Self-Stabilizing Clock Synchronization in Dynamic Networks

Bernadette Charron-Bost, Louis Penet de Monterno

Self-Stabilizing Clock Synchronization in Dynamic Networks

Bernadette Charron-Bost
DI ENS, École Normale Supérieure, 75005 Paris, France

Louis Penet de Monterno
École polytechnique, IP Paris, 91128 Palaiseau, France

Abstract

We consider the fundamental problem of periodic clock synchronization in a synchronous multi-agent system. Each agent holds a clock with an arbitrary initial value, and clocks must eventually be congruent, modulo some positive integer P. Previous algorithms worked in static networks with drastic connectivity properties and assumed that global informations are available at each node. In this paper, we propose a finite-state algorithm for time-varying topologies that does not require any global knowledge on the network. The only assumption is the existence of some integer D such that any two nodes can communicate in each sequence of D consecutive rounds, which extends the notion of strong connectivity in static network to dynamic communication patterns. The smallest such D is called the *dynamic diameter* of the network. If an upper bound on the diameter is provided, then our algorithm achieves synchronization within $3D$ rounds, whatever the value of the upper bound. Otherwise, using an adaptive mechanism, synchronization is achieved with little performance overhead. Our algorithm is parameterized by a function g, which can be tuned to favor either time or space complexity. Then, we explore a further relaxation of the connectivity requirement: our algorithm still works if there exists a positive integer R such that the network is rooted over each sequence of R consecutive rounds, and if eventually the set of roots is stable. In particular, it works in any rooted static network.

2012 ACM Subject Classification Theory of computation → Distributed algorithms; Theory of computation → Dynamic graph algorithms

Keywords and phrases Self-stabilization, Clock synchronization, Dynamic networks

Digital Object Identifier 10.4230/LIPIcs.OPODIS.2022.28

Supplementary Material *Software (Source Code)*: `https://gitlab.com/bossuet/sap_execution`

Acknowledgements We would like to thank Patrick Lambein-Monette, Stephan Merz, and Guillaume Prémel for very useful discussions. We are also indebted to Paolo Boldi and Sebastiano Vigna for their deep and inspiring work on self-stabilization.

1 Introduction

There is a considerable interest in distributed systems consisting of multiple, potentially mobile, agents. This is mainly motivated by the emergence of large scale networks, characterized by the lack of centralized control, the access to limited information and a time-varying connectivity. Control and optimization algorithms deployed in such networks should be completely distributed, relying only on local observations and informations, and robust against unexpected changes in topology.

A canonical problem in distributed control is the *mod P-synchronization problem*: In a system where each agent is equipped with a local discrete clock, the objective is that all clocks are eventually congruent modulo some integer P, despite arbitrary initializations. This synchronization problem arises in a number of applications, both in engineering and natural systems. It is a basic block in many engineering systems, e.g., in the universal self-stabilizing algorithm developed by Boldi and Vigna [8], or for deploying distributed algorithms structured

into synchronized *phases* (e.g., the *Two-Phase* and *Three-Phase Commit* algorithms [6], or many consensus algorithms [5, 15, 22, 11]). Periodic clock synchronization also corresponds to an ubiquitous phenomenon in the natural world and finds numerous applications in physics and biology, e.g., the Kuramoto model for the synchronization of coupled oscillators [23], synchronous flashing fireflies, collective synchronization of pancreatic beta cells [20].

Our goal is the design of distributed algorithms achieving mod P-synchronization in a networked system of n agents that operate in synchronous rounds and communicate by broadcast. The network is supposed to be uniform and anonymous, i.e., agents are identical and have no identifiers. We consider the self-stabilization model where the initial state of each agent is arbitrary. In particular, agents do not have a consistent numbering of the rounds. Moreover, agents may use only local informations.

The communication pattern at each round is modeled by a directed graph that may change continually from one round to the next. In other words, we allow for time-varying communication graphs, which is important if we want to take into account link failure and link creation, reconfigurable networks, or for dealing with probabilistic communication models like the rumor spreading models. We impose weak assumptions on the communication topology; in particular, we allow for non-bidirectional links and do not assume full connectivity. Even the assumption of strong connectivity may be too restrictive in various settings: for instance, asynchrony and benign failures in a fully connected network may be handled in the model of this paper (i.e., synchronous and non-faulty networks) by dynamic graphs that are permanently rooted, but not strongly connected [11].

Contribution. Our contribution in this paper is a finite state algorithm, called *SAP* (for self-adaptive period), that synchronizes periodic clocks in a large class of dynamic networks. As opposed to most of previous solutions, our algorithm does not assume any global knowledge on the network, and tolerates time-varying topologies.

First, we show that the *SAP* algorithm solves the mod P-synchronization problem in any dynamic network with a finite *dynamic diameter*, i.e., from any time onward and for every pair of agents i and j, there is a temporal path of bounded length connecting i to j.[1] If a bound on the diameter is given, its stabilization time is bounded above by three times the diameter, whatever the value of the bound. However, the *SAP* algorithm fundamentally works when no bound is available, with a limited increase of stabilization time.

Interestingly, the *SAP* algorithm unifies several seemingly different algorithms for the synchronization of periodic clocks in static networks, including the algorithms in [3, 19, 9] and the one deployed in the finite-state universal self-stabilizing protocol in [8], with useful insights for improving their solvability powers. In particular, we show that the pioneer algorithm proposed by Arora et al. [3] works for a period $P \geqslant 6n$ while the authors proved its correctness only when $P \geqslant n^2$.

Then we study how to relax the assumption of a finite diameter and introduce the class of *strongly centered networks*: Roughly speaking, a strongly centered network corresponds to a dynamic graph containing at least one *central node*, that is, a node that can reach any nodes through a temporal path of bounded length. Moreover, in such a network, non-central nodes are not allowed to infinitely often communicate with central nodes. This class strictly contains all the dynamic networks with a finite dynamic diameter and all the static rooted networks. Thus the property of a strongly centered network allows for non-strong connectivity while authorizing dynamic links. We prove that the *SAP* algorithm still works in this class

[1] Observe that the diameter of a static strongly connected network is less than the number of agents, while it may be arbitrarily large for a dynamic network. This is why the assumption of a bound on the diameter available at each agent may be quite problematic in the dynamic setting.

of dynamic networks. Once again, no global knowledge is assumed. In particular, neither the bound on the length of the paths (for the central nodes to communicate with all nodes) nor the set of central nodes are supposed to be known. Finally, we provide upper bounds on the stabiliization time and the space complexity of each execution of the *SAP* algorithm.

Related work. Self-stabilizing clocks have been extensively studied in different communication models, under different assumptions, and with various problem specifications. In particular, clocks may be unbounded, in which case they are required to be eventually equal, instead of only congruent. The synchronization problem of unbounded clocks admits simple solutions in strongly connected networks, namely the *Min* and *Max* algorithms [16, 18].

The point of periodic clocks is the use finite memory, as opposed to unbounded clocks which inherently require infinite memory. This is why the use of a synchronization algorithm for unbounded clocks with a modulo operation at each round is not appropriate for synchronizing periodic clocks. In addition to strong connectivity and static networks, the pioneer papers on periodic clock synchronization [3, 19, 9, 1] all assume that a bound on the diameter is available. To the best of our knowledge, only the synchronization algorithm in [8] for a *static* communication graph dispenses with the latter assumption.

More recently, periodic clock synchronization has been studied in the *Beeping model* [12] in which agents have severely limited communication capabilities: given a connected bidirectional communication graph, in each round, each agent can either send a "beep" to all its neighbors or stay silent. A self-stabilizing algorithm has been proposed by Feldmann et al. [17], which is optimal both in time and space, but which, unfortunately, requires that a bound on the network size is available for each agent.[2]

There are also numerous results for mod P-synchronization with faulty agents. The fault-tolerant solutions that have been proposed in various failure models, including the Byzantine failure model, using algorithmic schemes initially developed for consensus (e.g., see [13, 14]). They typically require a bidirectional connected (most of the time fully connected) network.

Clock synchronization has also been studied in the model of *population protocols* [2], consisting of a set of agents, interacting in randomly chosen pairs. In this model, the underlying network is assumed to be fully connected, and the pairwise interactions are modeled by bidirectional links. Moreover, only stabilization with probability one or with high probability is required. The same weakening of problem specification is considered for another popular probabilistic communication model, namely the *PULL* model [21], in which, at each round each agent interacts with one random incoming neighbor in a fixed directed graph G. Unfortunately, in addition to a probabilistic weakening of the problem, the self-stabilizing clock synchronization algorithms developed in this model [7, 4] highly rely on the assumption that G is the fully connected graph.

2 Preliminaries

2.1 The computing model

We consider a networked system with a fixed and finite set V of agents. We assume a round-based computational model in the spirit of the Heard-Of model [11]. Point-to-point communications are organized into *synchronized rounds*: each node sends messages to all

[2] In [17], Feldmann et al. also proposed an algorithm that does not use any bound on the network size, but that only tolerates asynchronous starts.

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. Communication at each round t is thus modeled by a directed graph (digraph) $\mathbb{G}(t) = (V, E_t)$: $(i, j) \in E_t$ iff communication from i to j is enabled at round t. There is a self-loop at each node i in all the digraphs $\mathbb{G}(t)$ as i communicates with itself instantaneously. The sequence of digraphs $\mathbb{G} = (\mathbb{G}(t))_{t \geqslant 1}$ is called a *dynamic graph*.

An *algorithm* $\mathcal{A}$ is given by a set $\mathcal{Q}$ of states, a set of messages $\mathcal{M}$, a sending function $\sigma : \mathcal{Q} \to \mathcal{M}$, and a transition function $\delta : \mathcal{Q} \times \mathcal{M}^{\oplus} \to \mathcal{Q}$, where $\mathcal{M}^{\oplus}$ is the set of finite multisets over $\mathcal{M}$.

We consider the *self-stabilization* model where all the nodes start to run the algorithm at round one but their initial states are arbitrary in the set $\mathcal{Q}$. An *execution of* $\mathcal{A}$ with the dynamic graph $\mathbb{G}$ proceeds as follows: In round t ($t = 1, 2 \dots$), every node applies the sending function σ to its current state to generate the message to be broadcasted, then it receives the messages sent by its incoming neighbors in $\mathbb{G}(t)$, and finally applies the transition function δ to its current state and the list of messages it has just received to go to a next state. Given an execution of $\mathcal{A}$, the value of any variable x_i at the end of round t is denoted by $x_i(t)$, and $x_i(0)$ is the initial value of x_i.

2.2 Dynamic graphs

The *product* of two digraphs $G_1 = (V, E_1)$ and $G_2 = (V, E_2)$, denoted $G_1 \circ G_2$, is the digraph with the set of nodes V and with an edge (i, j) if there exists $k \in V$ such that $(i, k) \in E_1$ and $(k, j) \in E_2$. For any dynamic graph $\mathbb{G}$ and any integers $t' \geqslant t \geqslant 1$, we let $\mathbb{G}(t : t') \stackrel{\text{def}}{=} \mathbb{G}(t) \circ \cdots \circ \mathbb{G}(t')$. By convention, $\mathbb{G}(t : t) = \mathbb{G}(t)$, and when $0 < t' < t$, $\mathbb{G}(t : t')$ is the digraph with only a self-loop at each node. The set of i's in-neighbors in $\mathbb{G}(t : t')$ is denoted by $\mathrm{In}_i(t : t')$, and simply by $\mathrm{In}_i(t)$ when $t' = t$.

Every edge (i, j) in $\mathbb{G}(t : t')$ corresponds to a *path in the round interval* $[t, t']$: there exist $t' - t + 2$ nodes $i = k_0, k_1, \dots, k_{t'-t+1} = j$ such that (k_r, k_{r+1}) is an edge of $\mathbb{G}(t + r)$ for each $r = 0, \dots, t' - t$.

The *eccentricity of a node* i in a dynamic graph $\mathbb{G}$, denoted $\mathrm{e}_{\mathbb{G}}(i)$, is defined as

$$\mathrm{e}_{\mathbb{G}}(i) \stackrel{\text{def}}{=} \inf\{d \in \mathbb{N}^+ \mid \forall t \in \mathbb{N}^+, \forall j \in V : (i, j) \text{ is an edge in } \mathbb{G}(t : t + d - 1)\}.$$

The *dynamic diameter* of $\mathbb{G}$ is then defined as:

$$\mathrm{diam}(\mathbb{G}) \stackrel{\text{def}}{=} \sup_{i \in V} \mathrm{e}_{\mathbb{G}}(i).$$

The notion of dynamic diameter generalizes the classical one of diameter of a digraph in the sense that $\mathrm{diam}(\mathbb{G}) = \mathrm{diam}(G)$ if for each positive integer t, $\mathbb{G}(t) = G$. As no confusion can arise, $\mathrm{diam}(\mathbb{G})$ will simply be called the *diameter* of $\mathbb{G}$.

3 The SAP_g algorithm

3.1 Informal description and pseudo-code

Let $\mathcal{A}$ be an algorithm where each node i maintains an *integer* variable C_i, called the clock of i. Given a positive integer P, an execution of $\mathcal{A}$ is said *to achieve mod P-synchronization* if

$$\exists t_0, c, \forall t \geqslant t_0, \forall i \in V,\ C_i(t) \equiv_P t + c,$$

where $C_i(t)$ denotes the value of C_i at the end of round t in the execution. Even in the case of a static strongly connected network, the naive algorithm in which, at each round, each node sends its own C_i and applies the following update rule:

$$C_i \leftarrow \left[\min_{j\in S} C_j + 1\right]_P,$$

(where S is the set of i's incoming neighbors, and $[c]_P$ denotes the remainder of the Euclidean division of c by P) does not work when the network diameter is too large compared to the period P. Theorem 10 provides an execution in which such a system never achieves mod P-synchronization. To overcome this problem, we present an algorithm, called *SAP* (for self-adaptive period), inspired by the ideas developed by Boldi and Vigna for their finite-state universal self-stabilizing algorithm [8]. The key point of the *SAP* algorithm lies in the fact that for any positive integer M, we have

$$\left[\left[\, c\,\right]_{PM}\right]_P = \left[\, c\,\right]_P.$$

More precisely, each node i uses two *integer* variables M_i and C_i, and computes the clock value C_i not modulo P, but rather modulo the time-varying period PM_i. The variable M_i is used as a guess to find a large enough multiple of P so to make the clocks eventually stabilized. Until synchronization, the variables M_i increase so that there is "enough space" between the largest clock value and the shortest period PM_i in the network. The update rule for M_i is parametrized by a non-decreasing function $g : \mathbb{N} \to \mathbb{N}$. The corresponding algorithm is denoted SAP_g, and its code is given below. Line 5 in the pseudo-code implies that $C_i(t) < PM_i(t)$, and for the sake of simplicity, we assume that this inequality also holds initially, that is, $C_i(0) < PM_i(0)$.

Let $g : \mathbb{N} \to \mathbb{N}$ be a non-decreasing function. If q is a positive integer, g^q denotes the q-th iterate of g, and g^0 is the identity function. For every non-negative integer m, we let

$$g^*(m) \stackrel{\text{def}}{=} \inf\{q \in \mathbb{N} \mid g^q(0) \geqslant m\}.$$

Algorithm 1 The SAP_g algorithm, pseudo-code of node i.

Variables:
1: $C_i \in \mathbb{N}$;
2: $M_i \in \mathbb{N}^+$;
In each round do:
3: send $\langle C_i, M_i\rangle$ to all
4: receive $\langle C_{j_1}, M_{j_1}\rangle, \langle C_{j_2}, M_{j_2}\rangle, \ldots$ from the set S of incoming neighbours
5: $C_i \leftarrow \left[\min_{j\in S} C_j + 1\right]_{PM_i}$
6: $M_i \leftarrow \max_{j\in S} M_j$
7: **if** $C_j \not\equiv_P C_{j'}$ for some $j, j' \in S$ **then**
8: $\quad M_i \leftarrow g(M_i)$
9: **end if**

3.2 Notation and basic invariants

We fix an execution of SAP_g with the dynamic graph $\mathbb{G}$, and for any $t \in \mathbb{N}$, we let

$$\tilde{M}(t) \stackrel{\text{def}}{=} \min_{i\in V} M_i(t).$$

For each round t in this execution, let i^+_t denote any one of i's incoming neighbours in $\mathbb{G}(t)$ that satisfies

$$C_{i^+_t}(t-1) = \min_{j \in \mathrm{In}_i(t)} C_j(t-1).$$

Given an integer $P > 0$, the system is said to be *synchronized* (for *mod P-synchronized*) *in round t* if

$$\forall i, j \in V, \;\; C_i(t) \equiv_P C_j(t).$$

We start with two preliminary lemmas.

▶ **Lemma 1.**
1. *If the system is synchronized in round s, then it is synchronized in any round $t \geqslant s$.*
2. *If (i,j) is an edge in $\mathbb{G}(s:t)$, then $C_j(t) \leqslant C_i(s-1) + t - s + 1$.*
3. *Each variable M_i is non-decreasing.*

The second claim of Lemma 1 simply follows from the fact that $C_i(t+1) \leqslant C_j(t) + 1$ for every round t and every pair of nodes $i \in V$ and $j \in \mathrm{In}_i(t)$. The first and third claims directly follow from the transition function.

▶ **Lemma 2.** *For each round $t \geqslant 1$ and each $i \in V$, one of the following statements is true:*
1. *$C_i(t)$ is positive and $C_i(t) = 1 + C_{i^+_t}(t-1)$*
2. *$C_i(t) = 0$, $C_i(t-1) = PM_i(t-1) - 1$, and $i^+_t = i$.*

Proof. The lemma just relies on the following series of inequalities:

$$C_{i^+_t}(t-1) \leqslant C_i(t-1) \leqslant PM_i(t-1) - 1.$$

The last inequality is clear for $t = 1$, and for $t \geqslant 2$, it is a consequence of $C_i(t-1) \leqslant PM_i(t-2) - 1$ and of the fact that M_i is non-decreasing. If one of those inequalities is strict, then assertion 1 in the lemma holds. Otherwise, assertion 2 holds. ◀

Given any node i and two rounds t and t', we introduce the set

$$S^t_i(t') \stackrel{\text{def}}{=} \{j \in V \mid C_j(t') \equiv_P C_i(t) + t' - t\},$$

and the integer

$$\tilde{M}^t_i(t') \stackrel{\text{def}}{=} \inf_{j \notin S^t_i(t')} M_j(t').$$

Intuitively, $S^t_i(t')$ is the set of nodes whose clocks in round t' are "in accordance" with i's clock at round t. In each round t', V may be partitioned into subsets of nodes whose clocks are all congruent mod P, and each $S^t_i(t')$ is either empty, or is equal to one part of this partition. Once the system is synchronized, this partition contains only one part. Clearly, i belongs to $S^t_i(t)$, but i may not belong to $S^t_i(t')$ when $t' \neq t$.

▶ **Lemma 3.**
1. *If $j \in S^t_i(t'+1)$, then $j^+_{t'+1} \in S^t_i(t')$.*
2. *If $t \geqslant t'$, then $S^t_i(t') \neq \emptyset$.*
3. *$\tilde{M}^t_i(t'+1) \geqslant \tilde{M}^t_i(t')$.*

Proof. The first claim (1) is a direct consequence of the definition of $j^+_{t'+1}$. Then, using (1), we demonstrate the second claim by induction on $t - t' \geqslant 0$. Finally, the pseudo-code of SAP_g implies $M_i(t'+1) \geqslant M_{i^+_{t'+1}}(t')$, and hence, $\tilde{M}^t_i(t'+1) \geqslant \tilde{M}^t_i(t')$. ◀

3.3 Correctness proof with a finite diameter

We fix a dynamic graph $\mathbb{G}$ whose diameter is finite and an execution of SAP_g with $\mathbb{G}$, and we let $\mathrm{diam}(\mathbb{G}) = D$.

▶ **Lemma 4.** *Let ℓ be any round. Let i_0 be a node whose clock value is minimum in some round t, and let δ be any integer that is greater or equal to $\mathrm{e}_{\mathbb{G}}(i_0)$. One of the following statements is true:*

1. *there exist a round $d \in \{1, \dots, \delta - 1\}$ and a node $i \notin S^{\ell}_{i_0}(t+d)$ such that $C_i(t+d) = 0$;*
2. *the system is synchronized in round $t + \delta$.*

Proof. Let us assume that the first proposition does not hold. First, we prove by induction on $d \in \{1, \dots, \delta - 1\}$, that

$$\forall i \notin S^{\ell}_{i_0}(t+d), \quad C_i(t+d) = d + \min_{j \in \mathrm{In}_i(t+1:t+d)} C_j(t). \tag{1}$$

The base case $d = 1$ is an immediate consequence of Lemma 2. For the inductive step, assume that Eq. (1) holds for some $d \in \{1, \dots, \delta - 1\}$. For every node $i \notin S^{\ell}_{i_0}(t+d+1)$, we have

$$\begin{aligned} C_i(t+d+1) &= 1 + \min_{j \in \mathrm{In}_i(t+d+1)} C_j(t+d) \\ &= 1 + d + \min_{j \in \mathrm{In}_i(t+d+1)} \left(\min_{k \in \mathrm{In}_j(t+1:t+d)} C_k(t) \right) \\ &= 1 + d + \min_{k \in \mathrm{In}_i(t+1:t+d+1)} C_k(t). \end{aligned}$$

The first equality is a direct consequence of Lemma 2. The second one is by the inductive hypothesis applied to i^+_{t+d+1}. Notice that $i^+_{t+d+1} \notin S^{\ell}_{i_0}(t+d)$ since $i \notin S^{\ell}_{i_0}(t+d+1)$. The third one is because $\mathbb{G}(t+1 : t+d+1) = \mathbb{G}(t+1 : t+d) \circ \mathbb{G}(t+d+1)$. This completes the proof of Eq. (1) for every integer $d \in \{1, \dots, \delta - 1\}$.

Then for each node $i \notin S^{\ell}_{i_0}(t+\delta)$, we get

$$\begin{aligned} C_i(t+\delta) &= \left[1 + \min_{j \in \mathrm{In}_i(t+\delta)} C_j(t+\delta-1) \right]_{PM_i(t+\delta-1)} \\ &= \left[\delta + \min_{j \in \mathrm{In}_i(t+\delta)} \left(\min_{k \in \mathrm{In}_j(t+1:t+\delta-1)} C_k(t) \right) \right]_{PM_i(t+\delta-1)} \\ &= \left[\delta + \min_{k \in \mathrm{In}_i(t+1:t+\delta)} C_k(t) \right]_{PM_i(t+\delta-1)} \\ &= [\delta + C_{i_0}(t)]_{PM_i(t+\delta-1)} \end{aligned}$$

The first equality is by line 5. The second equality is due to Eq. (1) at round $t + \delta - 1$ and the fact that if $i \notin S^{\ell}_{i_0}(t+\delta)$ implies that $i^+_{t+\delta} \notin S^{\ell}_{i_0}(t+\delta-1)$. The fourth one is a consequence of the definition of i_0 and $\mathrm{e}_{i_0}(\mathbb{G}) \leqslant \delta$. It follows that all the clocks $C_i(t+\delta)$ are equal modulo P, i.e., the system is synchronized in round $t + \delta$. ◀

Using the assumption of a finite diameter D, we then derive the following lemma.

▶ **Lemma 5.** *Let t be a round in which $C_i(t) + D \leqslant PM_i(t)$ holds for each node i. Then the system is synchronized in round $t + D$.*

Proof. Let i be any node, and let $d \in \{1, \ldots, D-1\}$. We have

$$\begin{aligned} C_i(t+d-1) &\leqslant d-1+C_i(t) \\ &< D-1+C_i(t) \\ &\leqslant PM_i(t)-1 \\ &\leqslant PM_i(t+d-1)-1. \end{aligned}$$

The first and fourth inequalities are direct consequences of Lemma 1, and the third inequality is the assumption of the lemma. By Lemma 2, it follows that $C_i(t+d) \neq 0$. Since D is greater of equal to each $\mathrm{e}_{\mathbb{G}}(i)$, Lemma 4 shows that the system is synchronized in round $t+D$. ◀

The next lemma focuses on the self-adaptive period mechanism in the SAP_g algorithm. It intuitively states that, as long as all nodes hear of at least one node whose clock is "in accordance" with i's clock at round t, every node j not in accordance with $C_i(t)$ increases its M_j variable.

▶ **Lemma 6.** *Let t, q and δ be three positive integers, and let $i \in V$. If it holds that*

$$\forall j \in V, \forall \ell \leqslant (q-1)\delta, \quad \mathrm{In}_j(\ell+1 : \ell+\delta) \cap S_i^t(\ell) \neq \emptyset,$$

then $\tilde{M}_i^t(q\delta) \geqslant g^q(0)$.

Proof. We proceed by induction on q. The base case is obvious since each $M_j(0)$ is non-negative. For the inductive step, assume that the lemma holds in round $q\delta$ and that some node j does not belong to $S_i^t((q+1)\delta)$. By assumption, the node j has an in-neighbor $k \in S_i^t(q\delta)$ in the digraph $\mathbb{G}(q\delta+1 : (q+1)\delta)$, i.e., there exist a node $k \in S_i^t(q\delta)$ and a path $k = j_0, j_1, \cdots, j_\delta = j$ in the round interval $[q\delta+1, (q+1)\delta]$. We have

$$k \in S_i^t(q\delta) \text{ and } j \notin S_i^t((q+1)\delta).$$

Let $d \in \{1, \ldots, \delta\}$ be the first index such that

$$j_{d-1} \in S_i^t(q\delta+d-1) \text{ and } j_d \notin S_i^t(q\delta+d).$$

By Lemma 3, $(j_d)^+_{q\delta+d} \notin S_i^t(q\delta+d-1)$ since $j_d \notin S_i^t(q\delta+d)$. Then j_{d-1} and $(j_d)^+_{q\delta+d}$ are two in-neighbors of j_d whose clocks are not congruent modulo P in round $q\delta+d-1$. It follows that:

$$M_j((q+1)\delta) \geqslant M_{j_d}(q\delta+d) \geqslant g(M_{(j_d)^+_{q\delta+d}}(q\delta+d-1)) \geqslant g(\tilde{M}_i^t(q\delta+d-1)) \geqslant g^{q+1}(0).$$

The first two inequalities are due to the update rules for M_j and M_{j_d}. The third one is by definition of $\tilde{M}_i^t$ and the fact that g is non-decreasing. The last one is a consequence of the inductive assumption and the third claim of Lemma 3. ◀

Using the assumption of a finite diameter D, we then derive the following lemma.

▶ **Lemma 7.** *For all non-negative integer $q \in \mathbb{N}$, one of the following statements is true:*

1. *the system is synchronized in round qD;*
2. *$\tilde{M}(qD) \geqslant g^q(0)$.*

Proof. Assume that two nodes i_0 and i_1 hold non-congruent clocks in round qD. For each positive integer $\ell \leqslant qD$, the second claim of Lemma 3 gives that $S_{i_0}^{qD}(\ell)$ and $S_{i_1}^{qD}(\ell)$ are both non-empty. Since D is the diameter of $\mathbb{G}$, for each node $j \in V$, we have

$$\mathrm{In}_j(\ell+1:\ell+D) \cap S_{i_0}^{qD}(\ell) \neq \emptyset \quad \text{and} \quad \mathrm{In}_j(\ell+1:\ell+D) \cap S_{i_1}^{qD}(\ell) \neq \emptyset.$$

By Lemma 6, this implies

$$\tilde{M}_{i_0}^{qD}(qD) \geqslant g^q(0) \text{ and } \tilde{M}_{i_1}^{qD}(qD) \geqslant g^q(0),$$

and hence, $\tilde{M}(qD) \geqslant g^q(0)$. ◀

▶ **Theorem 8.** *In any execution with a dynamic graph whose diameter D is finite, the SAP_g algorithm achieves mod P-synchronization for any non-decreasing function $g : \mathbb{N} \to \mathbb{N}$ such that $g^*\left(\lceil \frac{2D}{P} \rceil\right)$ is finite. Moreover, the stabilization time is bounded by $\left(g^*\left(\lceil \frac{2D}{P} \rceil\right) + 2\right) D$.*

Proof. We let $q_0 = g^*\left(\lceil \frac{2D}{P} \rceil\right)$; hence $q_0 \geqslant 1$. Applying Lemma 4 with $\delta = D$ and $t = (q_0 - 1)D$, we obtain that either the system is synchronized in round $q_0\, D$, or there exist a node j and an integer $d \in \{1, \ldots, D-1\}$ such that $C_j(q_0\, D + d - D) = 0$. The digraph $\mathbb{G}(q_0\, D + d - D + 1 : q_0\, D + d)$ is complete since D is the diameter of $\mathbb{G}$, and the second claim in Lemma 1 implies that $C_i(q_0\, D + d) \leqslant D$ for any node $i \in V$. Hence,

$$\begin{aligned} PM_i(q_0 D + d) &\geqslant PM_i(q_0\, D) \\ &\geqslant P\tilde{M}(q_0\, D) \\ &\geqslant Pg^{q_0}(0) \\ &\geqslant 2D \\ &\geqslant C_i(q_0\, D + d) + D \end{aligned}$$

The third inequality is by Lemma 7 and the fourth one is due to the definition of q_0. Finally, Lemma 5 shows that the system is synchronized in round $(q_0 + 1)D + d$. ◀

3.4 Specializations of the SAP_g Algorithm

We consider the following two strategies:

1. The function g is constant and equal to M, in which case $g^*(m) = 1$ if $m \leqslant M$, and $g^*(m) = \infty$ otherwise.
2. The function g^* takes only finite values. This is equivalent to the fact that g has no fixed point, or g is strictly inflationary, i.e., $m < g(m)$ for every non-negative integer m.

Theorem 8 leads to two corollaries corresponding to each of the strategies on g. Firstly, when some bound B on the diameter of the dynamic graph is given, we may choose g to be the constant function $g = \lambda x.M$ with $M = \lceil \frac{2B}{P} \rceil$. Then we get $g^*\left(\frac{2D}{P}\right) = 1$ and the pseudo-code SAP_g reduces to Algorithm 2.

▶ **Corollary 9.** *The $SAP_{\lambda x.M}$ algorithm solves the mod P-synchronization problem in any dynamic graph with a diameter less than or equal to $PM/2$.*

Let us observe that Theorem 8 provides an upper bound of three times the diameter D on $SAP_{\lambda x.M}$'s stabilization time, which is independent on the bound B. The limit of $PM/2$ in Corollary 9 is tight, as proved by the following result.

Algorithm 2 The $SAP_{\lambda x.M}$ algorithm.

Variables:
1: $C_i \in \mathbb{N}$;
In each round do:
2: send $\langle C_i \rangle$ to all
3: receive $\langle C_{j_1} \rangle, \langle C_{j_2} \rangle, \ldots$ from the set S of in-neighbors
4: $C_i \leftarrow \left[\min_{j \in S} C_j + 1\right]_{PM}$

▶ **Theorem 10** (Theorem 4.13 in [1]). *For any even integers P and D satisfying $P < 2D$, there exists an execution of $SAP_{\lambda x.1}$ with a dynamic graph $\mathbb{G}$ whose diameter is D in which mod P-synchronization is never achieved.*

Interestingly, the self-stabilizing algorithm in [9], called *SS-MinSU* and developed for clock synchronization in a static and strongly connected network when a bound B on the diameter[3] is available, is actually an optimization of the $SAP_{\lambda x.M}$ algorithm.

As for the algorithm proposed in [3] for a static strongly connected digraph G, it corresponds to the $SAP_{\lambda x.1}$ algorithm, combined with a round-robin strategy which consists, for each node, to send one message per round according to this fixed cyclic order amongst the outgoing neighbors in G. This strategy thus translates the fixed digraph G into a dynamic graph $\mathbb{G}$. Using Proposition 24 in [10], $\mathbb{G}$'s diameter can be upper bounded by $3|V|$. Via Corollary 9, the interpretation of the algorithm in [3] for a fixed digraph G in terms of a run of $SAP_{\lambda x.1}$ over the corresponding dynamic graph $\mathbb{G}$ shows that this algorithm works when $P \geqslant 6|V|$, and its stabilization time is less than $9|V|$ (instead of the correctness condition $P \geqslant n^2$ and the stabilization bound of $\frac{3}{2}n^2$, given both in [3]).

When the diameter of the dynamic graph is finite but no bound is available, we may use the following corollary of Theorem 8:

▶ **Corollary 11.** *For any non-decreasing and inflationary function g, SAP_g solves the mod P-synchronization in any dynamic graph whose diameter is finite.*

The idea of a self-adaptive period is borrowed from the seminal paper by Boldi and Vigna [8], and the SAP_g algorithm is a variant of the algorithm they presented for static strongly connected communication graphs. From the viewpoint of design, the main discrepancy lies in the period lengths equal to PM_i in SAP_g, instead of PM_i^2 in Boldi and Vigna's algorithm. As a result, the two algorithms differ in space complexity: while the variables C_i in SAP_g are of the order of $PM(q_0\, D)$, the algorithm in [8] uses variables of the order of $PM(q_0\, D)^2$, where $q_0 = g^*\left(\left\lceil \frac{2D}{P} \right\rceil\right)$; see Section 5.

4 The SAP_g algorithm with infinite diameter

The aim of this section is to study how the assumption of a finite diameter can be relaxed so that the SAP_g algorithm still achieves mod P-synchronization.

[3] The bound B is denoted α in the *SS-MinSU* algorithm.

4.1 Extending the class of static rooted networks

A node i is said to be *central in* a dynamic graph $\mathbb{G}$ if its eccentricity is finite, and the *center* of $\mathbb{G}$, denoted by $Z(\mathbb{G})$, is defined as the set of $\mathbb{G}$'s central nodes.

$$Z(\mathbb{G}) \stackrel{\text{def}}{=} \{i \in V \mid \mathrm{e}_{\mathbb{G}}(i) < \infty\}.$$

If $Z(\mathbb{G})$ is non-empty, then the following integer is well-defined and finite.

$$R \stackrel{\text{def}}{=} \max_{i \in Z(\mathbb{G})} \mathrm{e}_{\mathbb{G}}(i). \tag{2}$$

The *kernel* of $\mathbb{G}$, denoted $K(\mathbb{G})$, is defined as

$$K(\mathbb{G}) \stackrel{\text{def}}{=} \{i \in V \mid \forall t \geqslant 1, \forall j \in V, \exists\, t' \geqslant t : (i,j) \text{ is an edge in } \mathbb{G}(t:t')\}\,.$$

Intuitively, a node belongs to $K(\mathbb{G})$ if it can infinitely often reach all nodes in finite time. Clearly, it holds that $Z(\mathbb{G}) \subseteq K(\mathbb{G})$. The inclusion is strict in general, as illustrated in Section 4.2. Indeed, the construction guarantees that $Z(\mathbb{G}) = \{i\}$ and $K(\mathbb{G}) = V$. A dynamic graph $\mathbb{G}$ is said to be *strongly centered* if $Z(\mathbb{G}) \neq \emptyset$ and $K(\mathbb{G}) = Z(\mathbb{G})$.

▶ **Lemma 12.** *The center of any strongly centered dynamic graph $\mathbb{G}$ has no incoming edge from some index t_0.*

Proof. We denote $\mathbb{G}(\infty)$ a digraph whose set of nodes is V that contains every edge that appears infinitely often in $\mathbb{G}$. By definition of $K(\mathbb{G})$, each node $i \in K(\mathbb{G})$ can infinitely often reach each node $j \in V$ in the dynamic graph $\mathbb{G}$, whereas there are finitely many paths between any two nodes. By the pigeonhole principle, each node in $K(\mathbb{G})$ is the root of a spanning tree in $\mathbb{G}(\infty)$. Using the definitions of $K(\mathbb{G})$ and $\mathbb{G}(\infty)$, the converse can also be proved. Then $K(\mathbb{G})$, and hence $Z(\mathbb{G})$ have no incoming edge in $\mathbb{G}(\infty)$, since if i is the root of some spanning tree in $\mathbb{G}(\infty)$, then all i's incoming neighbours are also roots of a spanning tree. Then, from a certain round, $Z(\mathbb{G})$ has no incoming edge in $\mathbb{G}$. ◀

In the self-stabilizing paradigm, any predicate that holds from a certain round can be assumed to hold from the beginning. We may then assume $t_0 = 0$ in the rest of the paper.

4.2 The SAP_g algorithm with a central node

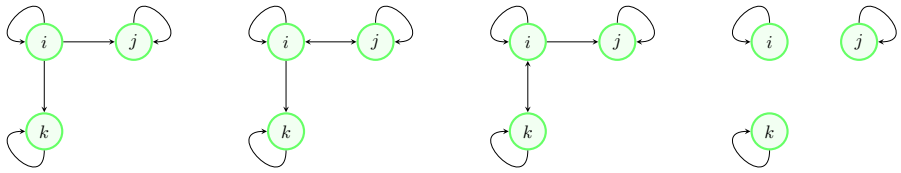
We now study whether SAP_g can achieve mod P-synchronization in networks with an infinite diameter. For that, we first demonstrate that the sole assumption of a non-empty center is not sufficient for SAP_g to achieve mod P-synchronization. We construct an execution of SAP_g with a central node i. The underlying idea of our scenario is that sporadic incoming neighbors disrupt the value of i's clock and hence preclude any alignment of the other clocks on C_i.[4]

Let G, H_j, H_k, I be the four digraphs defined in Figure 1 with three nodes i, j, k, and let Φ_k be the following predicate on the rounds of a SAP_g execution:

$$\big(M_i = M_j\big) \wedge \big(M_i \geqslant M_k\big) \wedge \big(C_i = C_j\big) \wedge \big(C_i \not\equiv_P 0\big) \wedge \big(C_i \leqslant PM_i - 2\big) \wedge \big(C_k = 0\big).$$

The predicate Φ_j is obtained by exchanging the roles of j and k. The proof of the following lemma, which is omitted, follows from a step by step execution of the SAP_g algorithm between rounds t and $t + PM - c$.

[4] We provide a Python script that may be helpful to verify the correctness of our construction: `https://gitlab.com/bossuet/sap_execution.git`.



(a) digraph G. **(b)** digraph H_j. **(c)** digraph H_k. **(d)** digraph I.

Figure 1 Four digraphs with three nodes.

▶ **Lemma 13.** *Let t be a round of a SAP_g execution with a dynamic graph $\mathbb{G}$, and let M and c denote $M_i(t)$ and $C_(t)$, respectively. Let $\mathbb{G}'$ be any dynamic graph that coincides with $\mathbb{G}$ up to t and such that:*

$$\mathbb{G}'(t+1) = \cdots = \mathbb{G}'(t+PM-c-2) = G,\ \mathbb{G}'(t+PM-c-1) = H_k,\ \mathbb{G}'(t+PM-c) = I.$$

If Φ_k holds at round t of the SAP_g execution with $\mathbb{G}'$, then Φ_j holds at round $t+PM-c$ of this execution.

We now fix two positive integers M^0 and c^0 such that $c^0 \in \{1, \cdots, PM^0-2\}$ and $c^0 \not\equiv_P 0$, and we consider the two sequences $(M^r)_{r\geqslant 0}$ and $(c^r)_{r\geqslant 0}$ defined by:

$$\begin{cases} M^{r+1} = g^{PM^r-c^r-1}(M^r) \\ c^{r+1} = PM^r - c^r. \end{cases}$$

We let $M^{-1} = 0$. The dynamic graph $\mathbb{G}$ defined as:

$$\begin{aligned} &\mathbb{G}(PM^{r-1}+1) = \cdots = \mathbb{G}(PM^r - c^r - 2) = G, \\ &\mathbb{G}(PM^r - c^r - 1) = H_k \text{ or } H_j, \\ &\mathbb{G}(PM^r - c^r - 1) = I, \end{aligned}$$

is rooted with delay two and i is its unique center. Lemma 13 shows that Φ_k holds infinitely often in the SAP_g execution with the dynamic graph $\mathbb{G}$ and starting with:

$$M_i(0) = M_j(0) = M_k(0) = M^0,\ C_i(0) = C_j(0) = c^0, \quad \text{and} \quad C_k(0) = 0.$$

Hence, the nodes are never synchronized.

4.3 The SAP_g algorithm in strongly centered network

That leads us to consider the stronger assumption that the network is strongly centered, without requiring any global knowledge on $Z(\mathbb{G})$. However, the simple but typical scenario below shows that the simplified version of SAP_g with a fixed period, namely the $SAP_{\lambda x.M}$ algorithm, does not achieve mod P-synchronization in the execution with the initial values $C_i(0) = C_j(0) = 1$ and $C_k(0) = 0$ and the static graph H defined in Figure 2, even for large value of M. Indeed, at each round t, it holds that $C_i(t) = [t+1]_{PM}$, $C_k(t) = [t]_{PM}$, and

$$C_j(t) = \begin{cases} 1 & \text{if } [t]_{PM} = 0 \\ [t]_{PM} & \text{otherwise.} \end{cases}$$

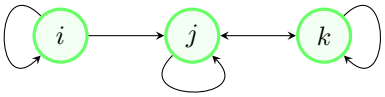


Figure 2 The digraph H with three nodes.

The striking point of increasing periods is precisely to overcome the above-mentioned limitation: we are going to prove that the SAP_g algorithm achieves mod P-synchronization in the case of a strongly centered network under the sole condition of a non-decreasing and strictly inflationary function g. In other words, while Corollary 9 has no counterpart for strongly centered dynamic graphs, we will show that Corollary 11 extends to this latter class of dynamic graphs.

We fix a strongly centered dynamic graph $\mathbb{G}$, and an execution σ of SAP_g with $\mathbb{G}$. Lemma 12 shows that, from a certain round t_0, the nodes in $Z(\mathbb{G})$ (denoted Z, for short) receive no message from the nodes in $V \setminus Z$. From round t_0, from the viewpoint of every node in Z, the execution σ is thus indistinguishable from an execution with the set of nodes equal to Z and a dynamic graph whose diameter is finite. Theorem 8 shows that mod P-synchronization is eventually achieved in Z. A closer look at the SAP_g algorithm yields the following more precise result: there exist two non-negative integers s and M such that

$$\forall t \geqslant s,\ \forall k, \ell \in Z,\ C_k(t) \equiv_P C_\ell(t) \quad \text{and} \quad M_k(t) = M. \tag{3}$$

The minimum integer s satisfying Eq. (3) is denoted by t_1. This integer corresponds to the round in which the subsystem composed of central nodes achieves mod P-synchronization. Recalling that R is the integer defined in Eq.(2), we also define the following two constants.

$$q_1 \stackrel{\text{def}}{=} g^*\left(\left\lceil M + \frac{R+1}{P}\right\rceil\right) \text{ and } t_2 \stackrel{\text{def}}{=} \max(t_1, q_1 R). \tag{4}$$

▶ **Lemma 14.** *Let i_0 be any central node. For any $t > t_2$, and any nodes $j \notin S^{t_2}_{i_0}(t)$, it holds that $C_j(t) \neq 0$.*

Proof. Let j be any node that does not belong to $S^{t_2}_{i_0}(t)$. By definition of R, there exists an edge (i_0, j^+_t) in each digraph $\mathbb{G}(t-R : t-1)$. The third claim of Lemma 1 and Eq. (3) imply that $C_{i_0}(t-R-1) < PM$. Then the second claim in Lemma 1 implies that $C_{j^+_t}(t-1) < PM + R$. Moreover, using the second claim of Lemma 3, each integer $t' \leqslant t_2$ satisfies

$$\mathrm{In}_j(t'+1 : t'+R) \cap S^{t_2}_{i_0}(t') \supseteq Z \cap S^{t_2}_{i_0}(t') \neq \emptyset.$$

Applying the third claim of Lemma 3, Lemma 6 and the definition of q_1,

$$P\tilde{M}^{t_2}_{i_0}(t_2) \geqslant P\tilde{M}^{t_2}_{i_0}(q_1 R) \geqslant P g^{q_1}(0) \geqslant PM + R + 1. \tag{5}$$

From $j \notin S^{t_2}_{i_0}(t)$, we have $j^+_t \notin S^{t_2}_{i_0}(t-1)$ by Lemma 3, and then

$$C_{j^+_t}(t-1) < PM + R \leqslant P\tilde{M}^{t_2}_{i_0}(t_2) - 1 \leqslant P\tilde{M}^{t_2}_{i_0}(t-1) - 1 \leqslant PM_{j^+_t}(t-1) - 1.$$

The second inequality comes from Eq. (5), the third from Lemma 3, and the fourth is a consequence of $j^+_t \notin S^{t_2}_{i_0}(t-1)$. We obtain $C_j(t) \neq 0$ by Lemma 2. ◀

▶ **Theorem 15.** *In any execution with a strongly centered dynamic graph, the SAP_g algorithm achieves mod P-synchronization for any non-decreasing and inflationary function g. Moreover, the stabilization time is bounded by t_2+PM+R, where R, M and t_2 are defined by Eq.(2), (3) and (4).*

Proof. Each node $i \in Z$ satisfies $C_i(t) < PM$ in each round. Then we obtain that some node $i_0 \in Z$ reaches $C_{i_0}(t_3) = 0$ in some round $t_3 \in \{t_2, \ldots, t_2 + PM - 1\}$ (otherwise, an inductive reasoning using Lemma 2 would contradict the above-mentioned fact). By Lemma 14, each node $j \notin S_{i_0}^{t_2}(t_3 + d)$ satisfies $C_j(t_3 + d) \neq 0$ for each $d > 0$. By Lemma 4, the system is synchronized in round $t_3 + R$. ◀

5 Complexity analysis

In this section, we provide a complexity analysis of SAP_g in the case of a network with finite diameter, and then in the case of a strongly centered network. We discuss the choice of the g function and its impact on both stabilization time and space complexity. We first state a theorem that will be used to bound the memory usage, measured in bits, of each node in any execution of SAP_g that achieves mod P-synchronization.

▶ **Theorem 16.** *In any execution of SAP_g that achieves mod P-synchronization, if q is the round in which the system synchronizes, then the memory usage of each node is less than $\log_2 P + 2\log_2\left(g^q\left(\max_{i\in V} M_i(0)\right)\right)$ bits.*

Proof. We define, for each round t,

$$\overline{M}(t) \stackrel{\text{def}}{=} \max_{i\in V} M_i(t).$$

From the pseudo-code of SAP_g, we directly obtain, for each positive integer t,

$$\overline{M}(t) \leqslant g\left(\overline{M}(t-1)\right),$$

and thus,

$$\overline{M}(t) \leqslant g^t\left(\overline{M}(0)\right). \tag{6}$$

As $\overline{M}(t)$ is non-decreasing as long as $t \leqslant q$ and is stable afterwards, each M_i belongs to the interval $\{1, \ldots, \overline{M}(q)\}$, and each C_i belongs to $\{0, \ldots, P\overline{M}(q) - 1\}$. The number of reachable states by any single node is at most equal to the cardinality of the product of these two sets, that is, $Pg^q\left(\overline{M}(0)\right)^2$. Then at most $\log_2 P + 2\log_2\left(g^q\left(\overline{M}(0)\right)\right)$ bits are needed to store the state of one node. ◀

5.1 Networks with finite diameter

Theorem 16 implies the following corollary in the case of a network with finite diameter.

▶ **Corollary 17.** *In any execution of SAP_g, if the diameter D of the network is finite, then the memory usage of each node is bounded by $\log_2 P + 2\log_2\left(g^{(g^*(2D/P+1)+2)D}\left(\max_{i\in V} M_i(0)\right)\right)$.*

Theorem 8 and Corollary 17 demonstrate some trade-off between stabilization time and space complexity. The faster g grows, the lower the synchronization time is, and the higher its space complexity is. To further illustrate this trade-off, Table 1 provides the time and

space complexity in three cases. First, when a bound B on the diameter is given, choosing $g = \lambda x. \lceil \frac{2B}{P} \rceil$ provides the best stabilization time, namely $3D$, which interestingly does not depend on the bound B. When no bound on the diameter is available, the overhead of $SAP_{\lambda x.2x}$ over $SAP_{\lambda x.\lceil 2B/P \rceil}$ is only logarithmic while $SAP_{\lambda x.x+1}$ results in an additional delay of $O(D^2)$ rounds for stabilization.

Regarding space complexity, $SAP_{\lambda x.\lceil 2B/P \rceil}$ and $SAP_{\lambda x.x+1}$ uses $O(\log B)$ and $O(\log D)$ bits, respectively. This illustrates how SAP_g may be more memory-efficient using its adaptive mechanism and a judicious choice of g. By contrast, the space complexity of $SAP_{\lambda x.2x}$ is only linear in D, which might be problematic for memory-constrained devices.

Table 1 Complexity bounds of SAP_g in networks with finite diameter D and $B \geqslant D$.

g	synchronization time	space complexity
$g = \lambda x. \lceil \frac{2B}{P} \rceil$	$3D$	$\log_2 P + 2\log_2 \lceil \frac{2B}{P} \rceil$
$g = \lambda x.x + 1$	$\left(\frac{2D}{P} + 3\right) D$	$\log_2 P + 2\log_2 \left(\max\limits_{i \in V} M_i(0) + \frac{2D^2}{P} + 3D \right)$
$g = \lambda x.2x$	$\left(\log_2 \left(1 + \frac{2D}{P}\right) + 2\right) D$	$\log_2 P + 2\log_2 \left(\max\limits_{i \in V} M_i(0) \right)$ $+ 2D \log_2 \left(1 + \frac{2D}{P}\right) + 4D$

5.2 Strongly centered networks

In the case of a strongly centered network, Theorem 15 bounds the stabilization time by $t_2 + PM + R$. Eq. (6) then provides the following upper bound for M, where t_1 is the minimum integer satisfying Eq.(3).

$$M \leqslant g^{t_1}(\max_{i \in V} M_i(0)),$$

and thus, we obtain:

$$\begin{aligned} t_2 &\leqslant \max\left(t_1, Rg^*\left(\left\lceil g^{t_1}(\max_{i \in V} M_i(0)) + \frac{R+1}{P} \right\rceil\right)\right) \\ &\leqslant Rg^*\left(g^{t_1}(\max_{i \in V} M_i(0)) + \frac{R+1}{P} + 1\right). \end{aligned}$$

Theorem 8 also provides the following upper bound for t_1:

$$t_1 \leqslant \left(g^*\left(\left\lceil \frac{2D}{P} \right\rceil\right) + 2\right) D,$$

where D is the diameter of the (dynamic) subgraph of $\mathbb{G}$ induced by Z. Theorem 16 then implies the following corollary in the case of a strongly centered network.

▶ **Corollary 18.** *In any execution of SAP_g in which the network is strongly centered, the memory usage of each node is bounded by*

$$\log_2 P + 2\log_2 \left(g^{Rg^*\left(M' + \frac{R+1}{P} + 1\right) + PM' + R}\left(\max_{i \in V} M_i(0) \right) \right),$$

where $M' = g^{(g^*(2D/P+1)+2)D}(\max\limits_{i \in V} M_i(0))$.

Table 2 Complexity bounds of SAP_g in strongly centered networks. Here, R is the quantity defined by Eq.(2), and D is the diameter of the dynamic subgraph induced by $Z(\mathbb{G})$.

g	synchronization time	space complexity
$g = \lambda x.x+1$	$\left(t_1 + \max_{i\in V} M_i(0)\right)(P+R)$ $+R\left(2+\frac{R+1}{P}\right)$ where t_1 satisfies $t_1 \leqslant \left(\frac{2D}{P}+3\right)D$	$\log_2 P + 2\log_2\left(\left(t_1 + \max_{i\in V} M_i(0)\right)(P+R)\right.$ $\left.+R\left(2+\frac{R+1}{P}\right) + \max_{i\in V} M_i(0)\right)$
$g = \lambda x.2x$	$P\left(\max_{i\in V} M_i(0)\right)2^{t_1} + R(1+t_1)$ $+R\log_2\left(\left(\max_{i\in V} M_i(0)\right)\left(1+\frac{R+1}{P}\right)\right)$ where $t_1 \leqslant \log_2\left(1+\frac{2D}{P}\right)D + 2D$	$\log_2 P + 2\log_2\left(\max_{i\in V} M_i(0)\right) + 2P\left(\max_{i\in V} M_i(0)\right)2^{t_1}$ $+2R\left(1+t_1+\log_2\left(\left(\max_{i\in V} M_i(0)\right)\left(1+\frac{R+1}{P}\right)\right)\right)$

Table 2 provides a bound on synchronization time and space complexity in the cases $g = \lambda x.x+1$ and $g = \lambda x.2x$. It shows that the trade-off presented in the previous section no longer applies. In the case $g = \lambda x.2x$, both time and space complexity contain exponential terms. A real-world device would quickly run out of memory. The SAP_g algorithm remains practical only if g is a slowly growing function. Comparing with Table 1, we observe that SAP_g achieves better performance in networks with finite diameter than in strongly centered networks. Choosing $g = \lambda x.x+1$, the time complexity is in $O\left(R\left(D^2 + R + \max_{i\in V} M_i(0)\right)\right)$ in the later case, compared to $O(D^2)$ in the earlier case. A similar overhead is added to space complexity. Overall, choosing $g = \lambda x.x+1$ seems to provide the best performances, as it is the "least inflationary" function.

6 Concluding remarks

We presented the SAP_g algorithm that solves the mod P-synchronization problem in any dynamic network that either has a finite diameter or is strongly centered. Both assumptions correspond to connectivity properties that have to hold in bounded periods of time. These results highlight the critical importance of timing bounds for the network to be connected enough and demonstrate how time may act as a healer.

The correctness proofs also provide bounds on stabilization time and space complexity of the SAP_g algorithm. The time bound and the space bound depend respectively on the functions g^* and g, leading thus to a time-space trade-off for choosing g: the more inflationary g is, the lower the time complexity is, and the higher its space complexity is. Moreover, these results show how the initial knowledge on a bound on the diameter allows for more efficient solutions in terms of both time and space.

The scenario in Section 4.2 shows that the SAP_g algorithm does not work anymore when relaxing the assumption of a strongly centered network into the one of a non-empty center. A natural question then arises about the possibility of designing a finite-state self-stabilizing algorithm that provides nodes with clocks modulo P which eventually synchronize in a dynamic network with at least one central node.

References

1 Karine Altisen, Stéphane Devismes, Swan Dubois, and Franck Petit. Introduction to distributed self-stabilizing algorithms. *Synthesis Lectures on Distributed Computing Theory*, 8(1):1–165, 2019.

2 D. Angluin, J. Aspnes, D. Eisenstat, and E. Ruppert. The computational power of population protocols. *Distributed Computing*, 20(4):279–304, 2007.

3 A. Arora, S. Dolev, and M. Gouda. Maintaining digital clocks in step. *Parallel Processing Letters*, 1:11–18, 1991.

4 P. Bastide, G. Giakkoupis, and H. Saribekyan. Self-stabilizing clock synchronization with 1-bit messages. In *Proceedings of the 2021 ACM-SIAM Symposium on Discrete Algorithms, SODA, 2021*, pages 2154–2173. SIAM, 2021.

5 M. Ben-Or. Another advantage of free choice: Completely asynchronous agreement protocols. In *Proceedings of the Second Symposium on Principles of Distributed Computing*, pages 27–30, 1983.

6 P. Bernstein, V. Hadzilacos, and N. Goodman. *Concurrency Control and Recovery in Database Systems*. Addison-Wesley, 1987.

7 L. Boczkowski, A. Korman, and E. Natale. Minimizing message size in stochastic communication patterns: fast self-stabilizing protocols with 3 bits. *Distributed Comput.*, 32(3):173–191, 2019.

8 P. Boldi and S. Vigna. Universal dynamic synchronous self-stabilization. *Distributed Computing*, 15(3):137–153, 2002.

9 C. Boulinier, F. Petit, and V. Villain. Synchronous vs. asynchronous unison. *Algorithmica*, 51(1):61–80, 2008.

10 B. Charron-Bost. Geometric bounds for convergence rates of averaging algorithms. *Information and Computation*, 2022. To appear, available at `arXiv:2007.04837`.

11 B. Charron-Bost and A. Schiper. The Heard-Of model: computing in distributed systems with benign faults. *Distributed Computing*, 22(1):49–71, 2009.

12 A. Cornejo and F. Kuhn. Deploying wireless networks with beeps. In *Distributed Computing, 24th International Symposium, DISC 2010, Cambridge, MA, USA, September 13-15, 2010. Proceedings*, volume 6343 of *Lecture Notes in Computer Science*, pages 148–162. Springer, 2010.

13 S. Dolev. Possible and impossible self-stabilizing digital clock synchronization in general graphs. *Real Time Syst.*, 12(1):95–107, 1997.

14 S. Dolev and J. L. Welch. Self-stabilizing clock synchronization in the presence of byzantine faults. *J. ACM*, 51(5):780–799, 2004.

15 C. Dwork, N. Lynch, and L. Stockmeyer. Consensus in the presence of partial synchrony. *Journal of the ACM*, 35(2):288–323, 1988.

16 S. Even and S. Rajsbaum. Unison, canon, and sluggish clocks in networks controlled by a synchronizer. *Math. Syst. Theory*, 28(5):421–435, 1995.

17 M. Feldmann, A. Khazraei, and C. Scheideler. Time- and space-optimal discrete clock synchronization in the beeping model. In *SPAA '20: 32nd ACM Symposium on Parallelism in Algorithms and Architectures, USA, 2020*, pages 223–233. ACM, 2020.

18 M. Gouda and T. Herman. Stabilizing unison. *Inf. Process. Lett.*, 35(4):171–175, 1990.

19 T. Herman and S. Ghosh. Stabilizing phase-clocks. *Inf. Process. Lett.*, 54(5):259–265, 1995.

20 A. Jadbabaie. Natural algorithms in a networked world: technical perspective. *Commun. ACM*, 55(12):100, 2012.

21 Ronald Kempe, Joseph Y. Dobra, and Moshe Y. Gehrke. Gossip-based computation of aggregate information. In *Proceeding of the 44th IEEE Symposium on Foundations of Computer Science, FOCS*, pages 482–491, Cambridge, MA, USA, 2003.

22 L. Lamport. The part-time parliament. *ACM Transactions on Computer Systems*, 16(2):133–169, 1998.

23 S. H. Strogatz. From kuramoto to crawford: exploring the onset of synchronization in populations of coupled oscillators. *Physica D*, 143(1-4):1–20, 2000.