On the Coverability Problem for Asynchronous Broadcast Networks (Extended and Revised Version)

Giorgio Delzanno and Riccardo Traverso

DIBRIS, Università di Genova, Italy
{Giorgio.Delzanno,Riccardo.Traverso}@unige.it

Abstract. We study decidability and complexity of verification problems for networks in which nodes communicate via asynchronous broadcast messages. This type of communication is achieved by using a distributed model in which nodes have a local buffer. We consider here safety properties expressed as a coverability problem with an arbitrary initial configuration. This formulation naturally models the search of an initial topology that may lead to an error state in the protocol. We consider here different policies for handling local buffers such as unordered and (lossy) FIFO queues.

1 Introduction

We present (un)decidability and complexity results for the coverability problem of Asynchronous Broadcast Networks (ABN), a mathematical model of distributed systems in which processes interact via topology-dependent and asynchronous communication. Our formal model of asynchronous broadcast communication, presented in the extended abstract [11], combines three main features: a graph representation of a network configuration decoupled from the specification of individual process behaviour, a topology-dependent semantics of synchronization, and the use of local mailboxes to deliver messages to individual nodes. Our main abstraction comes from considering protocols defined via a communicating finite-state automaton replicated on each node of the network.

In our setting the coverability problem is formulated as follows. We first define an initial configuration as any graph in which nodes have labels that represents the initial state of the protocol (and no constraints on edges). Coverability consists then in checking whether there exists an initial configuration that can reach a target configuration that contains a specific pattern given in form of subgraphs or nodes in a given state. A similar decision problem is considered in [8] for a mathematical model with synchronous communication and dynamic reconfiguration of the topology called Reconfigurable Broadcast Network.

Our analysis is carried out with different policies to handle buffers, namely unordered bags (an abstraction of a tuple space), and perfect or lossy FIFO channels. Our technical contribution is as follows. We first show that, in contrast with the synchronous case discussed in [9,10], coverability is decidable when

local buffers are unordered. For the proof, we first give a reduction to the restricted case of fully connected topologies. We then solve the coverability problem through a reduction to the Cardinality Reachability Problem for Reconfigurable Broadcast Networks, a PTIME-complete problem [8]. This reduction improves the complexity of the decision procedure based on well-structured transition systems and symbolic backward analysis sketched in [11]. The resulting algorithm is based on a forward labelling procedure described in detail in [8].

When mailboxes are ordered buffers, we obtain undecidability already in the case of fully connected topologies. The undecidability proof is based on a non-trivial encoding of the set of operations of a two counter machine in form of a cooperation protocol between distinct nodes. The protocol consists of different phases, each one is defined over a distinct set of control messages. The difficulty of the encoding comes from the fact that it is not possible to infer well-formedness properties for the content of the mailbox of an individual node. Thus it is not possible to encode the current value of the counters using the current content of a set of mailboxes. The current value of the counters is represented however in the flow of messages consumed by a pair of nodes elected in a preliminary phase of the protocol, which, in turn, completes successfully only under certain conditions on the sequence of consumed control messages.

The coverability problem is decidable when introducing non-deterministic message losses. The results again follows from a reduction to the Cardinality Reachability Problem for RBN that improves the complexity w.r.t. our preliminary analysis [11].

In an extended model in which a node can test if its mailbox is empty, we obtain undecidability with unordered bags and both arbitrary or fully-connected topologies. For this reduction we need to control the interferences due to the simultaneous communication with several neighbours. We exploit here the emptiness test in order to enforce the well-formedness of the mailboxes of nodes involved in the simulation of counter machines.

To our knowledge, the present work shows the first complexity analysis for (parameterized) coverability in formal models of asynchronous broadcast communication.

Detailed proofs and encoding are presented in the technical report [12].

2 Asynchronous Broadcast Network (ABN)

In this section we formally define our asynchronous model for broadcast communication. A configuration is defined as a labelled graph. Nodes correspond to processes running a pre-defined protocol. Each node has a local message buffer used to collect messages sent by neighbours. We forbid self-loops in the communication graph to model half-duplex communication, a natural assumption for ad hoc and wireless networks (most of the results are invariant by adding self-loops, which however introduce additional technicalities in some of the proofs).

A protocol is specified via a finite-state automaton with send and receive operations that correspond to write [resp. read] on remote [resp. local] buffers.

Communication is topology-dependent, anonymous and asynchronous: when a process at node n sends a message a, the process does not block, and the message is added to the local mailbox of all of its neighbours without explicit information about the sender (i.e. messages do not contain node identifiers).

Formally, we consider a finite set Σ of messages, and different disciplines for handling the mailbox (message buffer), e.g., unordered mailboxes that we represent as bags over Σ, and ordered mailboxes that we represent as words over Σ.

In order to deal in a uniform way with different mailbox types we define a transition system parametric on the data structures used to model mailboxes. More specifically, we consider a mailbox structure $\mathbb{M} = \langle \mathcal{M}, del?, add, del, [] \rangle$, where $\mathcal{M}$ is a denumerable set of elements denoting possible mailbox contents; for $a \in \Sigma$ and $m \in \mathcal{M}$, $add(a, m)$ denotes the mailbox obtained by adding a to m, $del?(a, m)$ is true if a can be removed from m; $del(a, m)$ denotes the mailbox obtained by removing a from m when possible, undefined otherwise. Finally, $[] \in \mathcal{M}$ denotes the empty mailbox. We call an element a of m *visible* when $del?(a, m) = true$. Their specific semantics and corresponding properties change with the type of mailbox considered.

Definition 1. *A protocol is defined by a process $\mathcal{P} = \langle Q, \Sigma, R, q_0 \rangle$, where Q is a finite set of control states, Σ is a finite message alphabet, $Act = \{\tau\} \cup \{!!a, ??a \mid a \in \Sigma\}$, $R \subseteq Q \times Act \times Q$ is a set of transition rules, $q_0 \in Q$ is an initial control state.*

The label τ represents the capability of performing an internal action, and the label $!!a$ [$??a$] represents the capability of broadcasting [receiving] a message $a \in \Sigma$.

Definition 2. *Configurations are undirected $(Q \times \mathcal{M})$-graphs. A $(Q \times \mathcal{M})$-graph γ is a tuple $\langle V, E, L \rangle$, where V is a finite set of nodes, $E \subseteq V \times V$ is a finite set of edges (such that E is symmetric and $\forall v \in V.(v, v) \notin E$ to model undirected edges and half-duplex communication), and $L : V \to (Q \times \mathcal{M})$ is a labelling function.*

In the rest of the paper, for an edge $\langle u, v \rangle$ in E, we use the notation $u \sim_\gamma v$ and say that the vertices u and v are adjacent to one another in γ. We omit γ, and simply write $u \sim v$, when it is made clear by the context. We use $L(\gamma)$ to represent the set of labels in γ. The set of all possible configurations is denoted Γ, while $\Gamma_0 \subseteq \Gamma$ is the set of all initial configurations, in which nodes always have the same label $\langle q_0, [] \rangle$.

Given the labelling L and the node v s.t. $L(v) = \langle q, m \rangle$, we define $L_s(v) = q$ (state component of $L(v)$) and $L_b(v) = m$ (buffer component of $L(v)$). Furthermore, for $\gamma = \langle V, E, L \rangle \in \Gamma$, we use $L_s(\gamma)$ to denote the set $\{L_s(v) \mid v \in V\}$.

Definition 3. *For $\mathbb{M} = \langle \mathcal{M}, del?, add, del, [] \rangle$, an Asynchronous Broadcast Network (ABN) associated to $\mathcal{P}$ is a tuple $\mathcal{T}(\mathcal{P}, \mathbb{M}) = \langle \Gamma, \Rightarrow_\mathbb{M}, \Gamma_0 \rangle$, where $\Rightarrow_\mathbb{M} \subseteq \Gamma \times \Gamma$ is the transition relation defined next. For $\gamma = \langle V, E, L \rangle$ and $\gamma' = \langle V, E, L' \rangle$, $\gamma \Rightarrow_\mathbb{M} \gamma'$ holds iff one of the following conditions on L and L' holds:*

Local *There exists* $v \in V$ *such that* $(L_s(v), \tau, L'_s(v)) \in R$, $L_b(v) = L'_b(v)$, *and* $L(u) = L'(u)$ *for each* $u \in V \setminus \{v\}$.

Broadcast *There exists* $v \in V$ *and* $a \in \Sigma$ *such that* $(L_s(v), !!a, L'_s(v)) \in R$, $L_b(v) = L'_b(v)$ *and for every* $u \in V \setminus \{v\}$
- *if* $u \sim v$ *then* $L'_b(u) = add(a, L_b(u))$ *and* $L_s(u) = L'_s(u)$,
- *otherwise* $L(u) = L'(u)$.

Receive *There exists* $v \in V$ *and* $a \in \Sigma$ *such that* $(L_s(v), ??a, L'_s(v)) \in R$, $del?(a, L_b(v))$ *is satisfied,* $L'_b(v) = del(a, L_b(v))$, *and* $L(u) = L'(u)$ *for each* $u \in V \setminus \{v\}$.

A local transition only affects the state of the process that executes it, while a broadcast also adds the corresponding message to the mailboxes of all the neighbours of the sender. Notice that broadcast is never blocking for the sender. Receivers can read the message in different instants. This models asynchronous communication. A reception of a message a is blocking for the receiver whenever the buffer is empty or the visible elements are all different from a. If a is visible in the mailbox, the message is removed and the process moves to the next state.

An *execution* is a sequence $\gamma_0\gamma_1\ldots$ such that γ_0 is an initial configuration, and $\gamma_i \Rightarrow_{\mathbb{M}} \gamma_{i+1}$ for $i \geq 0$. We use $\Rightarrow^*_{\mathbb{M}}$ to denote the reflexive and transitive closure of $\Rightarrow_{\mathbb{M}}$. We drop $\mathbb{M}$ when the mailbox type is clear from the context.

Decision Problem The *Coverability Problem* parametric on the mailbox structure $\mathbb{M}$, abbreviated as $COV(\mathbb{M})$, is defined as follows.

Definition 4. *Given a protocol* $\mathcal{P}$ *with transition system* $\mathcal{T}(\mathcal{P}, \mathbb{M}) = \langle \Gamma, \Rightarrow_{\mathbb{M}}, \Gamma_0 \rangle$ *and a control state* q, *the coverability problem* $COV(\mathbb{M})$ *states: are there two configurations* $\gamma_0 \in \Gamma_0$ *and* $\gamma_1 \in \Gamma$ *such that* $\gamma_0 \Rightarrow^*_{\mathbb{M}} \gamma_1$ *and* $q \in L_s(\gamma_1)$*?*

We often use the terminology γ_0 reaches state q as an abbreviation for $\gamma_0 \Rightarrow^*_{\mathbb{M}} \gamma_1$ and $q \in L_s(\gamma_1)$ for some configuration γ_1. Besides being parametric on the mailbox structure, our decision problem is parametric on the shape of the initial configuration. As mentioned in the introduction, this feature models in a natural way verification problems for protocols with partial information about the structure of the network.

ABN vs RBN In the rest of the paper we will often refer to the semantics of RBN models [8]. Protocols in RBN adhere to the same syntax as ABN. Configurations are simply Q-graphs, i.e., graphs in which nodes have labels in Q via the labelling function L. The semantics of broadcast communication however is synchronous instead of asynchronous. Furthermore, the topology of the network may non-deterministically change. Formally, given $R_a(q) = \{q' \in Q \mid \langle q, ??a, q' \rangle \in R\}$ and two Q-graphs θ, θ' with $\theta = \langle V, E, L \rangle$, we have $\theta \to \theta'$ iff $\theta' = \langle V, E', L' \rangle$ and one of the following conditions holds:

Synch Broadcast $E' = E$ and $\exists v \in V$ s.t. $\langle L(v), !!a, L'(v) \rangle \in R$ and $L'(u) \in R_a(L(u))$ for every $u \sim v$, and $L(w) = L'(w)$ for any other node w.

Reconfiguration $E' \subseteq V \times V \setminus \{\langle v, v \rangle \mid v \in V\}$ and $L = L'$.

3 Unordered Mailboxes

In this section we study the coverability problems for ABNs in which mailboxes are unordered buffers modelled as bags over the finite message alphabet Σ. The mailbox structure Bag is defined as follows: $\mathcal{M}$ is the denumerable set of bags over Σ, $add(a, m) = [a] \oplus m$ (multiset sum of the singleton $[a]$ and m), $del?(a, m) = true$ iff $m(a) > 0$, $del(a, m) = m \ominus [a]$ (multiset removal of $[a]$ from m), and $[] \in \mathcal{M}$ is the empty bag $[]$. The operational semantics follows from the general definitions.

Let us consider the instance $COV(Bag)$ of the coverability problem. For synchronous broadcast, coverability is undecidable for arbitrary topologies [9]. We show next that coverability is in PTIME for unordered mailboxes.

For the ease of notation, we use $\mathcal{T}^{\mathcal{K}}(\mathcal{P}, \mathbb{M})$ [resp. $COV_{fc}(\mathbb{M})$] to denote the restriction of $\mathcal{T}(\mathcal{P}, \mathbb{M})$ [resp. $COV(\mathbb{M})$] to fully connected configurations only, i.e., configurations such that $u \sim_\gamma v$ for each pair of distinct nodes $u, v \in V$. We prove the results in two different steps. We first show that, for the purpose of deciding $COV(Bag)$, we can focus on fully connected topologies only. We then show a reduction from $COV_{fc}(Bag)$ to the Cardinality Reachability Problem for Reconfigurable Broadcast Networks, that, for short, we will refer to as CRP. The reduction requires reachability queries of the form $\#q \geq 1$ (at least one occurrence of control state q). The latter problem is PTIME-complete [8].

For asynchronous communication with unordered mailboxes, coverability for arbitrary topologies case can be reduced to the fully connected case. One side of the property is immediate. If there exists a fully connected initial configuration that reaches a configuration in which state q occurs, then coverability is solved. In order to prove the other implication, the intuition is that we can exploit the fact that mailboxes are unordered to ignore messages sent along links that are not present in a given topology. The following lemma then holds.

Lemma 1. *If there exists an arbitrary topology from which we can reach state q, then there exists a fully connected topology from which we can also reach q.*

Given an ABN protocol $\mathcal{P} = \langle Q, \Sigma, R, q_0 \rangle$ and a state $q \in Q$, we can solve $COV_{fc}(Bag)$ by solving CRP for the same protocol rules but with $\{q_0\}$ as singleton set of initial states and such that communication is intended via the synchronous semantics of RBN. We indicate such protocol $\mathcal{P}'$.

Lemma 2. *There exists an execution of $\mathcal{P}'$ that satisfies CRP if and only if there exists an execution of $\mathcal{P}$ satisfying $COV_{fc}(Bag)$.*

The previous reduction is done in constant time, since there is no need of modifying the protocol specification. We can therefore conclude that the following property holds.

Theorem 1. *$COV(Bag)$ is PTIME-complete.*

Proof. Thanks to Lemmas 1 and 2, we know that $COV(Bag)$ is in PTIME. Completeness follows from a reduction of the circuit value problem (CVP) to

$COV(Bag)$. For a given acyclic gate G and a fixed evaluation of its inputs, the reduction is based on a protocol in which a special node broadcasts the evaluation of a single input (in form of a message with label $true/false$ and an index associated to the corresponding variable). For each gate, we have nodes that receive the input and broadcast their output to the other nodes. A special node intercepts the *true* message corresponding to the output of the whole circuit and moves in an acceptance state. Regardless the type of communication topology and possible delays, coverability of the acceptance state corresponds to satisfiability of G w.r.t. the given assignment.

4 FIFO Mailboxes

In this section we move to ABN with perfect FIFO buffers as communication media. In this context we instantiate the mailbox structure $FIFO$ as follows: $\mathcal{M}$ is defined as Σ^*; $add(a, m) = m \cdot a$ (concatenation of a and m); $del?(a, m) = true$ iff $m = a \cdot m'$; $del(a, m)$ is the string m' whenever $m = a \cdot m'$, undefined otherwise; finally, $[] \in \mathcal{M}$ is the empty string ϵ.

Theorem 2. *$COV(FIFO)$ and $COV_{fc}(FIFO)$ are undecidable.*

Proof. The proof is based on a reduction of the halting problem for two-counter machines – a well known undecidable problem – to $COV(FIFO)$. A two-counter machine is defined by a pair $\langle Loc, Inst \rangle$ where Loc is a finite set of control locations and $Inst \subseteq Loc \times Op \times Loc$ is a finite set of instructions such that $Op = \{c{+}{+}, c{-}{-}, c == 0 \mid c \in \{x_1, x_2\}\}$ is a set of operators over the counters x_1 and x_2, and $\ell_0 \in Loc$ is the initial location. Configurations are tuples $\langle \ell, v_1, v_2 \rangle$ such that $\ell \in L$ is the current location and v_1, v_2 are natural numbers that denote the current value of x_1 and x_2, respectively. The operational semantics is defined in a standard way: the execution of increment and decrement updates the control location and the current value of the corresponding counter, a zero-test updates the location whenever the test is satisfied in the current state of the counter.

The rationale behind the reduction of coverability to the halting problem of two-counter machines is as follows. We first use an election protocol that assigns fixed roles (controller/slave) to a pair of adjacent nodes. Since the initial configuration is not fixed a priori our election protocol does not forbid the election of multiple pairs of controller/slave nodes, but we only require that at least one pair is elected in order to succeed. The controller/slave nodes set up their mailboxes in order to use them as overlapping circular queues. Messages represent the current value (in unary) of the counters. The simulation is guided by the controller. The slave forwards all received messages back to the controller. As an example, to check that x_1 is zero, the controller reads all messages in the mailbox and checks that in between two successive reads of the marker for x_1 there are no units. We use interference to denote an unwanted message occurring in the mailbox of a controller/slave node. Since the network topology is not fixed a priori, a key point of the whole construction is the capability of controlling interferences with other nodes, e.g., avoiding the adjacency between multiple

controllers and slaves. For this purpose, we use special control messages to coordinate the different phases and exploit the FIFO mailboxes in order to enforce the simulation to get into a deadlock state whenever the same control message is received more than once. A detailed description of the protocol is in [12]. The same construction can be used for the fully connected case.

5 Lossy FIFO Mailboxes

We now consider coverability for ABNs in which mailboxes are lossy FIFO channels, i.e., channels in which messages may non-deterministically be lost. Given a protocol $\mathcal{P}$, a configuration γ of $\mathcal{T}^K(\mathcal{P}, LFIFO)$ is a multiset of pairs $\langle q, m\rangle$ where $q \in Q$ and $m \in \Sigma^*$. To model non-deterministic loss of messages, we modify the operational semantics by introducing lossy steps.

We first need to define the ordering $\preccurlyeq$ between configurations. For $\gamma = \langle V, V \times V, L\rangle$ and $\gamma' = \langle V', V' \times V', L'\rangle$ $\gamma \preccurlyeq \gamma'$ iff there exists an injection $h : V \to V'$ s.t. $L_s(v) = L_s(h(v))$ and $L_b(v) \prec L_b(h(v))$ for each $v \in V$, where $\prec$ denotes the subword relation, namely, for $w, w' \in \Sigma^*$, $w \prec w'$ iff there exists an injective and strictly monotone mapping $h : |w| \to |w'|$ s.t. $w_i = w'_{h(i)}$ for $i : 1, \ldots, |w|$, where v_i denotes the i-th symbol in the word v. Intuitively, $\gamma \preccurlyeq \gamma'$ means that γ is obtained from γ' by removing nodes (and all corresponding edges) and messages from the buffers. We modify the transition relation $\Rightarrow$ to include lossy steps before and after each transition in the original system as follows: $\gamma \longmapsto \gamma'$ iff there exists η and ν s.t. $\eta \preccurlyeq \gamma$, $\eta \Rightarrow \nu$, and $\gamma' \preccurlyeq \nu$.

The ordering $\preccurlyeq$ is a simulation relation and well-quasi ordered. These two properties pave the way for a possible application of the theory of well-structured transition systems [11] to solve coverability. In the rest of the section we use a reduction to RBN-coverability to obtain better complexity results. As for unordered mailbox we first show that we can focus our attention on fully connected topologies, only.

Lemma 3. *There exists an execution of $\mathcal{P}$ that satisfies $COV(LFIFO)$ if and only if there is one of $\mathcal{P}$ satisfying $COV_{fc}(LFIFO)$.*

We are now at the most tricky part of the proof that consists in proving that $COV_{fc}(LFIFO)$ can be reduced to CRP. Let $\mathcal{P}$ be an ABN protocol, and let $\mathcal{P}'$ be the corresponding RBN protocol derived as in Section 3.

Lemma 4. *There exists an execution of $\mathcal{P}'$ that satisfies CRP if and only if there is one of $\mathcal{P}$ satisfying $COV_{fc}(LFIFO)$.*

The coverability problem for lossy FIFO mailboxes has a property in common with the one for bags, that is in both cases processes are able to ignore incoming messages indefinitely; this is achieved by either leaving the message in the multiset or by deleting it from the lossy FIFO queue. We can therefore take again advantage of this property to obtain the following theorem.

Theorem 3. *$COV_{fc}(LFIFO)$ is* PTIME*-complete.*

Proof. Membership to PTIME follows from the reduction to CRP for RBNs. Hardness follows again from a reduction of CVP to COV for ABN lossy FIFO queues. The encoding protocol is the same as for unordered mailboxes.

6 ABN with Emptiness Test

In this section we enrich the ABN model with a new type of transitions in order to enable nodes to test whether their mailbox is empty. We call the resulting model ABN_ϵ. The set Act of action labels is extended to include ϵ, i.e., $Act = \{\tau, \epsilon\} \cup \{!!a, ??a \mid a \in \Sigma\}$. The transition systems associated to an ABN_ϵ are changed accordingly to take ϵ into account; given two configurations $\gamma = \langle V, E, L\rangle$ and $\gamma' = \langle V, E, L'\rangle$, $\gamma \Rightarrow \gamma'$ holds also if the following condition is met.

Emptiness test There exists a $v \in V$ such that $(L_s(v), \epsilon, L'_s(v)) \in R$, $L_b(v) = L'_b(v) = []$, and $L(u) = L'(u)$ for each $u \in V \setminus \{v\}$.

The only difference w.r.t. the semantics of τ-transitions consists in the $L_b(v) = []$ condition, that ensures that ϵ-transitions only fire when the mailbox is empty.

The introduction of ϵ-transitions affects the different instances of the coverability problem in different ways. The simplest case is for $COV_{fc}(FIFO)$ and $COV(FIFO)$, which of course are still undecidable: the possibility to test the emptiness of the mailbox does not have any effect on the reduction from two-counter machines. The reduction from $COV_{fc}(LFIFO)$ to CRP of Lemma 4 has to be modified in order to consider also ϵ-transitions. Given two configurations $\gamma, \gamma' \in \Gamma$ such that $\gamma \preccurlyeq \gamma'$ (see Section 5 for the definition of the $\preccurlyeq$ ordering), if ϵ is enabled in γ then it can be fired starting from γ' too, through a preliminary lossy step that empties the relevant mailbox. This means that ϵ-transitions are pretty much the same as internal transitions in case of $LFIFO$ mailboxes. Therefore, given a protocol $\mathcal{P} = \langle Q, \Sigma, R, q_0\rangle$ and a target state $q \in Q$, we derive an RBN protocol $\mathcal{P}' = \langle Q, \Sigma, R', \{q_0\}\rangle$ where R' is the set of rules R where all occurrences of ϵ have been replaced by τ, and then we solve CRP for the target state q. Thanks to the previously mentioned property of ϵ-transitions, one could adapt easily enough the proof of Lemma 4 to this case. From these observations we can therefore derive that both $COV(LFIFO)$ and $COV_{fc}(LFIFO)$ are decidable even with ϵ-transitions.

We incur in a completely different case when considering bags: as it can be shown, the extended semantics traces indeed a sharp boundary between decidability and undecidability. Without the emptiness test, both reachability problems $COV(Bag)$ and $COV_{fc}(Bag)$ are decidable; we prove that the operator ϵ introduced with the extended model is sufficient to make them undecidable. The proof proceeds by building a reduction from the control state reachability problem for two-counter machines to $COV(Bag)$. The reduction encodes a counter machine $\mathcal{M}$ with an ABN protocol $\mathcal{P} = \langle Q, \Sigma, R, q_0\rangle$ where, like before, each location $\ell \in Loc$ and each instruction $i \in Inst$ corresponds respectively to a state $\mathcal{P}(\ell) \in Q$ and to a set of intermediate states and rules. The protocol is split in two phases. In the first phase processes follow a distributed election protocol

to identify who takes care of which role and who is excluded from the simulation. The second phase is the simulation of $\mathcal{M}$. The alphabet is partitioned in two sets, Σ_e for the election and Σ_s for the simulation. Since we do not make any particular assumption on the connectivity graph, the proof works for both $COV_{fc}(Bag)$ and $COV(Bag)$.

Election A simulation must be carried out by three nodes: a controller and two slaves, one per counter. As Figure 1 shows, the election protocol is similar to the one from section 4: the process which chooses to be a controller must receive announcements from two different slaves, while each slave must receive the announce message from a controller. We say that a node is *in simulation* if it reaches (at least once) $\mathcal{P}(\ell_0)$, q_{S_1}, or q_{S_2}. The election guarantees minimal

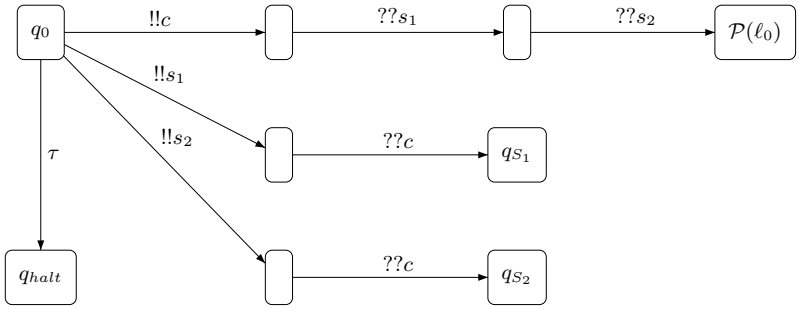

Fig. 1. $COV(Bag)$: Election protocol

connectivity requirements, as stated in the following Lemma.

Lemma 5. *If a node is in state $\mathcal{P}(\ell_0)$, then at least two of its neighbours are already respectively in state q_{S_1} and q_{S_2} or they can possibly move only those states. If a node is in state q_{S_1} or q_{S_2}, then at least one of its neighbours is already in state $\mathcal{P}(\ell_0)$ or it can possibly move only to $\mathcal{P}(\ell_0)$*

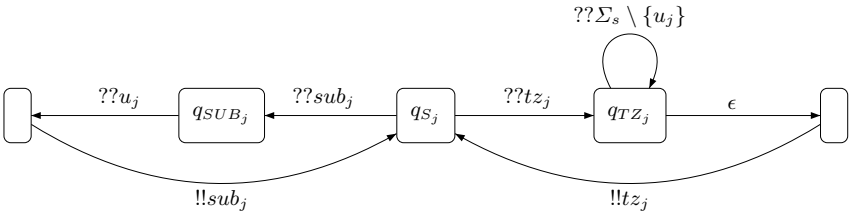

Fig. 2. $COV(Bag)$: Slave process

Simulation Each slave S_j keeps in its mailbox a number of u_j messages equal to the current value of counter x_j. The controller sends messages sub_j or tz_j to give

orders depending on the instruction (ℓ, op, ℓ') that is going to be simulated by the system and waits for the interested slave to react accordingly (see Figure 2). Once the slave is done, the same control message is sent back to the controller as acknowledgement and the controller is able to proceed. When A is a set we write $??A$ to mean that for every $a \in A$ the protocol has a reception rule $??a$ with the same endpoints. Again, the increment can be done directly by the controller with a single broadcast $!!u_j$. In order to be able to prove the correctness of the reduction, we first state some properties of the simulation phase.

Lemma 6. *Any $m \in \Sigma_e$ received by a node in simulation will persist in its mailbox forever. Such a node is said to be* in interference.

Proof. By construction, for all $m \in \Sigma_e$, there are no receptions of m starting from any state which may be reached by simulating nodes.

Lemma 7. *At any time, the value of the counter i is equal to the number of occurrences of units u_i in the mailbox of the corresponding slave, provided that no simulating node is in interference. We say than that the counters are* valid.

We remark that the notion of validity of the counters does not have anything to do with the compliance of their values w.r.t. the ones of the two-counter machine being simulated. Moreover, since the simulation may proceed even with invalid counters, the reduction does not compute reachability of the encoding $\mathcal{P}(\ell_f)$ of the target state ℓ_f, but instead it checks for the reachability of a fresh state q_{target} added according to Figure 3. This is needed in order to ensure the correctness of the simulation. It is straightforward to check that the instructions

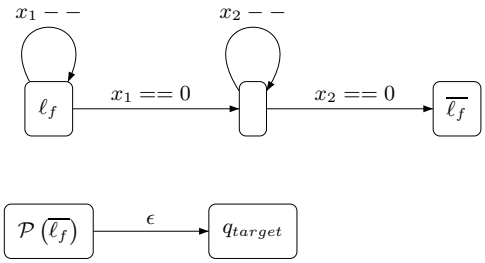

Fig. 3. $COV(Bag)$: Interference detection

added to $\mathcal{M}$ do not have any impact on the reachability of the target location, as they just decrement down to zero both counters before reaching the destination. We are now ready to prove that the reduction is indeed a correct simulation of the given two-counter machine.

Theorem 4. *$COV(Bag)$ [$COV_{fc}(Bag)$] is undecidable in ABN_ϵ.*

Proof. The correctness of the reduction can be proved by first demonstrating by induction on the number of simulated instructions that for any number of steps,

either the counters will be valid and consistent w.r.t. the corresponding state of the two-counter machine or they will be (and remain) invalid. Given this property, we can exploit Lemma 6 in order to show that the added, final transitions from Figure 3 ensure that the controller will deadlock before reaching the target state when the counters are invalid. A detailed proof is given in appendix. □

7 Related Work

Our analysis completes previous work on verification and expressiveness (w.r.t. coverability) of broadcast communication. More specifically, for synchronous broadcast communication, the coverability problem is decidable for fully connected graphs [7,14,6] and undecidable for arbitrary graphs in the AHN model studied in [9,10] and, for a timed version, in [1] . Broadcast in AHN is topology-dependent. Synchronous communication is used here to implement a discovery protocol that, by a careful control of interferences, allows individual nodes to infer precise information about their vicinity (e.g. the existence of one and only one neighbour with a certain role). In this paper the idea is similar, but to obtain undecidability the reductions exploit mailboxes, rather than linked structures, to encode counters. Failures and interferences for synchronous semantics have been studied in [16].

Concerning other models of broadcast communication, we would like to mention the CBS process calculi by Prasad [19] for fully connected networks with synchronous broadcast communication, the ω-calculus by Singh et al. [20,21] for fully connected networks with synchronous broadcast communication, and the model with topology-dependent broadcast by Ene and Muntean [13]. More recently, a process algebra for different types of communication, including asynchronous broadcast, called AWN, has been proposed in [15]. Semantics that take into consideration interferences and conflicts during a transmission have been proposed in [18,17]. Verification of unreliable communicating FIFO systems have been studied in [2,4]. In [5] the authors consider different classes of topologies with mixed lossy and perfect channels [5]. Reachability problems for graph-rewriting systems have been studied in [3].

Differently from all the previous works, we consider here coverability for parametric initial configurations for a distributed model with asynchronous broadcast. Furthermore, we also consider different policies to handle the message buffers (bags/queues) and as well as unreliability of the communication media.

Finally, our new complexity results improve the preliminary analysis presented in the extended abstract [11], where we used well-structured transition systems only for evaluating decidability for bags and lossy FIFO systems.

References

1. Parosh Aziz Abdulla, Giorgio Delzanno, Othmane Rezine, Arnaud Sangnier, and Riccardo Traverso. On the verification of timed ad hoc networks. In *FORMATS '11*, pages 256–270, 2011.

2. Parosh Aziz Abdulla and Bengt Jonsson. Undecidable verification problems for programs with unreliable channels. *Inf. Comput.*, 130(1):71–90, 1996.
3. Nathalie Bertrand, Giorgio Delzanno, Barbara König, Arnaud Sangnier, and Jan Stückrath. On the decidability status of reachability and coverability in graph transformation systems. In *RTA '12*, pages 101–116, 2012.
4. Gérard Cécé, Alain Finkel, and S. Purushothaman Iyer. Unreliable channels are easier to verify than perfect channels. *Inf. Comput.*, 124(1):20–31, 1996.
5. Pierre Chambart and Ph. Schnoebelen. Mixing lossy and perfect fifo channels. In *CONCUR*, pages 340–355, 2008.
6. Giorgio Delzanno. Constraint-based verification of parameterized cache coherence protocols. *FMSD*, 23(3):257–301, 2003.
7. Giorgio Delzanno, Javier Esparza, and Andreas Podelski. Constraint-based analysis of broadcast protocols. In *CSL*, pages 50–66, 1999.
8. Giorgio Delzanno, Arnaud Sangnier, Riccardo Traverso, and Gianluigi Zavattaro. On the complexity of parameterized reachability in reconfigurable broadcast networks. In *FSTTCS '12*, 2012. To appear.
9. Giorgio Delzanno, Arnaud Sangnier, and Gianluigi Zavattaro. Parameterized verification of ad hoc networks. In *CONCUR '10*, volume 6269 of *Lecture Notes in Computer Science*, pages 313–327. Springer, 2010.
10. Giorgio Delzanno, Arnaud Sangnier, and Gianluigi Zavattaro. On the power of cliques in the parameterized verification of ad hoc networks. In *FOSSACS '11*, pages 441–455, 2011.
11. Giorgio Delzanno and Riccardo Traverso. A formal model of asynchronous broadcast communication (preliminary results). In *ICTCS 12*, 2012. Available at the URL `http://ictcs.di.unimi.it/papers/paper_29.pdf`.
12. Giorgio Delzanno and Riccardo Traverso. On the coverability problem for asynchronous broadcast networks (extended and revised version). Technical report, TR-12-05, DIBRIS, University of Genova, November 2012. Available at the URL `http://www.disi.unige.it/research/expand-techrep?id_tr=103`.
13. Cristian Ene and Traian Muntean. A broadcast-based calculus for communicating systems. In *IPDPS '01*, page 149, 2001.
14. Javier Esparza, Alain Finkel, and Richard Mayr. On the verification of broadcast protocols. In *LICS '99*, pages 352–359, 1999.
15. Ansgar Fehnker, Rob J. van Glabbeek, Peter Höfner, Annabelle McIver, Marius Portmann, and Wee Lum Tan. A process algebra for wireless mesh networks. In *ESOP*, pages 295–315, 2012.
16. Gianluigi Zavattaro Giorgio Delzanno, Arnaud Sangnier. Verification of ad hoc networks with node and communication failures. In *FORTE*, pages 235–250, 2012.
17. Massimo Merro, Francesco Ballardin, and Eleonora Sibilio. A timed calculus for wireless systems. *Theor. Comput. Sci.*, 412(47):6585–6611, 2011.
18. Nicola Mezzetti and Davide Sangiorgi. Towards a calculus for wireless systems. *Electr. Notes Theor. Comput. Sci.*, 158:331–353, 2006.
19. K. V. S. Prasad. A calculus of broadcasting systems. *Sci. Comput. Program.*, 25(2-3):285–327, 1995.
20. Anu Singh, C. R. Ramakrishnan, and Scott A. Smolka. Query-based model checking of ad hoc network protocols. In *CONCUR '09*, volume 5710 of *Lecture Notes in Computer Science*, pages 603–619. Springer, 2009.
21. Anu Singh, C. R. Ramakrishnan, and Scott A. Smolka. A process calculus for mobile ad hoc networks. *Sci. Comput. Program.*, 75(6):440–469, 2010.

A Examples: executions with different mailbox policies

In the figures from 5 to 7 we show a few executions of an hypothetical protocol (in Figure 4) in order to give some examples on the runtime behaviour of an ABN. We represent with different colours the different local control states of individual process, and with square brackets the contents of mailboxes, where the leftmost item is to be considered the first one in case of ordered queues. Depending on the policy considered, some possibilities may be forbidden.

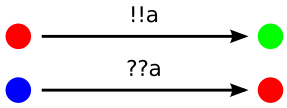

Fig. 4. Rules used in the examples from Figure 5 to 7

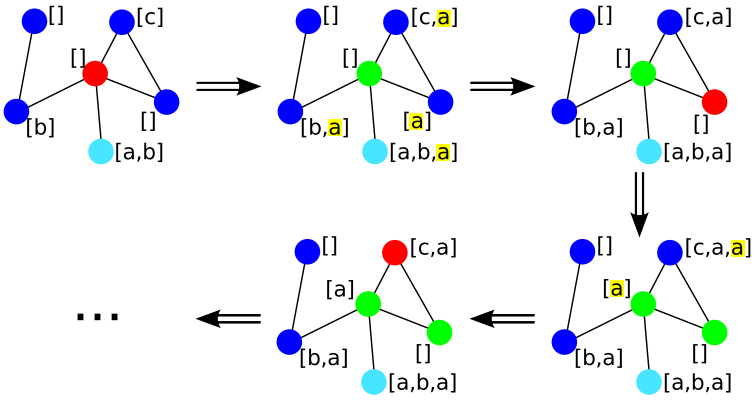

Fig. 5. Execution with *Bag* mailboxes

B Proofs

B.1 Proof of Lemma 1

Let $\mathcal{P} = \langle Q, \Sigma, R, q_0 \rangle$ be a protocol and let $\gamma = \langle V, E, L \rangle$ and $\gamma' = \langle V, E, L' \rangle$ be two configurations from the transition system $\mathcal{T}(\mathcal{P}, Bag)$ such that $\gamma \Rightarrow \gamma'$ by applying some rule $r \in R$ to a node $v \in V$.

Let $\subseteq^b$ denote the submultiset relation, i.e., for m_1, m_2 bags with symbols in Σ, $m_1 \subseteq^b m_2$ iff $m_1(a) \leq m_2(a)$ for every $a \in \Sigma$.

We first show the following lemma.

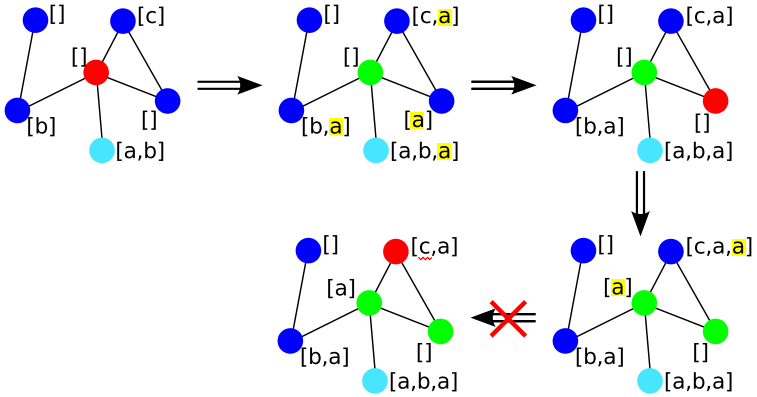

Fig. 6. Execution with *FIFO* mailboxes

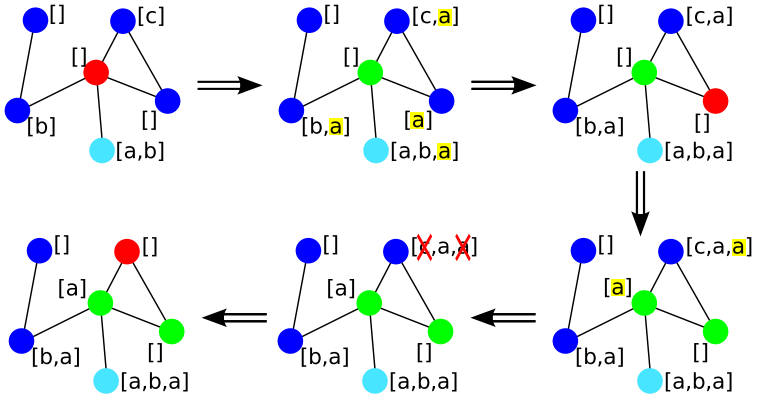

Fig. 7. Execution with *LFIFO* mailboxes

Lemma 8. *Given a configuration $\delta = \langle V, V \times V, K \rangle$ in the transition system $\mathcal{T}^{\mathcal{K}}(\mathcal{P}, Bag)$ such that, for all $u \in V$, $L_s(u) = K_s(u) \wedge L_b(u) \subseteq^b K_b(u)$ we can apply r at node v in order to reach any configuration $\delta' = \langle V, V \times V, K' \rangle$ such that $L'_s(u) = K'_s(u)$ and $L'_b(u) \subseteq^b K'_b(u)$ for all $u \in V$.*

Proof. The proof is by cases on the type of r. Specifically, for all $u \in V$ we have that

- if $r = (L_s(v), \tau, L'_s(v))$ then $L'_b(u) = K'_b(u)$ for every $u \in V$;
- if $r = (L_s(v), !!a, L'_s(v))$ then $L'_b(v) = K'_b(v)$ and $K'_b(u) = add(a, K_b(u))$ for every $u \in V \setminus \{v\}$;
- if $r = (L_s(v), ??a, L'_s(v))$ then $del?(a, K_b(v))$ holds because $L_b(v) \subseteq^b K_b(v)$ and by hypothesis $del?(a, L_b(v))$ is satisfied, $L'_b(v) = del(a, K_b(v))$, and $L'_b(u) = K'_b(u)$ for every remaining $u \in V \setminus \{v\}$.

We can now show by induction that, given an execution $\gamma_0\gamma_1 \dots \gamma_k$ in $\mathcal{T}(\mathcal{P}, Bag)$ where $q \in L_s(\gamma_k)$ and $\gamma_0 = \langle V, E, L^0 \rangle$ we can build another one $\delta_0\delta_1 \dots \delta_k$ in $\mathcal{T}^{\mathcal{K}}(\mathcal{P}, Bag)$ by starting from $\delta_0 = \langle V, V \times V, L^0 \rangle$ and replicating each rule application as shown in the previous scheme.

- The base case is immediate since initial configurations are graphs in which nodes have the same label, i.e., we can always find a sufficiently large fully connected initial configuration that contains any initial configuration (of arbitrary topology).
- For the inductive step, let $\gamma_0\gamma_1 \dots \gamma_k\gamma_{k+1}$ be an execution in $\mathcal{T}(\mathcal{P}, Bag)$. From the inductive hypothesis, we know that there exists $\delta_0\delta_1 \dots \delta_k$ such that the mailboxes in the nodes of δ_i contain the mailboxes in the nodes of γ_i for every i. To conclude the proof, we extend the property to $k+1$ steps by choosing an adequate, w.r.t. lemma 8, successor configuration δ_{k+1} of δ_k.

The thesis then follows by observing that, because of the construction, the set of control states in δ_i is the same as those in γ_i for $0 \leq i \leq k$ and in particular $q \in K_s(\gamma_k)$. □

B.2 Proof of Lemma 2

We start by stating another preliminary lemma on RBNs that we will need later. The lemma shows that for every state q that occurs in some intermediate state during a computation σ we can build an execution σ' that contains (in a sense specified below) the same sequence of steps as those in σ and that exposes q in its final configuration.

Lemma 9. *Let $\pi = \theta_0\theta_1 \dots \theta_n$ be an execution of an RBN with nodes in V and $\theta_n = \langle V, E, L \rangle$. For every state q that appears in at least one of the configurations of π, there exists another execution $\pi' = \theta'_0\theta'_1 \dots \theta'_m$ with nodes in $V' \supseteq V$ and $\theta'_m = \langle V', E', L' \rangle$ such that $L'(v) = L(v)$ for all $v \in V$ and there exists $v' \in V' \setminus V$ such that $L'(v') = q$.*

Proof. Let $\theta_0 = \langle V, E_0, L_0 \rangle$. We take $V' = V \cup (V \times \{1\})$ nodes and we show how to build the execution π' by having two replicas of the given execution π running in parallel. We start from the initial configuration $\theta'_0 = \langle V', E'_0, L'_0 \rangle$ such that $L'_0(v) = L'_0((v,1)) = L_0(v)$ and $E'_0 = \{(v_1, v_2), ((v_1,1),(v_2,1)) \mid (v_1, v_2) \in E_0\}$. It follows from the semantics of the RBN that we have an execution $\theta'_0\theta'_1 \dots \theta'_n$ where the nodes $v \in V$ mimics π step by step, and leaves every other $v' \in V' \setminus V$ in its initial state. Let $i \in \{0, \dots, n\}$ be the index of the first configuration θ_i that exposes a process in state q. The other half of the network, $V' \setminus V$, may now proceed mimicking the execution $\theta_0 \dots \theta_i$ without involving the nodes in V. We conclude that the Lemma holds by remarking that $\pi' = \theta'_0 \dots \theta'_n \dots \theta'_{n+i}$ satisfies all requirements. □

We can now prove the correctness of the reduction. Let $\theta_0 \to \theta_1 \to \cdots \to \theta_m$ be an execution of $\mathcal{P}'$ such that $\theta_i = \langle V, E'_i, L'_i \rangle$ for all $i \in \{1, \dots, m\}$. We now proceed by induction on m to prove that there exists an execution $\gamma_0 \Rightarrow \cdots \Rightarrow \gamma_n = \langle V, E, L^n \rangle$ such that $\forall v \in V$, $L^n_s(q) = L'_m(q)$.

- For $m = 0$, the configuration $\gamma_0 \in \Gamma_0$ with vertices V trivially satisfies the thesis.
- For $m \geq 1$, from the inductive hypothesis we know that given an execution $\theta_0 \to \cdots \to \theta_{m-1}$ of $\mathcal{P}'$ we have another one $\gamma_0 \Rightarrow \cdots \Rightarrow \gamma_k$ of $\mathcal{P}$ such that $L^k_s(q) = L'_{m-1}(q)$ for all $v \in V$.
  We want to prove that, given $\theta_m \in \Theta$ such that $\theta_{m-1} \to \theta_m$, there exists $\gamma = \langle V, E, L \rangle \in \Gamma$ such that $\gamma_k \Rightarrow^* \gamma$ and $L_s(\gamma) = L'(\theta_m)$. In the case when $\theta_{m-1} \to \theta_m$ because of a graph reconfiguration, then the thesis follows immediately by considering that $\gamma_k \Rightarrow^* \gamma$. In all other cases there exists a rule $r = \langle q, !!a, q' \rangle \in R$ such that $\theta_{m-1} \to \theta_m$ thanks to a broadcast by a node $v \in V$. We build the execution $\gamma_k \Rightarrow \gamma_{k+1} \Rightarrow \cdots \Rightarrow \gamma_{k+1+r}$ where $\gamma_k \Rightarrow \gamma_{k+1}$ corresponds to an application of rule r to the same node v that also occurs in γ_k. As a consequence, $L_b(u)$ contains the message a for each node $u \sim v$ in γ_{k+1}. The remaining r transitions are now reception steps, one for each of the r neighbours $u \in V$ of v that are enabled to react to the message sent by v. Finally, by choosing for the r steps the same exact reception rules that fired during the synchronous broadcast step, we obtain that for $\gamma = \gamma_{k+1+r} = \langle V, E, L \rangle$ we have that $L_s(v) = L'_m(v)$ for all $v \in V$. All those rules are necessarily enabled because the local state of individual processes by the inductive hypothesis matches the one in θ_{m-1} and, since their mailboxes are unordered, processes may indefinitely ignore previously received messages.

In order to prove the second implication we will show how we can rely on graph reconfigurations (instead of unordered mailboxes) in order to simulate the execution of an ABN. Let $\gamma_0 \Rightarrow \gamma_1 \Rightarrow \cdots \Rightarrow \gamma_n = \langle V, E, L^n \rangle$ be an execution of $\mathcal{P}$. We show by induction on n that there exists an execution $\theta_0 \to \theta_1 \to \cdots \to \theta_m$ of $\mathcal{P}'$ (with $\theta_i = \langle V', E'_i, L'_i \rangle$ for all $i \in \{1, \dots, m\}$), such that $V \subseteq V'$ and $\forall v \in V$, $L'_m(v) = L^n_s(v)$. In both semantics the sets of nodes cannot change during an execution, thus we will call V the set of nodes of the ABN and V' the set of nodes of the RBN.

- For $n = 0$, every configuration $\theta_0 \in \Theta_0$ with vertices $V' \supseteq V$ satisfies the thesis.
- Now, let us consider $n \geq 1$. We know by hypothesis that given an execution $\gamma_0 \Rightarrow^* \gamma_{n-1}$ of $\mathcal{P}$, there exists an execution $\theta_0 \rightarrow^* \theta_{m'}$ of $\mathcal{P}'$ such that $V \subseteq V'$ and $\forall v \in V$, $L'_{m'}(v) = L_s^{n-1}(v)$, and we want to prove that, given $\gamma_n \in \Gamma$ such that $\gamma_{n-1} \Rightarrow \gamma_n$, there exists an execution $\theta'_0 \rightarrow^* \theta'_{m''}$ of $\mathcal{P}'$ such that if $\theta'_{m''} = \langle V'', E'', L'' \rangle$ then $V \subseteq V''$ and $L''(v) = L_s^n(v)$ for all $v \in V$. There are three cases to consider, namely local step, asynchronous broadcast and receive. In the first two cases, it suffices to note that by first removing all edges of $\theta_{m'}$ through a reconfiguration, the execution of the same step as in the ABN does not involve any other node. The configuration $\theta_{m'+2} = \langle V', E'_{m'+2}, L'_{m'+2} \rangle$ reached through these two steps, thanks to the inductive hypothesis, easily satisfies all requirements. In the third case, reception steps, we will need a slightly more complex construction. Let $r = \langle q_1, ??a, q_2 \rangle \in R$ be the rule fired during the step $\gamma_{n-1} \Rightarrow \gamma_n$. Since $\gamma_0 \dots \gamma_n$ is an execution, we know that there exist a natural $i \in \{0, \dots, n\}$ and a state $q_3 \in L_s(\gamma_i)$ such that there exists a rule $r' = \langle q_3, !!a, q_4 \rangle \in R$. By applying the inductive hypothesis, some configuration of the execution $\theta_0 \dots \theta_{m'}$ must also expose a node in state q_3, thus, by Lemma 9, we have another execution $\theta'_0 \dots \theta'_{m''}$ with nodes in $V'' \supseteq V'$ and $\theta'_{m''} = \langle V'', E'', L'' \rangle$ such that $L''(v) = L'_{m'}(v)$ for all $v \in V'$ and there exists $u \in V'' \setminus V'$ such that $L''(u) = q_3$. We can then reconfigure the set of edges of $\theta'_{m''}$ to $\{(u, v), (v, u)\}$ and let this new node u to synchronize through r' with the node $v \in V$ that fires the reception rule r in the ABN.

□

B.3 Proof of Theorem 2

Formally, let $\mathcal{M} = \langle Loc, Inst \rangle$ be a two-counter machine. The encoding of $\mathcal{M}$ is defined via an ABN protocol $\mathcal{P}_\mathcal{M} = \langle Q, \Sigma, R, q_0 \rangle$, where each location $\ell \in Loc$ corresponds to a state $\mathcal{P}(l) \in Q$, and each instruction $r \in Inst$ corresponds to a set of auxiliary states and rules. The i-th counter is represented by the FIFO mailboxes of two collaborating processes that forward each other the units u_i and a distinguished token t_i which marks the beginning of the circular queue. The protocol is split in three phases: election, initialization and simulation. The alphabet Σ is partitioned in Σ_e, the messages exchanged during the election, and Σ_s, the messages used for the simulation.

Election Since all processes start the protocol in the same initial state, the first thing is to distinguish their roles and make sure that all required communication links are present. This is the purpose of the election phase, during which processes try to build pairs of communicating nodes. An active process may be either a controller or a slave. The duty of the controller is to orchestrate the whole simulation; the purpose of the slave is to bounce back every simulation message received from the controller. We do not care of how pairs will be able to proceed to the next phase, as long as their minimum connectivity

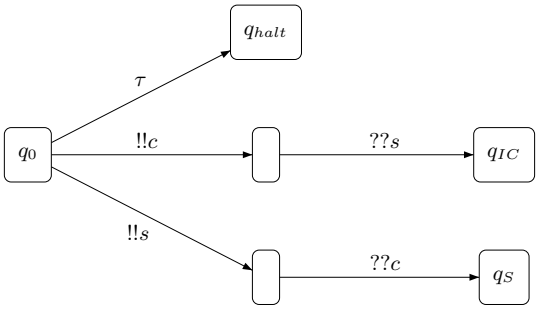

Fig. 8. Distributed election protocol

requirements are fulfilled: each of them will try to independently carry on its own simulation, or deadlock due to interferences.
Figure 8 shows the election protocol. At first every node non-deterministically chooses an active role and announces it to its neighbours ($!!c$ or $!!s$) or shuts down itself by going to q_{halt}. After choosing a role, an handshake is needed to ensure the existence of the interconnection between controller and slave: a controller needs to receive the announce from a slave, and vice versa. The slave reaches the state q_S that starts its main loop, while the controller goes to an intermediate initialization state q_{IC}.

At the end of the election protocol we have the following properties.

Lemma 10. *If a node is in state q_{IC} [resp. q_S], then at least one of its neighbours is already in state q_S [resp. q_{IC}] or it can possibly move only to q_S [resp. q_{IC}].*

Proof. In order for a node n to reach state q_{IC}, n has to read message s from the buffer, but this can happen only if there exists a neighbour m of n that sent s. If that is the case, then m will either move to state q_S by consuming the message c previously sent by m or it will idle forever without taking part in the simulation (e.g. because the visible message in its mailbox is s and not c). For a node in state q_S we can apply a symmetric reasoning. □

Notice that if two neighbours are respectively in state q_{IC} and q_S, there is no guarantee that their mailboxes will be empty. E.g. a node in state q_{IC} may contain a message c sent by another controller. In the rest of the simulation we will enforce a sanity check on the mailboxes in order to ensure that spurious control messages block the execution. This idea is formalized in Lemma 11, but we need to introduce some definitions first. A node in any local state reached after q_S or q_{IC} is said to be *in simulation*. A node *produces* [resp. *consumes*] a (non-empty) string $w \in \Sigma^*$ between q and q' if the corresponding automaton, starting from the local state q, executes a sequence of transitions leading to q' while broadcasting [resp. receiving] the messages in w in the same order. We are now ready to state the lemma.

Lemma 11. *Any string w consumed by a node n in simulation belongs to Σ_s^*. Furthermore, there is a single neighbour m which is in simulation and produces every w consumed by n.*

Proof. We enforce this properties by construction. After successfully completing an election (i.e. reaching q_S or q_{IC}) n begins the simulation, during which it only sends and reads messages from Σ_s, and it has no transitions from states used during the simulation back to states used for the election. Thanks to this, the first property holds. For what concerns the second property, we observe that any node in simulation has been able to consume just one Σ_e-message, during the election, and that the strings it can produce are words $e' \cdot w'$ with $e' \in \Sigma_e$ and $w' \in \Sigma_s^*$. Thanks to Lemma 10 and to the fact that w is not empty, at least neighbour m in simulation has to exist. Whenever another neighbour m' completes the election and starts the simulation, then n will receive two (interleaved) strings $e_1 \cdot w_1$ and $e_2 \cdot w_2$, with $e_1, e_2 \in \Sigma_e$ and $w_1, w_2 \in \Sigma_s^*$, respectively from m and m'. Therefore, because of the first property, we can conclude that n will not be able to consume any more messages as soon as e_2 becomes visible in its mailbox: any string w consumed by n in simulation must be a substring of w_1. □

In order to simplify the presentation we introduce some syntactic sugar. Let $q \in Q$ be a state with an outgoing reception transition labelled by $??(m)$; then, in the actual ABN automaton, the transition is replaced by a regular $??m$ and appropriate deletion rules (q, m', q) are added to get rid of all those messages $m' \in \Sigma_s$ that cannot be consumed by some outgoing transition.

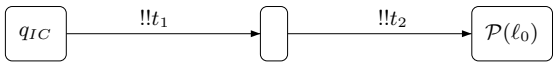

Fig. 9. Initialization protocol

Initialization Once all nodes have been elected, it is time for the controller to initialize the representation of the counters. This is simply achieved by creating two tokens t_1 and t_2 to be used as markers for the circular queues representing the counters (Figure 9). At this point the controller is able to move to the encoding of the initial location of $\mathcal{M}$, $\mathcal{P}(\ell_0)$.

Slave nodes When everything is ready, the nodes proceed to the simulation phase. The slave node starts in q_s, a sort of idle state which manages the forwarding of messages of the counters (Figure 10). By applying Lemma 11 we know that each slave node from q_s to q_s consumes a string of messages which is always produced by the same controller node, thus it will never forward any third-party messages.

We are ready now to discuss the simulation of each operation.

Increment In order to increment the j-th counter the controller needs to put one more unit u_j in the circular queue, so a simple $!!u_j$ is enough.

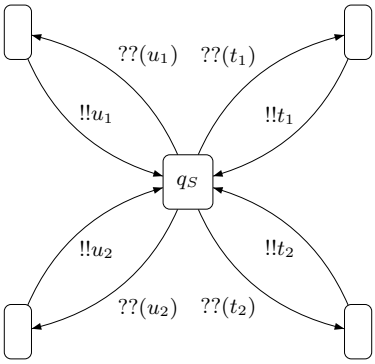

Fig. 10. Main loop of the slave node q_S

Subtraction Subtraction is achieved by removing a u_j token from the loop (Figure 11); if x_j is already at 0, the controller will loop forever on q_{SUB_j} without being able to proceed.

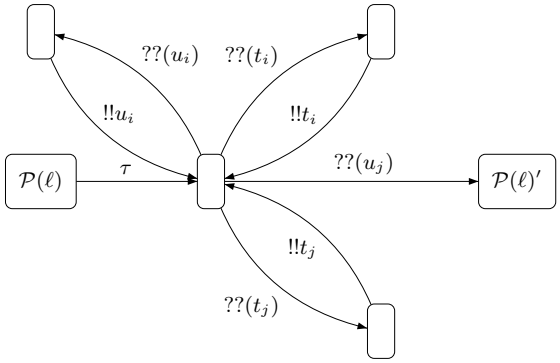

Fig. 11. Subtraction from counter j, with $i \neq j$

Testing for zero Testing for zero x_j can be done by checking for a sequence of two messages t_j in a row – or going to deadlock in any other case. Of course both operations must carefully keep forwarding messages for the other counter x_i.

Subtraction and zero testing are not atomic, and to be sure that the slave was alive during the whole execution the reduction applies the greeting protocol shown in Figure 13 after the initialization and after each operation, so that for each instruction of the form $(\ell, op, \ell') \in Inst$ the system may reach the $\mathcal{P}(\ell')$ state only after exchanging hello messages; in case of interferences deadlocks are propagated to the controller before completing the simulation of the transition. Furthermore, the greeting protocol allows us to extract an exact representation of a counter to reason about, as stated in the following lemma.

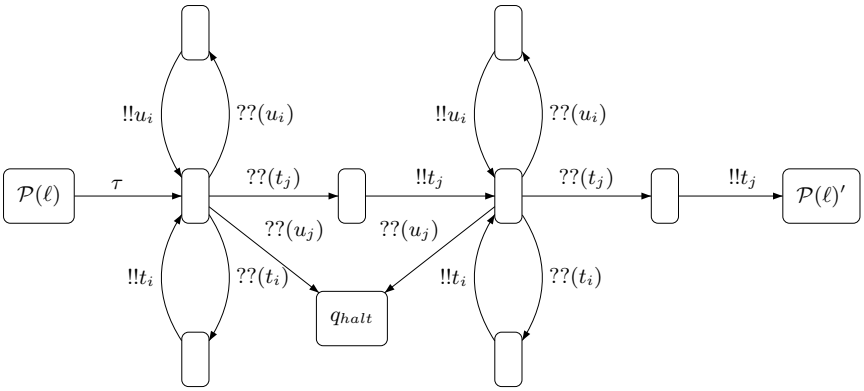

Fig. 12. Testing for zero counter j, with $i \neq j$

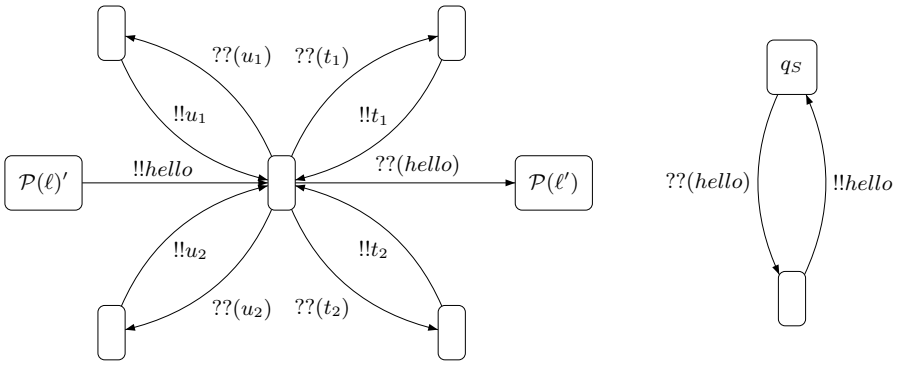

Fig. 13. Greeting protocol

Lemma 12. *For each instruction $(\ell, op, \ell') \in Inst$, the string consumed (and produced) by a controller node between $\mathcal{P}(\ell)'$ and $\mathcal{P}(\ell')$ is a representation of the current state of the two counters (the value of the counter i is equal to the number of occurrences of units u_i).*

Proof. The only difference between the string consumed and the one produced between $\mathcal{P}(\ell)'$ and $\mathcal{P}(\ell')$ is in the position of the unique *hello* message, which is situated at the beginning of the produced string and at the end of the consumed one. Apart from that, the rest of the two strings will be the same, because the controller broadcasts each message immediately after receiving it in the same order. Since the slave also forwards messages as they appear in his mailbox, a run of the greeting protocol does not change the representation of the counters and does not add new messages into the circular queue (*hello* is first produced and then consumed by the controller itself).

Thanks to Lemma 11 we know that the controller always exchanges messages with the same slave, and because of the introduction in the circular queue of a unique *hello* token it can stop consuming messages as soon as it receives it again, when he read each message in the queues exactly once. Therefore, if w is the string consumed by the controller between $\mathcal{P}(\ell)'$ and $\mathcal{P}(\ell')$, then the number of u_1 messages contained in w is the current value of the counter x_1, and the same applies for u_2 and x_2. □

We now show by induction on the number of instructions $k \in \mathbb{N}$ in the execution of a two-counter machine that for each $(\ell, op, \ell') \in Inst$ the simulation either leads the system to a deadlock state or it moves the controller to $\mathcal{P}(\ell')$ while maintaining a consistent representation of the counters w.r.t. the corresponding state of the two-counter machine. For the base case $k = 0$, after the initialization we reach $\mathcal{P}(\ell_0)$ only if the controller node is paired with a single slave (Lemma 11) and the first execution of the greeting protocol succeeded, exposing a configuration of the circular queue in which both counters have value 0 (the controller only produced one t_1 and one t_2 message so far, and no units). Assume now that we simulated $i > 0$ instructions, and we are going to simulate the $(k+1)$-th one, $(\ell, op, \ell') \in Inst$. For the inductive hypothesis, we know that we reached $\mathcal{P}(\ell)$ if and only if the last execution of the greeting protocol showed consistent values for x_1 and x_2. There are three cases to consider, depending on the nature of the instruction. An increment $x_i{+}{+}$ is encoded through a single broadcast of a new unit, therefore if the previous greeting showed c_i units u_i and d_j units u_j then the one leading the controller from $\mathcal{P}(\ell)'$ to $\mathcal{P}(\ell')$ will expose $c_i + 1$ units u_i and d_j units u_j for $i \neq j$. A decrement $x_i{-}{-}$ works by forwarding to the slave every message in $\{t_i, t_j, u_j\}$ and by consuming a single u_i message, leading again to a consistent state of the representation of the counters in the greeting after the decrement. Finally, in case of a test $x_i == 0$ the controller is able to reach the next greeting only by producing the same string consumed from $\mathcal{P}(\ell)$ to $\mathcal{P}(\ell)'$: by construction, the only cases which may lose some messages in the circular queue are those leading to the q_{halt} state, which is a deadlock.

The undecidability proof is valid also for fully connected topologies.

B.4 Proof of Lemma 3

Let $\mathcal{P} = \langle Q, \Sigma, R, q_0 \rangle$ be a protocol and let $\gamma = \langle V, E, L \rangle$ and $\gamma' = \langle V, E, L' \rangle$ be two configurations from the transition system $\mathcal{T}(\mathcal{P}, LFIFO)$ such that $\gamma \Rightarrow \gamma'$ by applying some rule $r \in R$ to a node $v \in V$.

We first show that $\preccurlyeq$ is a simulation relation.

Lemma 13. *Given a configuration $\delta = \langle V, V \times V, K \rangle$ s.t. $\gamma \preccurlyeq \delta$ we can apply r at node v in order to reach a configuration $\delta' = \langle V, V \times V, K' \rangle$ such that $\gamma' \preccurlyeq \delta'$.*

Proof. The proof is by cases on the type of r. Specifically, for all $u \in V$ we have that

- if $r = (L_s(v), \tau, L'_s(v))$ then, since the semantics applies lossy steps, $L'_b(v) \prec L_b(v)$ for every $u \in V$ (messages can get lost).
- if $r = (L_s(v), !!a, L'_s(v))$ then $L'_b(v) \prec K'_b(v)$ and $K'_b(u) \prec add(a, K_b(u))$ for every $u \in V \setminus \{v\}$.
- if $r = (L_s(v), ??a, L'_s(v))$ then $del?(a, K_b(v))$ holds because $L_b(v) \prec K_b(v)$ and by hypothesis $del?(a, L_b(v))$ is satisfied, $L'_b(v) \prec del(a, K_b(v))$, and $L'_b(u) \prec K'_b(u)$ for every remaining $u \in V \setminus \{v\}$.

In all three cases by applying the same transition to δ and assuming that no message is lost in δ we obtain a configuration δ' such that $\gamma' \preccurlyeq \delta'$.

We can now show by induction that, given an execution $\gamma_0\gamma_1 \dots \gamma_k$ in $\mathcal{T}(\mathcal{P}, LFIFO)$ where $q \in L_s(\gamma_k)$ and $\gamma_0 = \langle V, E, L^0 \rangle$ we can build another one $\delta_0\delta_1 \dots \delta_k$ in $\mathcal{T}^{\mathcal{K}}(\mathcal{P}, LFIFO)$ by starting from $\delta_0 = \langle V, V \times V, L^0 \rangle$ and replicating each rule application as shown in the previous scheme.

- The base case is immediate since initial configurations are graphs in which nodes have the same label, i.e., we can always find a sufficiently large fully connected initial configuration that contains any initial configuration (of arbitrary topology).
- For the inductive step, let $\gamma_0\gamma_1 \dots \gamma_k\gamma_{k+1}$ be an execution in $\mathcal{T}(\mathcal{P}, LFIFO)$. From the inductive hypothesis, we know that there exists $\delta_0\delta_1 \dots \delta_k$ with $\gamma_i \preccurlyeq \delta_i$ for $i : 1, \dots, k$. To conclude the proof, we can now apply lemma 13 to extend the property to $k+1$ steps by choosing an adequate (according to the lemma T) successor configuration δ_{k+1} of δ_k.

The thesis then follows by observing that, because of the construction, the set of control states in δ_i is the same as those in γ_i for $0 \leq i \leq k$ and in particular $q \in K_s(\gamma_k)$. □

B.5 Proof of Lemma 4

Let $\theta_0 \to \theta_1 \to \dots \to \theta_m$ be an execution of $\mathcal{P}'$ such that $\theta_i = \langle V, E'_i, L'_i \rangle$ for all $i \in \{1, \dots, m\}$. We now proceed by induction on m to prove that there exists an execution $\gamma_0 \Rightarrow \dots \Rightarrow \gamma_n = \langle V, E, L^n \rangle$ such that $\forall v \in V$, $L^n_s(q) = L'_m(q)$ and such that $L^i_b(v)$ has always at most one message.

- For $m = 0$, the configuration $\gamma_0 \in \Gamma_0$ with vertices V trivially satisfies the thesis.
- For $m \geq 1$, from the inductive hypothesis we know that given an execution $\theta_0 \to \cdots \to \theta_{m-1}$ of $\mathcal{P}'$ we have another one $\gamma_0 \Rightarrow \cdots \Rightarrow \gamma_k$ of $\mathcal{P}$ such that $L_s^k(q) = L'_{m-1}(q)$ for all $v \in V$ and such that all queues have at most one message. We want to prove that, given $\theta_m \in \Theta$ such that $\theta_{m-1} \to \theta_m$, there exists $\gamma = \langle V, E, L \rangle \in \Gamma$ such that $\gamma_k \Rightarrow^* \gamma$ and $L_s(\gamma) = L'(\theta_m)$. In the case when $\theta_{m-1} \to \theta_m$ because of a graph reconfiguration, then the thesis follows immediately by considering that $\gamma_k \Rightarrow^* \gamma$. In all other cases there exists a rule $r = \langle q, !!a, q' \rangle \in R$ such that $\theta_{m-1} \to \theta_m$ thanks to a broadcast by a node $v \in V$. We build the execution $\gamma_k \Rightarrow \gamma_{k+1} \Rightarrow \cdots \Rightarrow \gamma_{k+1+r}$ where $\gamma_k \Rightarrow \gamma_{k+1}$ corresponds to an application of rule r to the same node v that also occurs in γ_k. Furthermore, we apply a lossy step to ensure that message a is always visible in the mailboxes of nodes that are connected to v in θ_{m-1} and to simply remove the message a from the mailboxes of nodes that are not connected to v in θ_{m-1}. As a consequence, $L_b(u)$ contains now the message a for each node $u \sim v$ in γ_{k+1}.
  The remaining r transitions are now reception steps, one for each of the r neighbours $u \in V$ of v that are enabled to react to the message sent by v. Finally, by choosing for the r steps the same exact reception rules that fired during the synchronous broadcast step (they are necessarily enabled because the local state of individual processes by the inductive hypothesis matches the one in θ_{m-1}), we obtain that for $\gamma = \gamma_{k+1+r} = \langle V, E, L \rangle$ we have that $L_s(v) = L'_m(v)$ and $L_b(v)$ has at most one message for all $v \in V$.

The other side of the implication can be proved in exactly the same way as for the case of unordered mailboxes, because when we assume an ABN execution the semantics of the mailbox is already implicitly considered and all preconditions to reception steps are satisfied (it would not be an execution otherwise). □

B.6 Proof of Lemma 7

As in the FIFO case, every node by construction broadcasts a single message in Σ_e before everything else. Since each node is only able to consume – during election – a number of Σ_e-messages equal to to the number of required neighbours, it follows that we should expect interferences as soon as a simulating node receives a message $m \in \Sigma_e$. Finally, we conclude that invalid counters will never become valid again thanks to Lemma 6. □

B.7 Proof of Theorem 4

Let us consider an execution simulating $k \in \mathbb{N}$ instructions of the two-counter machine. At $k = 0$, either the counters are invalid or they are valid and they are trivially both equal to 0. At the $(k+1)$-th step, we simulated the previous k instructions either by ending up with invalid counters, or with valid counters and c_i units u_i and d_j units u_j. In the first case, after the step $k+1$ the counters

will still be invalid (Lemma 7). For the second case, let $(\ell, op, \ell') \in Inst$ be the $(k+1)$-th instruction that is going to be simulated. If interferences occurs during its execution then the counters will be invalid. Otherwise, there are three cases to consider. When op is an increment x_i++, the whole instruction is simulated by a single broadcast that will add a unit u_j to the mailbox of slave S_i, therefore the new values for the counters will be respectively $c_i + 1$ and d_j. When op is a decrement x_i--, the slave S_i will be forced to read a unit u_i from its mailbox (while S_j will not react), in such a way that only the value of counter x_i will change (into $c_i - 1$) after the step, provided that $c_i > 0$. When op is a test $x_i == 0$, the slave S_j will not react and S_i will delete only unuseful messages in $\Sigma_s \setminus \{u_i\}$ in order to complete the operation, therefore both of the counters after the execution of the step will still have the same values as before. In any case, we can say that after the execution of a new step, either the counters will be valid and consistent w.r.t. the corresponding state of the two-counter machine or they will be invalid.

Because of the instructions and transitions introduced in Figure 3, when the controller is in state $\mathcal{P}(\ell_f)$ both slave nodes have to execute an ϵ-transition (during the latest tests for zero) in order for the controller to be able to try to execute his latest ϵ-transition leading to the target state q_{target}. Thanks to the previously proved property on executions we can conclude that either the system will deadlock just before reaching q_{target} when the counters are not valid (thanks to Lemma 6, ϵ-transitions cannot be executed), or will reach q_{target} with valid and consistent counters. □