, 2, true⟩ to all
6          c_u ← 1 + c_u mod k
7          if level_u < 2 ∧ c_u = 0 then
8              level_u ← 2
9          end
10     end
11     else
12         apply SynchMod_k's transition function
13     end
```

Theorem 4. *Under the assumption of a Δ-centered dynamic graph with $\Delta \leq k$ and a complete activation schedule the Algorithm 2 solves the* mod k*-synchronization problem. Moreover, in each execution of Algorithm 2, the memory usage of each node is finite.*

5 Conclusion and Future Work

In this paper, we presented the mod k-synchronization problem, and we introduced an algorithm solving this problem. We provided an optimized version of this algorithm to tackle large memory usage. We also provided an upper-bound on the number of rounds between the start of all nodes and the firing of all nodes. This bound is linear in both k and n, which is not bad. However, this bound is deteriorated by a few nasty worst-case scenarios. We believe that some additional assumptions could provide a much tighter bound, which would not depend on n. That would be especially useful in very large systems. This consitutes a possible topic for a future work.

References

1. Bernstein, P.A., Hadzilacos, V., Goodman, N.: Concurrency Control and Recovery in Database Systems. Addison-Wesley, Boston (1987)
2. Boldi, P., Vigna, S.: Universal dynamic synchronous self-stabilization. Distrib. Comput. **15**(3), 137–153 (2002)
3. Casteigts, A., Flocchini, P., Quattrociocchi, W., Santoro, N.: Time-varying graphs and dynamic networks. In: Frey, H., Li, X., Ruehrup, S. (eds.) ADHOC-NOW 2011. LNCS, vol. 6811, pp. 346–359. Springer, Heidelberg (2011). https://doi.org/10.1007/978-3-642-22450-8_27
4. Charron-Bost, B., Moran, S.: The firing squad problem revisited. In: 35th Symposium on Theoretical Aspects of Computer Science (STACS 2018). Leibniz International Proceedings in Informatics (LIPIcs), vol. 96, pp. 20:1–20:14 (2018)

5. Charron-Bost, B., Moran, S.: Minmax algorithms for stabilizing consensus. CoRR abs/1906.09073 (2019). http://arxiv.org/abs/1906.09073
6. Charron-Bost, B., Schiper, A.: The heard-of model: computing in distributed systems with benign faults. Distrib. Comput. **22**(1), 49–71 (2009)
7. Debrat, H., Merz, S.: Verifying fault-tolerant distributed algorithms in the heard-of model. Archive of Formal Proofs (2012). http://isa-afp.org/entries/Heard_Of.html. Formal proof development
8. Dolev, D., Strong, H.R.: Authenticated algorithms for Byzantine agreement. SIAM J. Comput. **12**(4), 656–666 (1983)
9. Dwork, C., Lynch, N.A., Stockmeyer, L.: Consensus in the presence of partial synchrony. J. ACM **35**(2), 288–323 (1988)
10. Kuhn, F., Lynch, N., Oshman, R.: Distributed computation in dynamic networks. In: Proceedings of 42nd ACM Symposium on Theory of Computing (STOC 2010), pp. 513–522. ACM, New York (2010). https://doi.org/10.1145/1806689.1806760
11. Kuhn, F., Moses, Y., Oshman, R.: Coordinated consensus in dynamic networks. In: Proceedings of 30th ACM Symposium on Principles of Distributed Computing (PODC). ACM (2011)
12. Nipkow, T., Paulson, L., Wenzel, M.: Isabelle/HOL. A Proof Assistant for Higher-Order Logic. Lecture Notes in Computer Science, vol. 2283. Springer, Heidelberg (2002). https://doi.org/10.1007/3-540-45949-9
13. Pease, M., Shostak, R., Lamport, L.: Reaching agreement in the presence of faults. J. ACM **27**(2), 228–234 (1980)
14. Srikanth, T.K., Toueg, S.: Simulating authenticated broadcasts to derive simple fault-tolerant algorithms. Distrib. Comput. **2**(2), 80–94 (1987)