Parameterized verification of synchronization in constrained reconfigurable broadcast networks

A.R. Balasubramanian, Nathalie Bertrand, Nicolas Markey

Parameterized verification of synchronization in constrained reconfigurable broadcast networks*

Balasubramanian A.R.[1], Nathalie Bertrand[2], and Nicolas Markey[2]

[1] Chennai Mathematical Institute – Chennai (India)
[2] Univ. Rennes, Inria, CNRS, IRISA – Rennes (France)

Abstract. Reconfigurable broadcast networks provide a convenient formalism for modelling and reasoning about networks of mobile agents broadcasting messages to other agents following some (evolving) communication topology. The parameterized verification of such models aims at checking whether a given property holds irrespective of the initial configuration (number of agents, initial states and initial communication topology). We focus here on the synchronization property, asking whether all agents converge to a set of target states after some execution. This problem is known to be decidable in polynomial time when no constraints are imposed on the evolution of the communication topology (while it is undecidable for static broadcast networks).
In this paper we investigate how various constraints on reconfigurations affect the decidability and complexity of the synchronization problem. In particular, we show that when bounding the number of reconfigured links between two communications steps by a constant, synchronization becomes undecidable; on the other hand, synchronization remains decidable in PTIME when the bound grows with the number of agents.

1 Introduction

There are numerous application domains for networks formed of an arbitrary number of anonymous agents executing the same code: prominent examples are distributed algorithms, communication protocols, cache-coherence protocols, and biological systems such as populations of cells or individuals, etc. The automated verification of such systems is challenging [15,12,8,3]: its aim is to validate at once all instances of the model, independently of the (parameterized) number of agents. Such a problem can be phrased in terms of infinite-state-system verification. Exploiting symmetries may lead to efficient algorithms for the verification of relevant properties [7].

Different means of interactions between agents can be considered in such networks, depending on the application domain. Typical examples are shared variables [13,10,4], *rendez-vous* [12], and broadcast communications [9,6]. In this paper, we target ad hoc networks [6], in which the agents can broadcast messages

* This work has been supported by the Indo-French research unit UMI Relax, and by ERC project EQualIS (308087).

simultaneously to all their neighbours, *i.e.*, to all the agents that are within their radio range. The number of agents and the communication topology are fixed once and for all at the beginning of the execution. Parameterized verification of broadcast networks checks if a specification is met independently of the number of agents and communication topology. It is usually simpler to reason about the dual problem of the existence of an initial configuration (consisting of a network size, an initial state for each agent, and a communication topology) from which some execution violates the given specification.

Several types of specifications have been considered in the literature. We focus here on coverability and synchronization: *does there exist an initial configuration from which some agent (resp. all agents at the same time) may reach a particular set of target states*. Both problems are undecidable; decidability of coverability can be regained by bounding the length of simple paths in the communication topology [6].

In the case of mobile ad hoc networks (MANETs), agents are mobile, so that the communication links (and thus the neighbourhood of each agent) may evolve over time. To reflect the mobility of agents, Delzanno *et al.* studied *reconfigurable* broadcast networks [6,5]. In such networks, the communication topology can change arbitrarily at any time. Perhaps surprisingly, this modification not only allows for a more faithful modelling of MANETs, but it also leads to decidability of both the coverability and the synchronization problems [6]. A probabilistic extension of reconfigurable broadcast networks has been studied in [1,2] to model randomized protocols.

A drawback of the semantics of reconfigurable broadcast networks is that they allow arbitrary changes at each reconfiguration. Such arbitrary reconfigurations may not be realistic, especially in settings where communications are frequent enough, and mobility is slow and not chaotic. In this paper, we limit the impact of reconfigurations in several ways, and study how those limitations affect the decidability and complexity of parameterized verification of synchronization.

More specifically, we restrict reconfigurations by limiting the number of changes in the communication graph, either by considering *global* constraints (on the total number of edges being modified), or by considering *local* constraints (on the number of updates affecting each individual node). We prove that synchronization is decidable when imposing constant local constraints, as well as when imposing global constraints depending (as a divergent function) on the number of agents. On the other hand, imposing a constant global bound makes synchronization undecidable. We recover decidability by bounding the maximal degree of each node by 1.

2 Broadcast networks with constrained reconfiguration

In this section, we first define reconfigurable broadcast networks; we then introduce several constraints on reconfigurations along executions, and investigate how they compare one to another and with unconstrained reconfigurations.

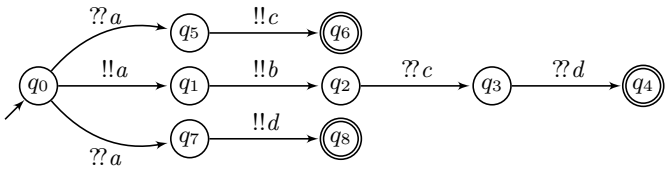

Fig. 1. Example of a broadcast protocol

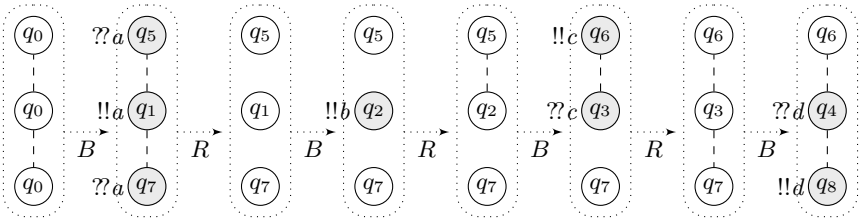

Fig. 2. Sample execution under reconfigurable semantics, synchronizing to $\{q_4, q_6, q_8\}$ (B-transitions are communications steps, R are reconfiguration steps).

2.1 Reconfigurable broadcast networks

Definition 1. *A* broadcast protocol *is a tuple* $\mathcal{P} = (Q, I, \Sigma, \Delta)$ *where* Q *is a finite set of control states;* $I \in Q$ *is the set of initial control states;* Σ *is a finite alphabet; and* $\Delta \subseteq (Q \times \{!!a, ??a \mid a \in \Sigma\} \times Q)$ *is the transition relation.*

A (reconfigurable) broadcast network is a system made of several copies of a single broadcast protocol $\mathcal{P}$. Configurations of such a network are undirected graphs whose each node is labelled with a state of $\mathcal{P}$. Transitions between configurations can either be reconfigurations of the communication topology (*i.e.*, changes in the edges of the graph), or a communication via broadcast of a message (*i.e.*, changes in the labelling of the graph). Figures 1 and 2 respectively display an example of a broadcast protocol and of an execution of a network made of three copies of that protocol.

Formally, we first define undirected labelled graphs. Given a set $\mathcal{L}$ of labels, an $\mathcal{L}$-*graph* is an undirected graph $\mathsf{G} = (\mathsf{N}, \mathsf{E}, \mathsf{L})$ where N is a finite set of nodes; $\mathsf{E} \subseteq \mathcal{P}_2(\mathsf{N})$[3] (notice in particular that such a graph has no self-loops); finally, $\mathsf{L} \colon \mathsf{N} \to \mathcal{L}$ is the labelling function. We let $\mathcal{G}_\mathcal{L}$ denote the (infinite) set of $\mathcal{L}$-labelled graphs. Given a graph $\mathsf{G} \in \mathcal{G}_\mathcal{L}$, we write $\mathsf{n} \sim \mathsf{n}'$ whenever $\{\mathsf{n}, \mathsf{n}'\} \in \mathsf{E}$ and we let $\mathsf{Neigh}_\mathsf{G}(\mathsf{n}) = \{\mathsf{n}' \mid \mathsf{n} \sim \mathsf{n}'\}$ be the neighbourhood of n, *i.e.* the set of nodes adjacent to n. For a label ℓ, we denote by $|\mathsf{G}|_\ell$ the number of nodes in G labelled by ℓ. Finally $\mathsf{L}(\mathsf{G})$ denotes the set of labels appearing in nodes of G.

The semantics of a reconfigurable broadcast network based on broadcast protocol $\mathcal{P}$ is an infinite-state transition system $\mathcal{T}(\mathcal{P})$. The configurations of

[3] For a finite set S and $1 \leq k \leq |S|$, we let $\mathcal{P}_k(S) = \{T \subseteq S \mid |T| = k\}$.

$\mathcal{T}(\mathcal{P})$ are Q-labelled graphs. Intuitively, each node of such a graph runs protocol $\mathcal{P}$, and may send/receive messages to/from its neighbours. A configuration $(\mathsf{N}, \mathsf{E}, \mathsf{L})$ is said *initial* if $\mathsf{L}(\mathsf{N}) \subseteq I$. From a configuration $\mathsf{G} = (\mathsf{N}, \mathsf{E}, \mathsf{L})$, two types of steps are possible. More precisely, there is a step from $(\mathsf{N}, \mathsf{E}, \mathsf{L})$ to $(\mathsf{N}', \mathsf{E}', \mathsf{L}')$ if one of the following two conditions holds:

(reconfiguration step) $\mathsf{N}' = \mathsf{N}$ and $\mathsf{L}' = \mathsf{L}$: a reconfiguration step does not change the set of nodes and their labels, but may change the edges arbitrarily;

(communication step) $\mathsf{N}' = \mathsf{N}$, $\mathsf{E}' = \mathsf{E}$, and there exists $\mathsf{n} \in \mathsf{N}$ and $a \in \Sigma$ such that $(\mathsf{L}(\mathsf{n}), !!a, \mathsf{L}'(\mathsf{n})) \in \Delta$, and for every n', if $\mathsf{n}' \in \mathsf{Neigh}_{\mathsf{G}}(\mathsf{n})$, then $(\mathsf{L}(\mathsf{n}'), ?!a, \mathsf{L}'(\mathsf{n}')) \in \Delta$, otherwise $\mathsf{L}'(\mathsf{n}') = \mathsf{L}(\mathsf{n}')$: a communication step reflects how nodes evolve when one of them broadcasts a message to its neighbours.

An *execution* of the reconfigurable broadcast network is a sequence $\rho = (\mathsf{G}_i)_{0 \leq i \leq r}$ of configurations such that for any $i < r$, there is a step from G_i to G_{i+1} and ρ strictly alternates communication and reconfiguration steps (the latter possibly being trivial). An execution is *initial* if it starts from an initial configuration.

An important ingredient that we heavily use in the sequel is *juxtaposition* of configurations and *shuffling* of executions. The juxtaposition of two configurations $\mathsf{G} = (\mathsf{N}, \mathsf{E}, \mathsf{L})$ and $\mathsf{G}' = (\mathsf{N}', \mathsf{E}', \mathsf{L}')$ is the configuration $\mathsf{G} \oplus \mathsf{G}' = (\mathsf{N} \uplus \mathsf{N}', \mathsf{E} \uplus \mathsf{E}', \mathsf{L}_{\oplus})$, in which $\mathsf{L}_{\oplus}$ extends both L and L': $\mathsf{L}_{\oplus}(\mathsf{n}) = \mathsf{L}(\mathsf{n})$ if $\mathsf{n} \in \mathsf{N}$ and $\mathsf{L}_{\oplus}(\mathsf{n}) = \mathsf{L}'(\mathsf{n})$ if $\mathsf{n} \in \mathsf{N}'$. We write G^2 for the juxtaposition of G with itself, and, inductively, G^N for the juxtaposition of G^{N-1} with G. A shuffle of two executions $\rho = (\mathsf{G}_i)_{0 \leq i \leq r}$ and $\rho' = (\mathsf{G}'_j)_{0 \leq j \leq r'}$ is an execution $\rho_{\oplus}$ from $\mathsf{G}_0 \oplus \mathsf{G}'_0$ to $\mathsf{G}_r \oplus \mathsf{G}'_{r'}$ obtained by interleaving ρ and ρ'. Note that a reconfiguration step in $\rho_{\oplus}$ may be composed of reconfigurations from both ρ and ρ'. We write $\rho \oplus \rho'$ for the set of shuffle executions obtained from ρ and ρ'.

Natural decision problems for reconfigurable broadcast networks include checking whether some node may reach a target state, or whether all nodes may synchronize to a set of target states. More precisely, given a broadcast protocol $\mathcal{P}$ and a subset $F \subseteq Q$, the *coverability problem* asks whether there exists an initial execution ρ that visits a configuration G with $\mathsf{L}(\mathsf{G}) \cap F \neq \emptyset$, and the *synchronization problem* asks whether there exists an initial execution ρ that visits a configuration G with $\mathsf{L}(\mathsf{G}) \subseteq F$. For unconstrained reconfigurations, we have:

Theorem 2 ([6,5,11]). *The coverability and synchronization problems are decidable in* PTIME *for reconfigurable broadcast protocols.*

Remark 1. The synchronization problem was proven decidable in [6], and PTIME membership was given in [11, p. 41]. The algorithm consists in computing the set of states of $\mathcal{P}$ that are both reachable (*i.e.*, coverable) from an initial configuration and co-reachable from a target configuration. This can be performed by applying iteratively the algorithm of [5] for computing the set of reachable states (with reversed transitions for computing co-reachable states).

Example 1. Consider the broadcast protocol of Fig. 1 with $I = \{q_0\}$. From each state, unspecified message receptions lead to an (omitted) sink state; this way, each broadcast message triggers a transition in all the neighbouring copies.

For that broadcast protocol, one easily sees that it is possible to synchronize to the set $\{q_4, q_6, q_8\}$. Moreover, three copies are needed and sufficient for that objective, as witnessed by the execution of Fig. 2. The initial configuration has three copies and two edges. If the central node broadcasts a, the other two nodes receive, one proceeding to q_5 and the other to q_7. Then, we assume the communication topology is emptied before the same node broadcasts b, moving to q_2. Finally the node in q_5 connects to the one in q_2 to communicate on c and then disconnects, followed by a similar communication on d initiated by the node in q_7.

2.2 Natural constraints for reconfiguration

Allowing arbitrary changes in the network topology may look unrealistic. In order to address this issue, we introduce several ways of bounding the number of reconfigurations after each communication step. For this, we consider the following natural pseudometric between graphs, which for simplicity we call *distance*.

Definition 3. *Let* $\mathsf{G} = (\mathsf{N}, \mathsf{E}, \mathsf{L})$ *and* $\mathsf{G}' = (\mathsf{N}', \mathsf{E}', \mathsf{L}')$ *be two* $\mathcal{L}$*-labelled graphs. The distance between* G *and* G' *is defined as*

$$\mathsf{dist}(\mathsf{G}, \mathsf{G}') = |E \cup E' \setminus (E \cap E')|$$

when $\mathsf{N} = \mathsf{N}'$ *and* $\mathsf{L} = \mathsf{L}'$*, and* $\mathsf{dist}(\mathsf{G}, \mathsf{G}') = 0$ *otherwise.*

Setting the "distance" to 0 for two graphs that do not agree on the set of nodes or on the labelling function might seem strange at first. This choice is motivated by the definition of constraints on executions (see below) and of the number of reconfigurations along an execution (see Section 2.3). Other distances may be of interest in this context; in particular, for a fixed node $\mathsf{n} \in \mathsf{N}$, we let $\mathsf{dist}_{\mathsf{n}}(\mathsf{G}, \mathsf{G}')$ be the number of edges involving node n in the symmetric difference of E and E' (still assuming $\mathsf{N} = \mathsf{N}'$ and $\mathsf{L} = \mathsf{L}'$).

Constant number of reconfigurations per step. A first natural constraint on reconfiguration consists in bounding the number of changes in a reconfiguration step by a constant number. Recall that along executions, communication and reconfiguration steps strictly alternate.

Definition 4. *Let* $k \in \mathbb{N}$*. An execution* $\rho = (\mathsf{G}_i)_{0 \leq i \leq r}$ *of a reconfigurable broadcast network is* k-constrained *if for every index* $i < r$*, it holds* $\mathsf{dist}(\mathsf{G}_i, \mathsf{G}_{i+1}) \leq k$*.*

Example 1 (Contd). For the synchronization problem, bounding the number of reconfigurations makes a difference. The sample execution from Fig. 2 is not 1-constrained, and actually no 1-constrained executions of that broadcast protocol can synchronize to $\{q_4, q_5, q_6\}$. This can be shown by exhibiting and proving an invariant on the reachable configurations (see Lemma 10).

Beyond constant number of reconfigurations per step. Bounding the number of reconfigurations per step by a constant is somewhat restrictive, especially when this constant does not depend on the size of the network. We introduce other kinds of constraints here, for instance by bounding the number of reconfigurations by k *on average* along the execution, or by having a bound that depends on the number of nodes executing the protocol.

For a finite execution $\rho = (\mathsf{G}_i)_{0 \leq i \leq r}$ of a reconfigurable broadcast network, we write $\texttt{nb\_comm}(\rho)$ for the number of communication steps along ρ (notice that $\lfloor r/2 \rfloor \leq \texttt{nb\_comm}(\rho) \leq \lceil r/2 \rceil$ since we require strict alternation between reconfiguration and communication steps), and $\texttt{nb\_reconf}(\rho)$ for the total number of edge reconfigurations in ρ, that is $\texttt{nb\_reconf}(\rho) = \sum_{i=0}^{r-2} \mathsf{dist}(\mathsf{G}_i, \mathsf{G}_{i+1})$.

Definition 5. *Let $k \in \mathbb{N}$. An execution ρ of a reconfigurable broadcast network is said* k-balanced *if it starts and ends with a communication step, and satisfies* $\texttt{nb\_reconf}(\rho) \leq k \cdot (\texttt{nb\_comm}(\rho) - 1)$.

This indeed captures our intuition that along a k-balanced execution, reconfigurations *on average* update less than k links.

Finally, we will also consider two relevant ways to constrain reconfigurations depending on the size of the network: first locally, bounding the number of reconfigurations *per node* by a constant; second globally, bounding the total number of reconfigurations by a function of the number of nodes.

We first bound reconfigurations locally.

Definition 6. *Let $k \in \mathbb{N}$. An execution $\rho = (\mathsf{G}_i)_{0 \leq i \leq r}$ of a reconfigurable broadcast network is* k-locally-constrained*, if, for every node n and for every index $i < r$, $\mathsf{dist}_{\mathsf{n}}(\mathsf{G}_i, \mathsf{G}_{i+1}) \leq k$.*

One may also bound the number of reconfigurations globally using bounding functions, that depend on the number of nodes in the network:

Definition 7. *Let $f \colon \mathbb{N} \to \mathbb{N}$ be a function. An execution $\rho = (\mathsf{G}_i)_{0 \leq i \leq r}$ of a reconfigurable broadcast network is* f-constrained*, if, writing n for the number of nodes in G_0, it holds $\mathsf{dist}(\mathsf{G}_i, \mathsf{G}_{i+1}) \leq f(n)$ for any $i < r$.*

Notice that if f is the constant function $n \in \mathbb{N} \mapsto k$ for some $k \in \mathbb{N}$, f-constrained executions coincide with k-constrained ones, so that our terminology is non-ambiguous. Other natural bounding functions are non-decreasing and *diverging*. This way, the number of possible reconfigurations tends to infinity when the network size grows, *i.e.* $\forall n.\ \exists k.\ f(k) \geq n$.

Remark 2. Coverability under constrained reconfigurations is easily observed to be equivalent to coverability with unconstrained reconfigurations: from an unconstrained execution, we can simply juxtapose extra copies of the protocol, which would perform extra communication steps so as to satisfy the constraint. When dealing with synchronization, this technique does not work since the extra copies would also have to synchronize to a target state. As a consequence, we only focus on synchronization in the rest of this paper.

2.3 Classification of constraints

In this section, we compare our restrictions. We prove that, for the synchronization problem, k-locally-constrained and f-constrained reconfigurations, for diverging functions f, are equivalent to unconstrained reconfigurations. On the other hand, we prove that k-constrained reconfigurations are equivalent to k-balanced reconfigurations, and do not coincide with unconstrained reconfigurations.

Equivalence between unconstrained and locally-constrained reconfigurations.

Lemma 8. *Let $\mathcal{P}$ be a broadcast protocol, $F \subseteq Q$ be a target set, and f be a non-decreasing diverging function. If the reconfigurable broadcast network defined by $\mathcal{P}$ has an initial execution synchronizing in F, then it has an f-constrained initial execution synchronizing in F.*

Proof. We first prove the lemma for the identity function Id. More precisely, we prove that for an execution $\rho = (\mathsf{G}_i)_{0 \leq i \leq n}$, of the reconfigurable broadcast network, there exists a Id-constrained execution $\rho' = (\mathsf{G}'_j)_{0 \leq j \leq m}$, whose last transition (if any) is a communication step, and such that for any control state q, $|\mathsf{G}_n|_q = |\mathsf{G}'_m|_q$. We reason by induction on the length of the execution. The claim is obvious for $n = 0$. Suppose the property is true for all naturals less than or equal to some $n \in \mathbb{N}$, and consider an execution $\rho = (\mathsf{G}_i)_{0 \leq i \leq n+1}$. The induction hypothesis ensures that there is an f-constrained execution $\rho' = (\mathsf{G}'_j)_{0 \leq j \leq m}$ with $|\mathsf{G}_n|_q = |\mathsf{G}'_m|_q$ for all q. If the last transition from G_n to G_{n+1} in ρ is a reconfiguration step, then the execution ρ' witnesses our claim. Otherwise, the transition from G_n to G_{n+1} is a communication step, involving a broadcasting node n of G_n labelled with q, and receiving nodes n_1 to n_r of G_n, respectively labelled with q_1 to q_r. By hypothesis, G'_m also contains a node n' labelled with q and r nodes n'_1 to n'_r, labelled with q_1 to q_r. We then add two steps after G'_m in ρ': we first reconfigure the graph so that $\mathsf{Neigh}_{\mathsf{G}'_{m+1}}(\mathsf{n}') = \{\mathsf{n}'_i \mid 0 \leq i \leq r\}$, which requires changing at most $|\mathsf{G}_0| - 1$ links, and then perform the same broadcast/receive transitions as between G_n and G_{n+1}.

For the general case of the lemma, suppose f is a non-decreasing diverging function. Further, let $\rho = (\mathsf{G}_i)_{0 \leq i \leq n}$ be an Id-constrained execution, and pick k such that $f(k \cdot |\mathsf{G}_0|) \geq |\mathsf{G}_0|$. Consider the initial configuration G_0^k, made of k copies of G_0, and the execution, denoted ρ^k, made of k copies of ρ running independently from each of the k copies of G_0 in G_0^k. Each reconfiguration step involves at most $|\mathsf{G}_0|$ links, so that ρ^k is f-constrained. □

Lemma 9. *Let $\mathcal{P}$ be a broadcast protocol with $F \subseteq Q$ a target set. If the reconfigurable broadcast network defined by $\mathcal{P}$ has an initial execution synchronizing in F, then it has a 1-locally-constrained initial execution synchronizing in F.*

k-constrained and k-balanced reconfigurations. We prove here that k-constrained and k-balanced reconfigurations are equivalent w.r.t. synchronization, and that they are strictly stronger than our other restrictions. We begin with the latter:

Lemma 10. *There exists a broadcast protocol $\mathcal{P}$ and a set $F \subseteq Q$ of target states for which synchronization is possible from some initial configuration when unconstrained reconfigurations are allowed, and impossible, from every initial configuration when only* 1*-constrained reconfigurations are allowed.*

A protocol with this property is the one from Example 1, for which we exhibited a 2-constrained synchronizing execution. It can be proved that no 1-constrained synchronizing executions exist for this protocol, whatever the number of copies. We now prove the main result of this section:

Theorem 11. *Let $\mathcal{P}$ be a broadcast protocol and $F \subseteq Q$. There exists a k-constrained initial execution synchronizing in F if, and only if, there exists a k-balanced initial execution synchronizing in F.*

Proof. The left-to-right implication is simple: if there is a k-constrained initial execution synchronizing in F, w.l.o.g. we can assume that this execution starts and ends with a communication step; moreover, each reconfiguration step contains at most k edge reconfigurations, so that the witness execution is k-balanced.

Let $\rho = (\mathsf{G}_i)_{0 \leq i \leq n}$ be a k-balanced execution synchronizing in F and starting and ending with communication steps (hence n is odd). We define the potential $(p_i)_{0 \leq i \leq n}$ of ρ as the sequence of $n+1$ integers obtained as follows:

- $p_0 = 0$;
- $p_{2i+1} = p_{2i} + k$ for $i \leq (n-1)/2$ (this corresponds to a communication step);
- $p_{2i+2} = p_{2i+1} - \mathsf{dist}(\mathsf{G}_{2i+1}, \mathsf{G}_{2i+2})$ for $i \leq (n-1)/2 - 1$ (reconfiguration step).

That ρ is k-balanced translates as $p_{n-1} \geq 0$: the sequence $(p_i)_{0 \leq i \leq n}$ stores the value of $k \cdot \mathtt{nb\_comm}(\rho_{\leq i}) - \mathtt{nb\_reconf}(\rho_{\leq i})$ for each prefix $\rho_{\leq i}$ of ρ; being k-balanced means that $p_n \geq k$, and since the last step is a communication step, this in turn means $p_{n-1} \geq 0$. On the other hand, in order to be k-constrained, it is necessary (but not sufficient) to have $p_i \geq 0$ for all $0 \leq i \leq n$.

We build a k-constrained execution by shuffling several copies of ρ. We actually begin with the case where $k = 1$, and then extend the proof to any k. We first compute how many copies we need. For this, we split ρ into several phases, based on the potential $(p_i)_{0 \leq i \leq n}$ defined above. A phase is a maximal segment of $\rho_{\leq n-1}$ (the prefix of ρ obtained by dropping the last (communication) step) along which the sign of the potential is constant (or zero): graphs G_i and G_j are in the same phase if, and only if, for all $i \leq l \leq l' \leq j$, it holds $p_l \cdot p_{l'} \geq 0$. We decompose ρ as the concatenation of phases $(\rho_j)_{0 \leq j \leq m}$; since ρ is k-balanced, m is even, and ρ_0, ρ_m, and all even-numbered phases are *non-negative* phases (*i.e.*, the potential is non-negative along those executions), while all odd-numbered executions are *non-positive* phases. Also, all phases end with potential zero, except possibly for ρ_m. See Fig. 3 for an example of a decomposition into phases.

Lemma 12. *For any phase $\rho_i = \mathsf{G}_{b_i} \cdots \mathsf{G}_{e_i}$ of a* 1*-balanced execution $\rho = \mathsf{G}_0 \cdots \mathsf{G}_n$, there exists $\kappa_i \leq (e_i - b_i)/2$ such that for any $N \in \mathbb{N}$, there exists a* 1*-constrained execution from $\mathsf{G}_0^{\kappa_i} \oplus \mathsf{G}_{b_i}^N$ to $\mathsf{G}_1^{\kappa_i} \oplus \mathsf{G}_{e_i}^N$.*

Proof. We handle non-negative and non-positive phases separately. In a non-negative phase, we name *repeated reconfiguration step* any reconfiguration step that immediately follows another (possibly from the previous phase) reconfiguration step (so that if there are four consecutive reconfiguration steps, the last three are said repeated); similarly, we name *repeated communication step* any communication step that is immediately followed (possibly in the next phase) by another communication step (hence the first three of fours consecutive communication steps are repeated).

We first claim that any non-negative phase contains at least as many repeated communication steps as it contains repeated reconfiguration steps. Indeed, any non-repeated communication step in a non-negative phase is necessarily followed by a non-repeated reconfiguration step, and conversely, and non-negative phases have at least as many communication steps as they have reconfiguration steps.

As a consequence, we can number all repeated reconfiguration steps from 1 (earliest) to κ_i (latest), for some κ_i, and similarly for repeated communication steps. Clearly enough, in a non-negative phase, for any $1 \leq j \leq \kappa_i$, the repeated communication step numbered j occurs before the repeated reconfiguration step carrying the same number.

We now build our 1-constrained execution from $\mathsf{G}_0^{\kappa_i} \oplus \mathsf{G}_{b_i}^N$ to $\mathsf{G}_1^{\kappa_i} \oplus \mathsf{G}_{e_i}^N$. We begin with a first part, where only the components starting from G_{b_i} move:

- the first copy starting in G_{b_i} follows the execution ρ_i until reaching the repeated reconfiguration step number 1. That reconfiguration step cannot be performed immediately as it follows another reconfiguration step. Notice that during this stage, this copy has taken at least one repeated communication step, numbered 1;
- the second copy then follows ρ_i until reaching its first repeated communication step (which must occur before the first repeated reconfiguration step). It takes this communication step, then allowing the first copy to perform its first repeated reconfiguration step;
- this simulation continues, each time having the $l+1$-st copy of the system taking its j-th repeated communication step in order to allow the l-th copy to perform its j-th repeated reconfiguration step. Non-repeated steps can always be performed individually by each single copy. Also, the first copy may always take repeated communication steps not having a corresponding reconfiguration step, as in the first stage of this part.

Notice that the number of copies involved in this process is arbitrary. The process lasts as long as some copies may advance within phase ρ_i. Hence, when the process stops, all copies of the original system either have reached the end of ρ_i, or are stopped before a repeated reconfiguration step. For the copies in the latter situation, we use the copies starting from G_0. It remains to prove that having κ_i such copies is enough to make all processes reach the end of ρ_i.

For this, we first assume that the potential associated with ρ_i ends with value zero. This must be the case of all phases except the last one, which we handle after the general case. We first notice that in the execution we are currently building, any repeated communication step performed by any (but the very first)

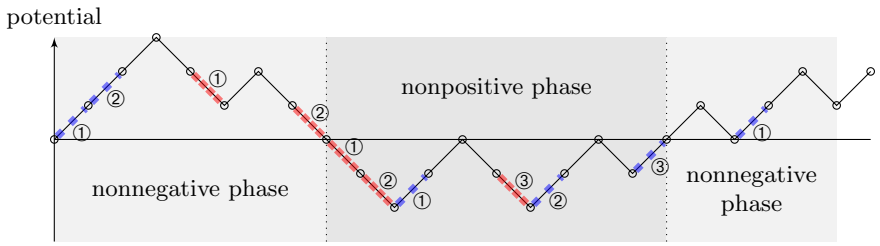

Fig. 3. Phases of a 1-balanced execution, and correspondence between repeated communication steps (loosely dotted blue steps) and repeated reconfiguration steps (densely dotted red steps)

copy that started from G_{b_i} is always followed by a repeated reconfiguration step. Similarly, non-repeated communication steps of any copy is followed by a non-repeated broadcast step of the same copy. As a consequence, the potential associated with the global execution we are currently building never exceeds the total number of repeated communication steps of performed by the first copy; hence it is bounded by κ_i, whatever the number N of copies involved. As a consequence, at most κ_i communication steps are sufficient in order to advance all copies that started from G_{b_i} to the end of ρ_i.

Finally, the case of the last phase ρ_m (possibly ending with positive potential) is easily handled, since it has more communication steps than reconfiguration steps.

The proof for non-positive phases is similar. □

Pick a 1-balanced execution $\rho = \mathsf{G}_0 \cdots \mathsf{G}_n$, and decompose it into phases $\rho_1 \cdots \rho_m$. For each phase ρ_i, we write κ_i for the total number of repeated reconfiguration steps, and we let $\kappa = \sum_{1 \leq i \leq m} \kappa_i$ for the total number of repeated reconfiguration steps along ρ. Notice that $\kappa \leq n/2$.

Lemma 13. *For every* 1*-balanced execution* $\rho = \mathsf{G}_0 \cdots \mathsf{G}_n$*, and for every* $N \in \mathbb{N}$*, there exists a* 1*-constrained execution from* $\mathsf{G}_1^N \oplus \mathsf{G}_{e_m}^{\kappa N}$ *to* $\mathsf{G}_n^{N+\kappa N}$*.*

Combining the above two lemmas, we obtain the following proposition, which refines the statement of the Theorem 11:

Proposition 14. *For every* 1*-balanced execution* $\rho = \mathsf{G}_0 \cdots \mathsf{G}_n$ *and every* $N \geq \kappa^2 + \kappa$*, there exists a* 1*-constrained execution from* G_0^N *to* G_n^N*.*

We finally extend this result to $k > 1$. In this case, splitting ρ into phases is not as convenient as when $k = 1$: indeed, a non-positive phase might not end with potential zero (because communication steps make the potential jump by k units). Lemma 12 would not hold in this case.

We circumvent this problem by first shuffling k copies of ρ in such a way that reconfigurations can be gathered into groups of size exactly k. This way, we can indeed split the resulting execution into non-negative and non-positive phases, always considering reconfigurations of size exactly k; we can then apply

the techniques above in order to build a synchronizing k-constrained execution. This completes our proof. □

3 Parameterized synchronization under reconfiguration constraints

3.1 Undecidability for k-constrained reconfiguration

Although synchronization is decidable in PTIME [6,11] for reconfigurable broadcast networks, the problem becomes undecidable when reconfigurations are k-constrained.

Theorem 15. *The synchronization problem is undecidable for reconfigurable broadcast networks under k-constrained reconfigurations.*

Proof. We prove this undecidability result for 1-constrained reconfigurations, by giving a reduction from the halting problem for Minsky machines [14]. We begin with some intuition. The state space of our protocol has two types of states:

- *control states* encode the control state of the 2-counter machine;
- *counter states* are used to model counter values: for each counter $c_j \in \{c_1, c_2\}$, we have a state zero_j and a state one_j. The value of counter c_j in the simulation will be encoded as the number of edges in the communication topology between the *control node* and *counter nodes* in state one_j; moreover, we will require that control nodes have no communication links with counter nodes in state zero_j.
  Incrementations and decrementations can then be performed by creating a link with a node in zero_j and sending this node to one_j, or sending a one_j-node to zero_j and removing the link.

In order to implement this, we have to take care of the facts that we may have several control nodes in our network, that we may have links between two control nodes or between two counter nodes, or that links between control nodes and counter nodes may appear or disappear at random. Intuitively, those problems will be handled as follows:

- we cannot avoid having several control nodes; instead, given a synchronizing execution of the broadcast protocol, we will select one control node and show that it encodes a correct execution of the 2-counter machine;
- in order to reach a synchronizing configuration, the selected control node will have to perform at least as many reconfiguration steps as broadcast steps. Because we consider 1-constrained runs, it will perform exactly the same number of reconfiguration steps as broadcast steps, so that no useless/ unexpected reconfigurations may take place during the simulation;
- control nodes will periodically run special broadcasts that would send any connected nodes (except nodes in state one_j) to a sink state, thus preventing synchronization. This way, we ensure that that particular control node is *clean*. Initially, we require that control nodes have no connections at all.

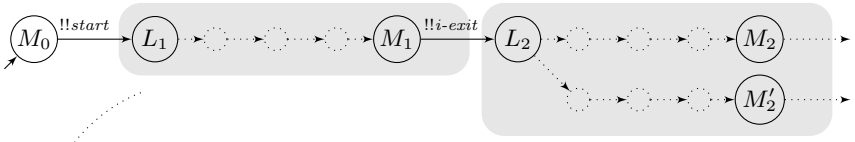

Fig. 4. Global view of the part of the protocol for control nodes

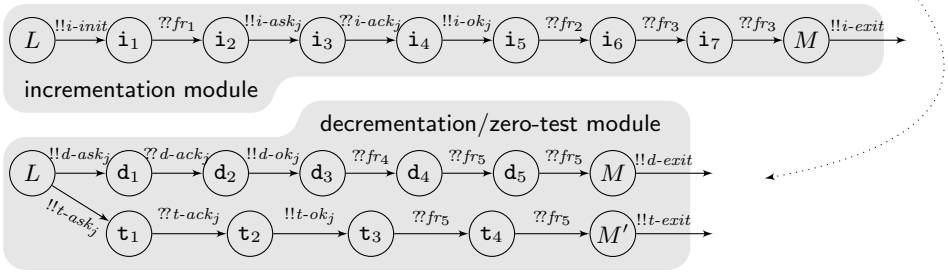

Fig. 5. Modules for simulating incrementation and decrementation/zero test

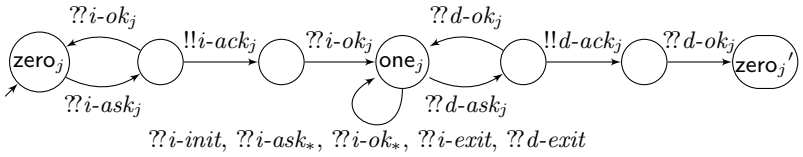

Fig. 6. The part of the protocol for counter nodes

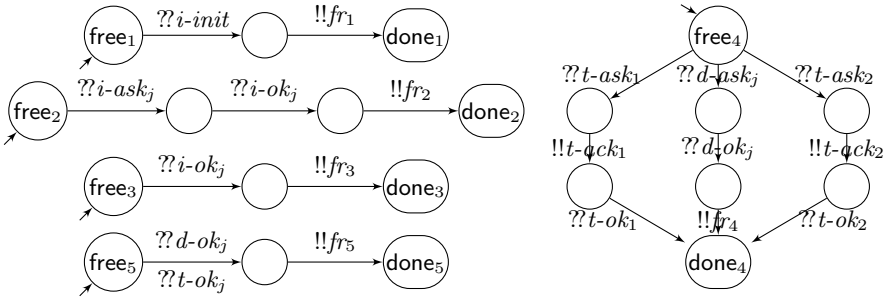

Fig. 7. Parts of the protocol for auxiliary nodes

We now present the detailed construction, depicted at Fig. 4 to 7. Each state of the protocol is actually able to synchronize with all the messages. Some transitions are not represented on the figures, to preserve readability: all nodes with no outgoing transitions (i.e., state L_{halt} corresponding to the halting state, as well as states $\mathsf{zero}_j{}'$ and done_i) actually carry a self-loop synchronizing on all messages; all other omitted transitions lead to a sink state, which is not part of the target set.

Let us explain the intended behaviour of the incrementation module of Fig. 5: when entering the module, our control node n in state L is linked to c_1 counter nodes in state one_1 and to c_2 counter nodes in state one_2; it has no other links. Moreover, all auxiliary nodes are either in state free_i or in state done_i. Running through the incrementation module from L will use one counter node m in state zero_j (which is used to effectively encode the increase of counter c_j) and four auxiliary nodes a_1 (initially in state free_1), a_2 (in state free_2), and a_3 and a'_3 (in state free_3).

The execution then runs as follows:

- a link is created between the control node n and the first auxiliary node a_1, followed by a message exchange $!!i\text{-}init$;
- a link is created between n and m, and node a_1 broadcasts $!!fr_1$;
- a link is created between n and a_2, and n broadcasts $!!i\text{-}ask_j$, which is received by both a_2 and m;
- a link is created between n and a_3; node m sends its acknowledgement $!!i\text{-}ack_j$ to n;
- a link is created between n and a'_3; node n sends $!!i\text{-}ok_j$, received by m, a_2, a_3 and a'_3;
- the link between n and a_1 is removed, and a_2 sends $!!fr_2$;
- the link between n and a_2 is removed, and a_3 sends $!!fr_3$;
- the link between n and a_3 is removed, and a'_3 sends $!!fr_3$;
- finally, the link between n and a'_3 is removed, and n sends $!!i\text{-}exit$.

After this sequence of steps, node n has an extra link to a counter node in state one_j, which indeed corresponds to incrementing counter c_j. Moreover, no nodes have been left in an intermediary state. A similar analysis can be done for the second module, which implements the zero-test and decrementation. This way, we can prove that if the two-counter machine has a halting computation, then there is an initial configuration of our broadcast protocol from which there is an execution synchronizing in the set F formed of the halting control state and states one_j, $\mathsf{zero}_j{}'$ and done_i.

It now remains to prove the other direction. More precisely, we prove that from a 1-constrained synchronizing execution of the protocol, we can extract a synchronizing execution in some normal form, from which we derive a halting execution of the two-counter machine.

Fix a 1-constrained synchronizing execution of the broadcast network. First notice that when a control node n reaches some state L (the first node of an incrementation or decrementation module), it may only be linked to counter nodes in state one_j: this is because states L can only be reached by sending $!!i\text{-}exit$, $!!d\text{-}exit$, $!!t\text{-}exit$, or $!!start$. The former two cases may only synchronize with counter nodes in state one_j; in the other two cases, node n may be linked to no other node. Hence, for a control node n to traverse an incrementation module, it must get links to four auxiliary nodes (in order to receive the four fr messages), those four links must be removed (to avoid reaching the sink state), and an extra link has to be created in order to receive message $i\text{-}ack_j$. In total, traversing

an incrementation module takes nine communication steps and at least nine reconfiguration steps. Similarly, traversing a decrementation module via any of the two branches takes at least as many reconfiguration steps as communication steps. In the end, taking into account the initial !!*start* communication step, if a control node n is involved in B_{n} communication steps, it must be involved in at least $B_{\mathsf{n}} - 1$ reconfiguration steps.

Assume that every control node n is involved in at least B_{n} reconfiguration steps: then we would have at least as many reconfiguration steps as communication steps, which in a 1-constrained execution is impossible. Hence there must be a control node n_0 performing B_{n_0} communication steps and exactly $B_{\mathsf{n}_0} - 1$ reconfiguration steps. As a consequence, when traversing an incrementation module, node n_0 indeed gets connected to exactly one new counter node, which indeed must be in state one_j when n_0 reaches the first state of the next module. Similarly, traversing a decrementation/zero-test module indeed performs the expected changes. It follows that the sequence of steps involving node n_0 encodes a halting execution of the two-counter machines. □

The 1-constrained executions in the proof of Theorem 15 have the additional property that all graphs describing configurations are 2-bounded-path configurations. For $K \in \mathbb{N}$ a configuration G is a *K-bounded-path configuration* if the length of all simple paths in G is bounded by K. Note that a constant bound on the length of simple paths implies that the diameter (*i.e.* the length of the longest shortest path between any pair of vertices) is itself bounded. The synchronization problem was proved to be undecidable for broadcast networks *without reconfiguration* when restricting to K-bounded-path configurations [6]. In comparison, for reconfigurable broadcast networks under k-constrained reconfigurations, the undecidability result stated in Theorem 15 can be strengthened into:

Corollary 16. *The synchronization problem is undecidable for reconfigurable broadcast networks under k-constrained reconfigurations when restricted either to bounded-path configurations, or to bounded-diameter configurations.*

3.2 Decidability results

f-constrained and k-locally-constrained reconfigurations. From the equivalence (w.r.t. synchronization) of k-locally-constrained, f-constrained and unconstrained executions (Lemmas 9 and 8), and thanks to Theorem 2, we immediately get:

Corollary 17. *Let $k \in \mathbb{N}$ and $f : \mathbb{N} \to \mathbb{N}$ be a non-decreasing diverging function. The synchronization problem for reconfigurable broadcast networks under k-locally-constrained (resp. f-constrained) reconfigurations is decidable in* PTIME.

Bounded degree topology. We now return to k-constrained reconfigurations, and explore restrictions that allow one to recover decidability of the synchronization problem. We further restrict k-constrained reconfigurations by requiring that the degree of nodes remains bounded, by 1; in other terms, communications correspond to *rendez-vous* between the broadcasting node and its single neighbour.

Theorem 18. *The synchronization problem is decidable for reconfigurable broadcast networks under k-constrained reconfiguration when restricted to 1-bounded-degree topologies.*

Sketch of proof. The proof consists in transforming the synchronization problem above into a reachability problem for some Petri net. The Petri net has two kinds of places (plus a few auxiliary places): one place for each state of the protocol, representing isolated nodes (*i.e.*, nodes having no neighbours), and one place for each pair of states of the protocol, representing pairs of connected nodes. Since we restrict to degree-1 topologies, any node of the network is in one of those two configurations. Places representing isolated nodes are simply called *isolated places* in the sequel, while places corresponding to pairs of connects nodes are called *connected places*.

An initialization phase stores tokens in the places described above, so as to represent the initial configuration. In a second phase, the Petri net simulates an execution of the reconfigurable broadcast network: communication steps and (k-constrained) reconfiguration steps are easily encoded as transitions of this Petri net: communication steps correspond to moving tokens from one place to the place obtained by updating the states as prescribed by the transitions of the broadcast protocol. Atomic reconfigurations may create or remove links, either consuming two tokens in isolated places and adding a token in the corresponding connected place, or the other way around. We use k auxiliary places in order to count the number of atomic reconfigurations, in order to enforce the k-constraint.

Finally, the Petri net may enter a terminal phase, where it checks synchronization by absorbing all tokens that lie in (isolated or connected) places corresponding to target states. In the end, the simulated execution has been synchronizing if, and only if, no tokens remain in any of the main states. □

4 Conclusion

Restricting reconfigurations in reconfigurable broadcast networks is natural to better reflect mobility when communications are frequent enough and the movement of nodes is not chaotic. In this paper, we studied how constraints on the number of reconfigurations (at each step and for each node, at each step and globally, or along an execution) change the semantics of networks, in particular with respect to the synchronization problem, and affect its decidability. Our main results are the equivalence of k-constrained and k-balanced semantics, the undecidability of synchronization under k-constrained reconfigurations, and its decidability when restricting to 1-bounded-degree topologies.

As future work, we propose to investigate, beyond the coverability and synchronization problems, richer objectives such as cardinality reachability problems as in [5]. Moreover, for semantics with constrained reconfigurations that are equivalent to the unconstrained one as far as the coverability and synchronization problems are concerned, it would be worth studying the impact of the reconfiguration restrictions (*e.g.* k-locally-constrained or f-constrained) on the minimum number of nodes for which a synchronizing execution exists, and on the minimum number of steps to synchronize.

References

1. Nathalie Bertrand, Paulin Fournier, and Arnaud Sangnier. Playing with probabilities in reconfigurable broadcast networks. In FoSSaCS'14, LNCS 8412, p. 134–148. Springer, 2014.
2. Nathalie Bertrand, Paulin Fournier, and Arnaud Sangnier. Distributed local strategies in broadcast networks. In CONCUR'15, LIPIcs 42, p. 44–57. LZI, 2015.
3. Roderick Bloem, Swen Jacobs, Ayrat Khalimov, Igor Konnov, Sasha Rubin, Helmut Veith, and Josef Widder. *Decidability of Parameterized Verification*, Synthesis Lectures on Distributed Computing Theory. Morgan & Claypool Publishers, 2015.
4. Patricia Bouyer, Nicolas Markey, Mickael Randour, Arnaud Sangnier, and Daniel Stan. Reachability in networks of register protocols under stochastic schedulers. In ICALP'16, LIPIcs 55, p. 106:1–106:14. LZI, 2016.
5. Giorgio Delzanno, Arnaud Sangnier, Riccardo Traverso, and Gianluigi Zavattaro. On the complexity of parameterized reachability in reconfigurable broadcast networks. In FSTTCS'12, LIPIcs 18, p. 289–300. LZI, 2012.
6. Giorgio Delzanno, Arnaud Sangnier, and Gianluigi Zavattaro. Parameterized verification of ad hoc networks. In CONCUR'10, LNCS 6269, p. 313–327. Springer, 2010.
7. E. Allen Emerson and A. Prasad Sistla. Symmetry and model checking. *Formal Methods in System Design*, 9(1-2):105–131, 1996.
8. Javier Esparza. Keeping a crowd safe: On the complexity of parameterized verification (invited talk). In STACS'14, LIPIcs 25, p. 1–10. LZI, 2014.
9. Javier Esparza, Alain Finkel, and Richard Mayr. On the verification of broadcast protocols. In LICS'99, p. 352–359. IEEE Comp. Soc. Press, 1999.
10. Javier Esparza, Pierre Ganty, and Rupak Majumdar. Parameterized verification of asynchronous shared-memory systems. In CAV'13, LNCS 8044, p. 124–140. Springer, 2013.
11. Paulin Fournier. *Parameterized verification of networks of many identical processes*. Thèse de doctorat, Université Rennes 1, France, 2015.
12. Steven M. German and A. Prasad Sistla. Reasoning about systems with many processes. *Journal of the ACM*, 39(3):675–735, 1992.
13. Matthew Hague. Parameterised pushdown systems with non-atomic writes. In FSTTCS'11, LIPIcs 13, p. 457–468. LZI, 2011.
14. Marvin Minsky. *Computation: Finite and Infinite Machines*. Prentice Hall, 1967.
15. Ichiro Suzuki. Proving properties of a ring of finite-state machines. *Information Processing Letters*, 28(4):213–214, 1988.