


1 Reconfiguration and message losses in 2 parameterized broadcast networks

3 **Nathalie Bertrand** 

4 Univ. Rennes, Inria, CNRS, IRISA – Rennes (France)

5 **Patricia Bouyer** 

6 LSV, CNRS & ENS Paris-Saclay, Univ. Paris-Saclay – Cachan (France)

7 **Anirban Majumdar**

8 Univ. Rennes, Inria, CNRS, IRISA – Rennes (France)

9 LSV, CNRS & ENS Paris-Saclay, Univ. Paris-Saclay – Cachan (France)

10 — Abstract —

11 Broadcast networks allow one to model networks of identical nodes communicating through message
12 broadcasts. Their parameterized verification aims at proving a property holds for any number
13 of nodes, under any communication topology, and on all possible executions. We focus on the
14 coverability problem which dually asks whether there exists an execution that visits a configuration
15 exhibiting some given state of the broadcast protocol. Coverability is known to be undecidable for
16 static networks, *i.e.* when the number of nodes and communication topology is fixed along executions.
17 In contrast, it is decidable in **PTIME** when the communication topology may change arbitrarily
18 along executions, that is for reconfigurable networks. Surprisingly, no lower nor upper bounds on the
19 minimal number of nodes, or the minimal length of covering execution in reconfigurable networks,
20 appear in the literature.

21 In this paper we show tight bounds for cutoff and length, which happen to be linear and quadratic,
22 respectively, in the number of states of the protocol. We also introduce an intermediary model with
23 static communication topology and non-deterministic message losses upon sending. We show that
24 the same tight bounds apply to lossy networks, although, reconfigurable executions may be linearly
25 more succinct than lossy executions. Finally, we show **NP-completeness** for the natural optimisation
26 problem associated with the cutoff.

27 **2012 ACM Subject Classification** Theory of computation → Verification by model checking

28 **Keywords and phrases** model checking – parameterized verification – broadcast networks

29 **Funding** The second author was supported by ERC project EQualIS (308087).

30 **1** Introduction

31 *Parameterized verification.* Systems formed of many identical agents arise in many concrete
32 areas: distributed algorithms, populations, communication or cache-coherence protocols,
33 chemical reactions etc. Models for such systems depend on the communication or interaction
34 means between the agents. For example pairwise interactions are commonly used for
35 populations of individuals, whereas selective broadcast communications are more relevant for
36 communication protocols on ad-hoc networks. The capacity of the agents, and thus models
37 that are used to represent their behaviour also vary.

38 Verifying such systems amounts to checking that a property holds independently of the
39 number of agents. Typically, a consensus algorithm should be correct for any number of
40 participants. We refer to these systems as parameterized systems, and the parameter is
41 the number of agents. The verification of parameterized systems started in the late 80's
42 and recently regained attention from the model-checking community [11, 8, 6, 1]. It can be
43 seen as particular cases of infinite-state-system verification, and the fact that all agents are
44 identical can sometimes lead to efficient algorithms [5].



© N. Bertrand, P. Bouyer, A. Majumdar;
licensed under Creative Commons License CC-BY

30th International Conference on Concurrency Theory (CONCUR 2019).

Editors: Wan Fokkink and Rob van Glabbeek; Article No. 28; pp. 28:1–28:15

Leibniz International Proceedings in Informatics



LIPIC Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

45 *Broadcast networks.* This paper targets the application to protocols over ad-hoc networks,
46 and we thus focus on the model of broadcast networks [3]. A broadcast network is composed
47 of several nodes that execute the same broadcast protocol. The latter is a finite automaton,
48 where transitions are labeled with message sendings or message receptions. Configuration in
49 broadcast networks is then comprised of a set of agents, their current local states, together
50 with a communication topology (which represents which agents are within radio range). A
51 transition represents the effect of one agent sending a message to its neighbours.

52 Parameterized verification of broadcast networks amounts to checking a given property
53 independently of the initial configuration, and in particular independently of the number
54 of agents and communication topology. A natural property one can be interested in is
55 coverability: a state of the broadcast protocol is coverable if some execution leads to a
56 configuration in which one node is in that local state. When considering error states, a
57 positive instance for the coverability problem thus corresponds to a network that can exhibit
58 a bad behaviour.

59 Coverability is undecidable for static broadcast networks [3], *i.e.* when the communication
60 topology is fixed along executions. Decidability can be recovered by relaxing the semantics and
61 allowing non-deterministic reconfigurations of the communication topology. In reconfigurable
62 broadcast networks, coverability of a control state is decidable in PTIME [2]. A simple
63 saturation algorithm allows to compute the set of all states of the broadcast protocol that
64 can be covered.

65 *Cutoff and covering length.* Two important characteristics of positive instances of the
66 coverability problem are the cutoff and the covering length. First, the *cutoff* is the minimal
67 number of agents for which a covering execution exists. The notion of cutoff is particularly
68 relevant for reconfigurable broadcast networks since they enjoy a monotonicity property: if a
69 state can be covered from a configuration, it can also be from any configuration with more
70 nodes. Second, the *covering length* is the minimal number of steps for covering executions. It
71 weighs how fast a network execution can go wrong. Both the cutoff and the covering length
72 are somehow complexity measures for the coverability problem. Surprisingly, no upper nor
73 lower bounds on these values appear in the literature for reconfigurable broadcast networks.

74 *Contributions.* In this paper, we prove a tight linear bound for the cutoff, and a tight
75 quadratic bound for the covering length in reconfigurable broadcast networks. Both are
76 expressed in the number of states of the broadcast protocol. These are obtained by refining
77 the saturation algorithm that computes the set of coverable states, and finely analysing it.

78 Another contribution is to introduce lossy broadcast networks, in which the communication
79 topology is fixed, however errors in message transmission may occur. In contrast with
80 broadcast networks with losses that appear in the literature [4], in our model, message
81 losses happen upon sending, rather than upon reception. This makes a crucial difference:
82 reconfiguration of the communication topology can easily be encoded by losses upon reception,
83 whereas it is not obvious for losses upon sending. Perhaps surprisingly, we prove that the set
84 of states that can be covered in reconfigurable semantics agrees with the one in static lossy
85 semantics. Using the same refined saturation algorithm, we prove that same tight bounds
86 hold for lossy broadcast networks: the cutoff is linear, and the covering length is quadratic
87 (in the number of states of the broadcast protocol). The two semantics thus appear quite
88 similar, yet, we show that the reconfigurable semantics can be linearly more succinct (in
89 terms of number of nodes) than the lossy semantics.

90 Finally, we study a natural decision problem related to the cutoff: decide whether a
91 state is coverable (in either semantics) with a fixed number of nodes. We prove it to be
92 NP-complete.

93 *Outline.* In Section 2, we define the broadcast networks, with static, reconfigurable and lossy
 94 semantics. In Section 3, we present our tight bounds for cutoff and covering length. In
 95 Section 4, we show our succinctness result. In Section 5, we give our NP-completeness result.

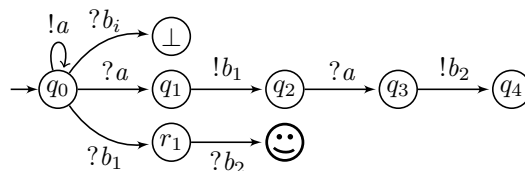
96 2 Broadcast networks

97 2.1 Static broadcast networks

98 ► **Definition 1.** A broadcast protocol is a tuple $\mathcal{P} = (Q, I, \Sigma, \Delta)$ where Q is a finite set
 99 of control states; $I \subseteq Q$ is the set of initial control states; Σ is a finite alphabet; and
 100 $\Delta \subseteq (Q \times \{!a, ?a \mid a \in \Sigma\} \times Q)$ is the transition relation.

101 For ease of readability, we often write $q \xrightarrow{!a} q'$ (resp. $q \xrightarrow{?a} q'$) for $(q, !a, q') \in \Delta$
 102 (resp. $(q, ?a, q') \in \Delta$). We assume all broadcast networks to be complete for receptions: for
 103 every $q \in Q$ and $a \in \Sigma$, there exists q' such that $q \xrightarrow{?a} q'$.

104 A broadcast protocol is represented in Figure 1. In this example and in the whole paper,
 105 for concision purposes, we assume that if the reception of a message is unspecified from some
 106 state, it implicitly represents a self-loop. For example here, from q_1 , receiving a leads to q_1
 again.



107 ■ **Figure 1** Example of a broadcast protocol.

108 Broadcast networks involve several copies, or *nodes*, of the same broadcast protocol \mathcal{P} . A
 109 configuration is an undirected graph whose vertices are labelled with a state of Q . Transitions
 110 between configurations happen by broadcasts from a node to its neighbours.

111 Formally, given a broadcast protocol $\mathcal{P} = (Q, I, \Sigma, \Delta)$, a *configuration* is an undirected
 112 graph $\gamma = (N, E, L)$ where N is a finite set of nodes; $E \subseteq N \times N$ is a symmetric and irreflexive
 113 relation describing the set of edges; finally, $L: N \rightarrow Q$ is the labelling function. We let $\Gamma(\mathcal{P})$
 114 denote the (infinite) set of Q -labelled graphs. Given a configuration $\gamma \in \Gamma(\mathcal{P})$, we write
 115 $n \sim n'$ whenever $(n, n') \in E$ and we let $\text{Neigh}_\gamma(n) = \{n' \in N \mid n \sim n'\}$ be the neighbourhood
 116 of n , *i.e.* the set of nodes adjacent to n . Finally $L(\gamma)$ denotes the set of labels appearing in
 117 nodes of γ . A configuration (N, E, L) is called *initial* if $L(N) \subseteq I$.

118 The operational semantics of a static broadcast network for a given broadcast protocol \mathcal{P}
 119 is an infinite-state transition system $\mathcal{T}(\mathcal{P})$. Intuitively, each node of a configuration runs
 120 protocol \mathcal{P} , and may send/receive messages to/from its neighbours. From a configuration
 121 $\gamma = (N, E, L)$, there is a step to $\gamma' = (N', E', L')$ if $N' = N$, $E' = E$, and there exists $n \in N$
 122 and $a \in \Sigma$ such that $(L(n), !a, L'(n)) \in \Delta$, and for every $n' \in N$, if $n' \in \text{Neigh}_\gamma(n)$, then
 123 $(L(n'), ?a, L'(n')) \in \Delta$, otherwise $L'(n') = L(n')$: a step reflects how nodes evolve when one of
 124 them broadcasts a message to its neighbours. We write $\gamma \xrightarrow{n, !a}_s \gamma'$, or simply $\gamma \rightarrow_s \gamma'$ (the s
 125 subscript emphasizes that the communication topology is *static*).

126 An *execution* of the static broadcast network is a sequence $\rho = (\gamma_i)_{0 \leq i \leq r}$ of configurations
 127 (N, E, L_i) such that γ_0 is an initial configuration, and for every $0 \leq i < r$, $\gamma_i \rightarrow_s \gamma_{i+1}$. We

128 write $\#\text{nodes}(\rho)$ for the number of nodes in γ_0 , $\#\text{steps}(\rho)$ for the number r of steps along ρ ,
 129 and for any node $n \in \mathbf{N}$, $\#\text{steps}(\rho, n)$ for the number of broadcasts, called the *active length*, of
 130 node n along ρ . Note that, along an execution, the number of nodes and the communication
 131 topology are fixed. The set of all static executions is denoted $\text{Exec}_s(\mathcal{P})$.

132 Coverability problem.

133 Given a broadcast protocol \mathcal{P} and a subset of target states $F \subseteq Q$, we write $\text{COVER}_s(\mathcal{P}, F)$
 134 for the set of all *covering* executions, that is, executions that visit a configuration with a
 135 node labelled by a state in F :

$$136 \quad \text{COVER}_s(\mathcal{P}, F) = \{(\gamma_i)_{0 \leq i \leq r} \in \text{Exec}_s(\mathcal{P}) \mid L(\gamma_r) \cap F \neq \emptyset\}.$$

137 The *coverability problem* is a decision problem that takes a broadcast protocol \mathcal{P} and a subset
 138 of target states F as inputs, and outputs whether $\text{COVER}_s(\mathcal{P}, F)$ is nonempty. For broadcast
 139 networks, the coverability problem is a parameterized verification problem, since the number
 140 of initial configurations is infinite. It is known that coverability is undecidable for static
 141 broadcast networks [3], since one can use the communication topology to build chains that
 142 may encode values of counters, and hence simulate Minsky machines [10].

143 If the broadcast protocol \mathcal{P} allows to cover the subset F , we define the *cutoff* as the
 144 minimal number of nodes required in an execution to cover F . Similarly, we define the
 145 *covering length* as the length of a shortest finite execution covering F . Those values are
 146 important to characterize the complexity of a broadcast protocol: assuming a safe set of
 147 states, coverability of the complement set represents bad behaviours, and cutoff and covering
 148 length measure the size of minimal witnesses for violation of the safety property.

149 2.2 Reconfigurable broadcast networks

150 To circumvent the undecidability of coverability for static broadcast networks, one attempt is
 151 to introduce non-deterministic reconfiguration of the communication topology. This solution
 152 also allows one to model arbitrary mobility of the nodes, which is meaningful, *e.g.* for mobile
 153 ad-hoc networks [3].

154 Under this semantics, configurations are the same as under the static semantics. Transi-
 155 tions between configurations however are enhanced by the ability to modify the communi-
 156 cation topology before performing a broadcast. Formally, from a configuration $\gamma = (\mathbf{N}, \mathbf{E}, \mathbf{L})$,
 157 there is a step to $\gamma' = (\mathbf{N}', \mathbf{E}', \mathbf{L}')$ if $\mathbf{N}' = \mathbf{N}$, and there exists $n \in \mathbf{N}$ and $a \in \Sigma$ such that
 158 $(\mathbf{L}(n), !a, \mathbf{L}'(n)) \in \Delta$, and for every $n' \in \mathbf{N}$, if $n' \in \text{Neigh}_{\gamma'}(n)$, then $(\mathbf{L}(n), ?a, \mathbf{L}'(n')) \in \Delta$,
 159 otherwise $\mathbf{L}'(n') = \mathbf{L}(n')$: a step thus reflects that the communication topology may change
 160 from \mathbf{E} to \mathbf{E}' followed by the broadcast of a message from a node to its neighbours in the
 161 new topology. We write $\gamma \xrightarrow{n, !a}_{\mathbf{r}} \gamma'$, or simply $\gamma \rightarrow_{\mathbf{r}} \gamma'$.

162 Similarly to the static case, we write $\text{Exec}_{\mathbf{r}}(\mathcal{P})$ and $\text{COVER}_{\mathbf{r}}(\mathcal{P}, F)$ for, respectively the
 163 set of all reconfigurable executions in \mathcal{P} , and the set of all reconfigurable executions in \mathcal{P}
 164 that cover F . We will also use the same notations $\#\text{nodes}(\rho)$, $\#\text{steps}(\rho)$ and $\#\text{steps}(\rho, n)$ as
 165 in the static case.

166 Figure 7 (with $n = 2$) gives an example of reconfigurable execution for the broadcast
 167 protocol of Figure 1 (which covers \odot). Note that the communication topology indeed evolves
 168 along the execution. Here the colored nodes broadcast a message in the step leading to the
 169 next configuration.

170 A noticeable property of reconfigurable broadcast networks is the following copycat
 171 property. Such a monotonicity property was originally shown in [7] for asynchronous shared-
 172 memory systems, and it also applies to our context.

173 **► Proposition 2** (Copycat for reconfigurable semantics). *Given $\rho : \gamma_0 \rightarrow_{\mathbf{x}} \gamma_1 \cdots \rightarrow_{\mathbf{x}} \gamma_s$ an*
 174 *execution, with $\gamma_s = (\mathbf{N}, \mathbf{E}, \mathbf{L})$, for every $q \in \mathbf{L}(\gamma_s)$, for every $n^q \in \mathbf{N}$ such that $\mathbf{L}(n^q) = q$,*
 175 *there exists $t \in \mathbb{N}$ and an execution $\rho' : \gamma'_0 \rightarrow_{\mathbf{x}} \gamma'_1 \cdots \rightarrow_{\mathbf{x}} \gamma'_t$ with $\gamma'_t = (\mathbf{N}', \mathbf{E}', \mathbf{L}')$ such that*
 176 *$|\mathbf{N}'| = |\mathbf{N}| + 1$, there is an injection $\iota : \mathbf{N} \rightarrow \mathbf{N}'$ with for every $n \in \mathbf{N}$, $\mathbf{L}'(\iota(n)) = \mathbf{L}(n)$, and for*
 177 *the extra node $n_{\text{fresh}} \in \mathbf{N}' \setminus \iota(\mathbf{N})$, $\mathbf{L}'(n_{\text{fresh}}) = q$, and $\#steps(\rho', n_{\text{fresh}}) = \#steps(\rho, n^q)$.*

178 Intuitively, the new node n_{fresh} will copy the moves of node n^q : it performs the same broadcasts
 179 (but to nobody) and receives the same messages. More precisely, when n^q broadcasts in ρ , it
 180 does so also in ρ' and then we disconnect all the nodes and n_{fresh} repeats the broadcast (no
 181 other node is affected because of the disconnection); when n^q receives a message in ρ , we
 182 connect n_{fresh} to the same neighbours as n^q (i.e., $\iota(n) \sim' n_{\text{fresh}}$ if and only if $n \sim n^q$) so that
 183 n_{fresh} also receives the same message in ρ' .

184 Relying on the copycat property, when reconfigurations are allowed, the coverability
 185 problem becomes decidable and solvable in polynomial time.

186 **► Theorem 3** ([2]). *Coverability is decidable in PTIME for reconfigurable broadcast networks.*

187 More precisely, a simple saturation algorithm allows one to compute in polynomial time,
 188 the set of all states that can be covered. Despite this complexity result, to the best of
 189 our knowledge, no bounds on the cutoff or length of witness executions are stated in the
 190 literature.

191 2.3 Broadcast networks with messages losses

192 Communication failures were studied for broadcast networks, assuming non-deterministic
 193 message losses could happen: when a message is broadcast, some of the neighbours of the
 194 sending node may not receive it [4]. As observed by the authors, the coverability problem
 195 for such networks easily reduces to the coverability problem in reconfigurable networks
 196 by considering a complete topology, and message losses are simulated by reconfigurations.
 197 Thus, message losses upon reception are equivalent to reconfiguration of the communication
 198 topology.

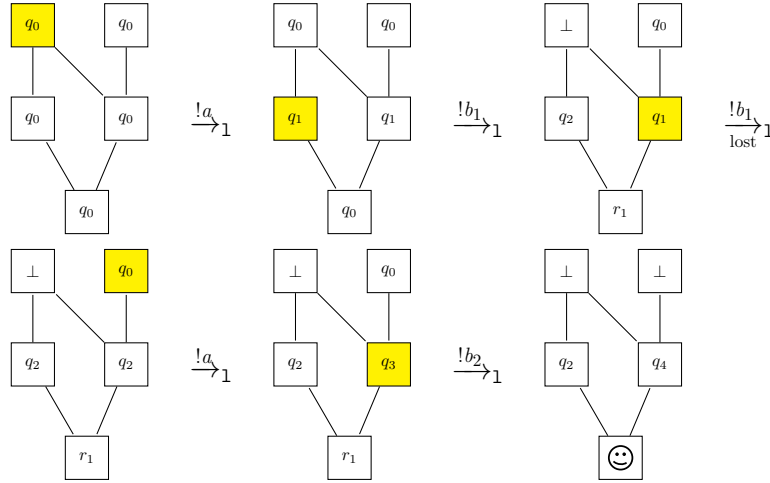
199 We propose an alternative semantics here: when a message is broadcast, it either reaches
 200 all neighbours of the sending node, or none of them. This is relevant in contexts where
 201 broadcasts are performed in an atomic manner and may fail. In contrast to message losses
 202 upon reception, it is not obvious to simulate arbitrary reconfigurations of the communication
 203 topology with such message losses.

204 Formally, from a configuration $\gamma = (\mathbf{N}, \mathbf{E}, \mathbf{L})$, there is a step to $\gamma' = (\mathbf{N}', \mathbf{E}', \mathbf{L}')$ if $\mathbf{N}' = \mathbf{N}$,
 205 $\mathbf{E}' = \mathbf{E}$ and there exists $n \in \mathbf{N}$ and $a \in \Sigma$ such that $(\mathbf{L}(n), !a, \mathbf{L}'(n)) \in \Delta$, and either (a) for
 206 every $n' \neq n$, $\mathbf{L}'(n') = \mathbf{L}(n')$ (no one has received the message, it has been lost), or (b) if
 207 $n' \in \text{Neigh}_{\gamma'}(n)$, then $(\mathbf{L}(n'), ?a, \mathbf{L}'(n')) \in \Delta$, otherwise $\mathbf{L}'(n') = \mathbf{L}(n')$: a step thus reflects that
 208 the broadcast message may be lost when it is sent. We write $\gamma \xrightarrow{n, !a} \gamma'$ or simply $\gamma \rightarrow_1 \gamma'$.
 209 Similarly to the static and reconfigurable semantics, $\#steps(\rho, n)$ is the number of broadcasts
 210 (including lost ones) by node n along ρ ; and we write $\#nonlost_steps(\rho, n)$ for the number of
 211 successful broadcasts by node n along ρ .

212 For lossy executions also, we use the following notations: $\text{Exec}_1(\mathcal{P})$ and $\text{COVER}_1(\mathcal{P}, F)$.
 213 Any lossy execution can be seen as a reconfigurable execution. Indeed, a lossy execution

214 with communication topology E can be transformed into a reconfigurable one in which
 215 the communication topology of each configuration is either \emptyset or E , depending on whether
 216 the next broadcast is lost or not. Therefore, with slight abuse of notation, we write
 217 $\text{Exec}_1(\mathcal{P}) \subseteq \text{Exec}_r(\mathcal{P})$.

218 Figure 2 gives an example of a lossy execution for the broadcast protocol of Figure 1.
 219 Note that in the third transition, some node indeed performs a lossy broadcast, emphasized
 220 by the subscript “lost”. As before, the colored nodes broadcast a message in the step leading
 221 to the next configuration.



■ **Figure 2** Example of a lossy execution on the protocol from Figure 1.

222 3 Tight bounds for reconfigurable and lossy broadcast networks

223 In this section, we will show tight bounds for the cutoff and the minimal length of a witness
 224 execution for the coverability problem. These hold both for the reconfigurable and the lossy
 225 semantics.

226 3.1 Upper bounds on cutoff and covering length for reconfigurable 227 networks

228 First, we will refine the polytime saturation algorithm of [2], which computes all states
 229 which can be covered in the reconfigurable semantics. We will then show that, based on
 230 the underlying computation, one can construct small witnesses for the two semantics (linear
 231 number of nodes and quadratic number of steps). While it would be enough to show the
 232 result for the lossy semantics (since, given a broadcast protocol \mathcal{P} , $\text{Exec}_1(\mathcal{P}) \subseteq \text{Exec}_r(\mathcal{P})$),
 233 for pedagogical reasons, we provide the two proofs, starting with the simplest one for
 234 reconfigurable semantics.

235 Let us fix for the rest of this section, a protocol $\mathcal{P} = (Q, I, \Sigma, \Delta)$. We slightly modify the
 236 algorithm given in [2] as follows: we include at most one state to the set S in each iteration.
 237 Additionally, we associate a labelling function $c : S \rightarrow \mathbb{N}$ with the set S in every iteration.
 238 More formally, we consider the modification of the previous saturation algorithm as shown
 239 in Algorithm 1.

Algorithm 1 Refined saturation algorithm for coverability

```

1:  $S := I; c(S) := |I|; S' := \emptyset$ 
2: while  $S \neq S'$  do
3:    $S' := S; c := c(S)$ 
4:   if  $\exists (q_1, !a, q_2) \in \Delta$  s.t.  $q_1 \in S'$  and  $q_2 \notin S'$  then
5:      $S := S \cup \{q_2\}; c(S) := c + 1$ 
6:   else if  $\exists (q_1, !a, q_2) \in \Delta$  and  $(q'_1, ?a, q'_2) \in \Delta$  s.t.  $q_1, q_2, q'_1 \in S'$  and  $q'_2 \notin S'$  then
7:      $S := S \cup \{q'_2\}; c(S) := c + 2$ 
8:   end if
9: end while
10: return  $S$ 

```

240 In Algorithm 1, the variable c counts the number of nodes that are sufficient to cover the
241 current set S , as we will prove later.

242 ► **Lemma 4** ([2]). *Algorithm 1 terminates and returns the set of coverable states. In*
243 *particular, $\text{COVER}_x(\mathcal{P}, F) \neq \emptyset$ iff $F \cap S \neq \emptyset$.*

244 Let S_0, S_1, \dots, S_m be the sets after each iteration of the algorithm, with $S_0 = I$ and
245 $S_m = S$. We fix an ordering on the states in S on the basis of insertion in S : for all $1 \leq i \leq m$,
246 q_i is such that $q_i \in S_i \setminus S_{i-1}$. In the following, we show the desired upper bounds, proving
247 that there exists an execution of size $O(n)$ and length $O(n^2)$ covering at the same time all
248 states of S_m .

249 ► **Theorem 5.** *Let $\mathcal{P} = (Q, I, \Sigma, \Delta)$ be a broadcast protocol, and $F \subseteq Q$. If $\text{COVER}_x(\mathcal{P}, F) \neq$
250 \emptyset (that is, if $F \cap S \neq \emptyset$), then there exists $\rho \in \text{COVER}_x(\mathcal{P}, F)$ with $\#\text{nodes}(\rho) \leq 2|Q|$ and
251 $\#\text{steps}(\rho) \leq 2|Q|^2$.*

252 Theorem 5 is a consequence of the following Lemma.

253 ► **Lemma 6.** *For every step i of Algorithm 1, there exists an initial configuration γ_0 , a*
254 *configuration γ and a reconfigurable execution $\rho : \gamma_0 \xrightarrow{*}_x \gamma$ such that $L(\gamma) = S_i$, $\#\text{nodes}(\rho) =$
255 $c(S_i)$, and $\max_n \#\text{steps}(\rho, n) \leq i$.*

256 **Proof.** The lemma is proved by induction on i . The base case $i = 0$ is obvious: take the
257 initial configuration γ_0 with $|I|$ nodes, and label each node with a different initial state; its
258 size is $|I|$, and the length of the execution is 0, hence so is the maximum active length.

259 To prove the induction step, we distinguish two cases: depending on whether q_{i+1} was
260 added as the target state of a broadcast transition $q \xrightarrow{!a}$ for some $q \in S_i$; or whether q_{i+1} is
261 the target state of a reception from some $q \in S_i$ with matching broadcast between two states
262 already in S_i .

263 *Case 1:* There exists $q \in S_i$ with $q \xrightarrow{!a} q_{i+1}$. We apply the induction hypothesis to
264 step i , and exhibit an execution $\rho : \gamma_0 \xrightarrow{*}_x \gamma$ such that $L(\gamma) = S_i$, $\#\text{nodes}(\rho) = c(S_i)$ and
265 $\max_n \#\text{steps}(\rho, n) \leq i$. Applying the copycat property (see Proposition 2), we construct an
266 execution $\rho' : \gamma'_0 \xrightarrow{*}_x \gamma'$ such that γ'_0 has one node more than γ_0 , and, focusing on the nodes
267 (since we are in a reconfigurable setting, edges in the configuration are not important), γ'
268 coincides with γ , with an extra node n labelled by q . We then disconnect all nodes and
269 extend with a transition $\gamma' \xrightarrow{n, !a}_x \gamma''$, which makes only progress node n from q to q_{i+1} ; the
270 resulting execution is denoted ρ'' . Then:

271 **1.** $L(\gamma'') = S_i \cup \{q_{i+1}\} = S_{i+1}$,

- 272 2. $\#nodes(\rho'') = c(S_i) + 1 = c(S_{i+1})$,
 273 3. $\max_n \#steps(\rho'', n) \leq \max_n \#steps(\rho, n) + 1 \leq i + 1$; Indeed, the active length of the
 274 copycat node along ρ' coincides with the active length of some existing node along ρ , and
 275 it is increased only by 1 in ρ'' .

276 This proves the induction step in the first case.

277 *Case 2:* There exists $q, q', q'' \in S_i$ with $q \xrightarrow{?a} q_{i+1}$ and $q' \xrightarrow{!a} q''$. The idea is similar to
 278 the previous case, but one should apply the copycat property twice, to both q and q' . We
 279 formalize this.

280 We apply the induction hypothesis to step i , and exhibit an execution $\rho : \gamma_0 \xrightarrow{*}_r \gamma$
 281 such that $L(\gamma) = S_i$, $\#nodes(\rho) = c(S_i)$ and $\max_n \#steps(\rho, n) \leq i$. Applying the copycat
 282 property (see Proposition 2) twice, to both q and q' , we construct an execution $\rho' : \gamma'_0 \xrightarrow{*}_r \gamma'$
 283 such that γ'_0 has two nodes more than γ_0 , and, focusing on the nodes, γ' coincides with γ ,
 284 with one extra node n labelled by q and one extra node n' labelled by q' . We then connect
 285 nodes n and n' and disconnect all other nodes, and extend with a transition $\gamma' \xrightarrow{n', !a}_r \gamma''$;
 286 this makes node n progress from q to q_{i+1} and node n' progress from q' to q'' ; all other nodes
 287 are unchanged; the resulting execution is denoted ρ'' . Then:

- 288 1. $L(\gamma'') = S_i \cup \{q'', q_{i+1}\} = S_{i+1}$ since $q'' \in S_i$,
 289 2. $\#nodes(\rho'') = c(S_i) + 2 = c(S_{i+1})$,
 290 3. $\max_n \#steps(\rho'', n) \leq \max_n \#steps(\rho, n) + 1 \leq i + 1$; Indeed the active length of any of
 291 the copycat node along ρ' coincides with the active length of some existing node along ρ ,
 292 and it is increased by at most 1 in ρ'' .

293 This proves the induction step in the second case, which allows to conclude the proof of the
 294 lemma. \blacktriangleleft

295 To conclude the proof of Theorem 5, we recall that Algorithm 1 is sound and complete:
 296 S_m is the set of states that can be covered. Hence, from Lemma 6, we deduce that if
 297 $\text{COVER}_r(\mathcal{P}, F) \neq \emptyset$, then there is $\rho \in \text{COVER}_r(\mathcal{P}, F)$ such that:

- 298 1. $L(\gamma) = S_m$;
 299 2. $\#nodes(\rho) = c(S_m) \leq |I| + 2m \leq |I| + 2(|Q| - |I|) = 2|Q| - |I|$;
 300 3. $\max_n \#steps(\rho, n) \leq m \leq |Q| - |I|$.

301 Therefore $\#steps(\rho) \leq (\#nodes(\rho)) \cdot (\max_n \#steps(\rho, n)) \leq 2|Q|^2$, so that we established
 302 the desired bounds for Theorem 5.

303 3.2 Upper bounds on cutoff and covering length for lossy networks

304 Perhaps surprisingly, Algorithm 1 also computes the set of states that can be covered by
 305 lossy executions. Concerning coverable states, the reconfigurable and lossy semantics thus
 306 agree. In Section 4, we will show that reconfigurable covering executions can be linearly
 307 more succinct than lossy covering executions.

308 **► Lemma 7.** *Algorithm 1 returns the set of coverable states for lossy broadcast networks. In*
 309 *particular, $\text{COVER}_1(\mathcal{P}, F) \neq \emptyset$ iff $F \cap S \neq \emptyset$.*

310 Indeed, we have $\text{Exec}_1(\mathcal{P}) \subseteq \text{Exec}_r(\mathcal{P})$. Therefore $\text{COVER}_1(\mathcal{P}, F) \neq \emptyset$ implies $\text{COVER}_r(\mathcal{P}, F) \neq$
 311 \emptyset and by Lemma 4, we conclude $F \cap S \neq \emptyset$. The other direction of Lemma 7 is a consequence
 312 of the following theorem.

313 **► Theorem 8.** *Let $\mathcal{P} = (Q, I, \Sigma, \Delta)$ be a broadcast protocol, and $F \subseteq Q$. If $S \cap F \neq \emptyset$, then*
 314 *there exists $\rho \in \text{COVER}_1(\mathcal{P}, F)$ with $\#nodes(\rho) \leq 2|Q|$ and $\#steps(\rho) \leq 2|Q|^2$.*

315 Before going to the proof of Theorem 8, we show a copycat property for the lossy
316 broadcast networks, as a counterpart of Proposition 2 for the lossy semantics. Since the
317 communication topology is static in lossy networks, the following proposition explicitly relates
318 the communication topologies in the initial execution and its copycat extension.

319 ► **Proposition 9** (Copycat for lossy semantics). *Given $\rho : \gamma_0 \rightarrow_1 \gamma_1 \cdots \rightarrow_1 \gamma_r$ an execution,*
320 *with $\gamma_r = (\mathbf{N}, \mathbf{E}, \mathbf{L})$, for every $q \in \mathbf{L}(\gamma_r)$, for every $n^q \in \mathbf{N}$ such that $\mathbf{L}(n^q) = q$, there*
321 *exists $s \in \mathbb{N}$ and an execution $\rho' : \gamma'_0 \rightarrow_1 \gamma'_1 \cdots \rightarrow_1 \gamma'_s$ with $\gamma'_s = (\mathbf{N}', \mathbf{E}', \mathbf{L}')$ such that*
322 *$|\mathbf{N}'| = |\mathbf{N}| + 1$, there is an injection $\iota : \mathbf{N} \rightarrow \mathbf{N}'$ with for every $n \in \mathbf{N}$, $\mathbf{L}'(\iota(n)) = \mathbf{L}(n)$, and*
323 *for the extra node $n_{\text{fresh}} \in \mathbf{N}' \setminus \iota(\mathbf{N})$, $\mathbf{L}'(n_{\text{fresh}}) = q$, for every $n \in \mathbf{N}$, $n_{\text{fresh}} \sim' \iota(n)$ iff $n^q \sim n$,*
324 *$\#steps(\rho', n_{\text{fresh}}) = \#steps(\rho, n^q)$, and $\#nonlost_steps(\rho', n_{\text{fresh}}) = 0$.*

325 **Proof.** First notice that, from our definition of lossy semantics, the topology should be the
326 same in γ_0 and in γ_r , hence we can write $\gamma_0 = (\mathbf{N}, \mathbf{E}, \mathbf{L}_0)$, and more generally, for every
327 i , $\gamma_i = (\mathbf{N}, \mathbf{E}, \mathbf{L}_i)$. Define \mathbf{N}' as a finite set such that $|\mathbf{N}'| = |\mathbf{N}| + 1$, and fix an injection
328 $\iota : \mathbf{N} \rightarrow \mathbf{N}'$. Write n_{fresh} for the unique element of $\mathbf{N}' \setminus \iota(\mathbf{N})$. Set $\mathbf{L}'_0(\iota(n)) = \mathbf{L}_0(n)$ for every
329 $n \in \mathbf{N}$, and $\mathbf{L}'_0(n_{\text{fresh}}) = \mathbf{L}_0(n^q)$. Define the edge relation \mathbf{E}' by its induced edge relation \sim'
330 such that $\iota(n) \sim' \iota(n')$ iff $n \sim n'$, and $n_{\text{fresh}} \sim' \iota(n')$ iff $n^q \sim n'$.

331 The idea will then be to make n_{fresh} follow what n^q is doing. Roughly, if n^q is receiving a
332 message to progress, then we will connect n_{fresh} to a relevant node to also receive the message;
333 if n^q is broadcasting a message, then we will make n_{fresh} broadcast a message and lose, so
334 that no other node is impacted.

335 Formally, we will show by induction on i that for every $0 \leq i \leq r$, there is an execution
336 $\rho'_i : \gamma'_0 \rightarrow_1 \gamma'_1 \cdots \rightarrow_1 \gamma'_{f(i)}$ for some $f(i)$, such that $\mathbf{L}'_i(\iota(n)) = \mathbf{L}_i(n)$ for every $n \in \mathbf{N}$ and
337 $\mathbf{L}'_i(n_{\text{fresh}}) = \mathbf{L}_i(n^q)$. The initial case $i = 0$ is obvious. We then assume that we have constructed
338 a relevant ρ'_i for some $i < r$, and we will extend it to ρ'_{i+1} as follows. We make a case
339 distinction depending on the nature of the step $\gamma_i \rightarrow_1 \gamma_{i+1}$:

- 340 ■ Assume $\gamma_i \xrightarrow{n, !a}_1 \gamma_{i+1}$ is a broadcast message with $n^q \neq n$, then ρ'_{i+1} is obtained by
341 extending ρ'_i with the broadcast $\gamma'_{f(i)} \xrightarrow{\iota(n), !a} \gamma'_{f(i)+1}$, with the condition that it should
342 be lost if and only if it was lost in the original execution. For checking correctness, we
343 distinguish two cases:
 - 344 ■ the broadcast message was not lost, and $n^q \sim n$. Then, it is the case that $n_{\text{fresh}} \sim' \iota(n)$,
345 hence n_{fresh} also receives the message. By resolving properly the nondeterminism, we
346 can make the label of n_{fresh} become the same as the label of n^q in $\gamma'_{f(i)+1}$. Note also
347 that all nodes in $\iota(\mathbf{N})$ can progress to the same states as those of \mathbf{N} in γ_{i+1} ;
 - 348 ■ the broadcast message was lost, or $n^q \not\sim n$, then it is the case that the label of n^q has
349 not been changed in $\gamma_i \xrightarrow{n, !a}_1 \gamma_{i+1}$, and so will the label of the fresh node in $\gamma'_{f(i)}$.
- 350 ■ Assume $\gamma_i \xrightarrow{n^q, !a}_1 \gamma_{i+1}$ is a broadcast message, then we extend ρ'_i with the two steps
351 $\gamma'_{f(i)} \xrightarrow{\iota(n^q), !a} \gamma'_{f(i)+1} \xrightarrow{n_{\text{fresh}}, !a} \gamma'_{f(i)+2}$ (resolving nondeterminism in a similar way as in
352 $\gamma_i \xrightarrow{n^q, !a}_1 \gamma_{i+1}$), and we make the last broadcast lossy whereas the broadcast from $\iota(n^q)$
353 is lossy if and only if it was lossy in $\gamma_i \rightarrow_1 \gamma_{i+1}$.

354 This concludes the induction. Notice that in the constructed execution, node n_{fresh} does not
355 make any real sending. ◀

356 For any configuration $\gamma = (\mathbf{N}, \mathbf{E}, \mathbf{L})$ and a node n , we write $\mathbf{L}(n) = \times$ if n is not important
357 anymore in the execution, in other words all the required conditions in γ' such that $\gamma \xrightarrow{*}_1 \gamma'$
358 are still satisfied whatever $\mathbf{L}(n)$ is.

359 Recall the saturation algorithm and the ordering of the sets and the states: $S_0 =$
 360 $I, S_1, \dots, S_m = S$ are the sets after each iteration and q_i is the state such that $q_i \in S_i \setminus S_{i-1}$
 361 for all $1 \leq i \leq m$. We will refine the construction from the proof of Lemma 6 (in the context
 362 of reconfigurable broadcast networks), and build inductively a lossy execution covering all
 363 states in S_i . Since the topology is static, some nodes which have “finished their jobs” will
 364 remain connected to other nodes, and may therefore continue to change states (contrary to
 365 Lemma 6 where they could be fully disconnected). Hence, in every such execution, every
 366 state $q \in S_i$ (which is then covered by the execution) will have a main corresponding node,
 367 whose label will remain q . All nodes which are not the main node of a state will be assigned
 368 \times , since their labels will become meaningless.

369 We formalize this idea in the lemma below. However, for better understanding, we
 370 also illustrate this inductive construction of a witness execution in Figure 4 on the simple
 371 broadcast protocol from Figure 3. Configurations are represented vertically: they involve 10
 372 nodes, and the communication topology is given for the first configuration only, for the sake
 373 of readability. To save space, several broadcasts (of the same message type, from different
 374 nodes) may happen in a *macrostep* that merges several steps. This is for instance the case in
 375 the first macrostep, where a is being broadcast from the node in set S_1 , as well as from the
 376 first node in set S_2 . Dashed arrows are used to represent that a node is not involved in some
 377 macrostep and thus stays in the same state. In the execution, the nodes that are performing
 378 a real broadcast are colored yellow, the ones which receive a message are colored gray, and
 379 blue nodes indicate the main nodes for the coverable states.

380 ► **Lemma 10.** *For every step i of the refined saturation algorithm, there exists a configuration*
 381 $\gamma = (\mathbf{N}, \mathbf{E}, \mathbf{L})$ *and an execution $\rho : \gamma_0 \xrightarrow{*} \gamma$ such that:*

- 382 ■ $\mathbf{L}(\gamma) \setminus \{\times\} = S_i$ *and* $\#\text{nodes}(\rho) = c(S_i)$,
- 383 ■ $\max_n \#\text{steps}(\rho, n) \leq i$ *and* $\max_n \#\text{nonlost_steps}(\rho, n) \leq 1$,
- 384 ■ *for every* $q \in S_i$, *there exists* $n_q^{\text{main}} \in \mathbf{N}$ *such that*
 - 385 ■ $\mathbf{L}(n_q^{\text{main}}) = q$ *and* $\#\text{nonlost_steps}(\rho, n_q^{\text{main}}) = 0$,
 - 386 ■ $n_q^{\text{main}} \sim n$ *implies* $\mathbf{L}(n) = \times$, *and if* $n \notin \{n_q^{\text{main}} \mid q \in S_i\}$, *then* $\mathbf{L}(n) = \times$.

387 **Proof.** We do the proof by induction on i . The case $i = 0$ is obvious, by picking one
 388 main node per initial state in I , and by disconnecting all nodes; hence forming an initial
 389 configuration satisfying all the requirements.

390 To prove the induction step, we distinguish two cases: depending on whether q_{i+1} was
 391 added as the target state of a broadcast action $!a$ from some $q \in S_i$; or whether q_{i+1} is the
 392 target state of a reception from some $q \in S_i$ with matching broadcast between two states
 393 already in S_i .

394 *Case 1:* There exists $q \in S_i$ with $q \xrightarrow{!a} q_{i+1}$. We apply the induction hypothesis to step
 395 i , and exhibit the various elements of the statement. Applying the copycat property for
 396 lossy broadcast systems (that is, Proposition 9) with node n_q^{main} , we build an execution
 397 $\rho' : \gamma'_0 \xrightarrow{*} \gamma'$ such that $\gamma' = (\mathbf{N}, \mathbf{E}, \mathbf{L}')$ with $|\mathbf{N}'| = |\mathbf{N}| + 1$, and an appropriate injection ι .
 398 The fresh node n_{fresh} is connected to nodes to which n_q^{main} was connected before; hence, by
 399 induction hypothesis, it is only connected to nodes labelled with \times . Then we extend ρ' with
 400 $\gamma' \xrightarrow{n_{\text{fresh}}, !a} \gamma''$ and lose the message (this is for condition $\#\text{nonlost_steps}(\rho, n_q^{\text{main}}) = 0$ to be
 401 satisfied). We declare $n_{q_{i+1}}^{\text{main}} = n_{\text{fresh}}$. All requirements for γ'' are easily checked to be satisfied
 402 (when a node is labelled with \times in γ' , then it remains labelled by \times in γ'').

403 *Case 2:* There exist $q, q', q'' \in S_i$ such that $q \xrightarrow{?a} q_{i+1}$ and $q' \xrightarrow{!a} q''$. We apply the
 404 induction hypothesis to step i , and exhibit the various elements of the statement. Applying
 405 twice the copycat property (that is, Proposition 9), once with node n_q^{main} and once with

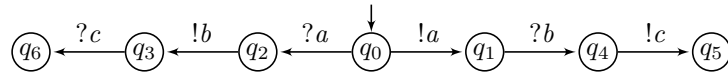


Figure 3 Illustrating example for the saturation algorithm.

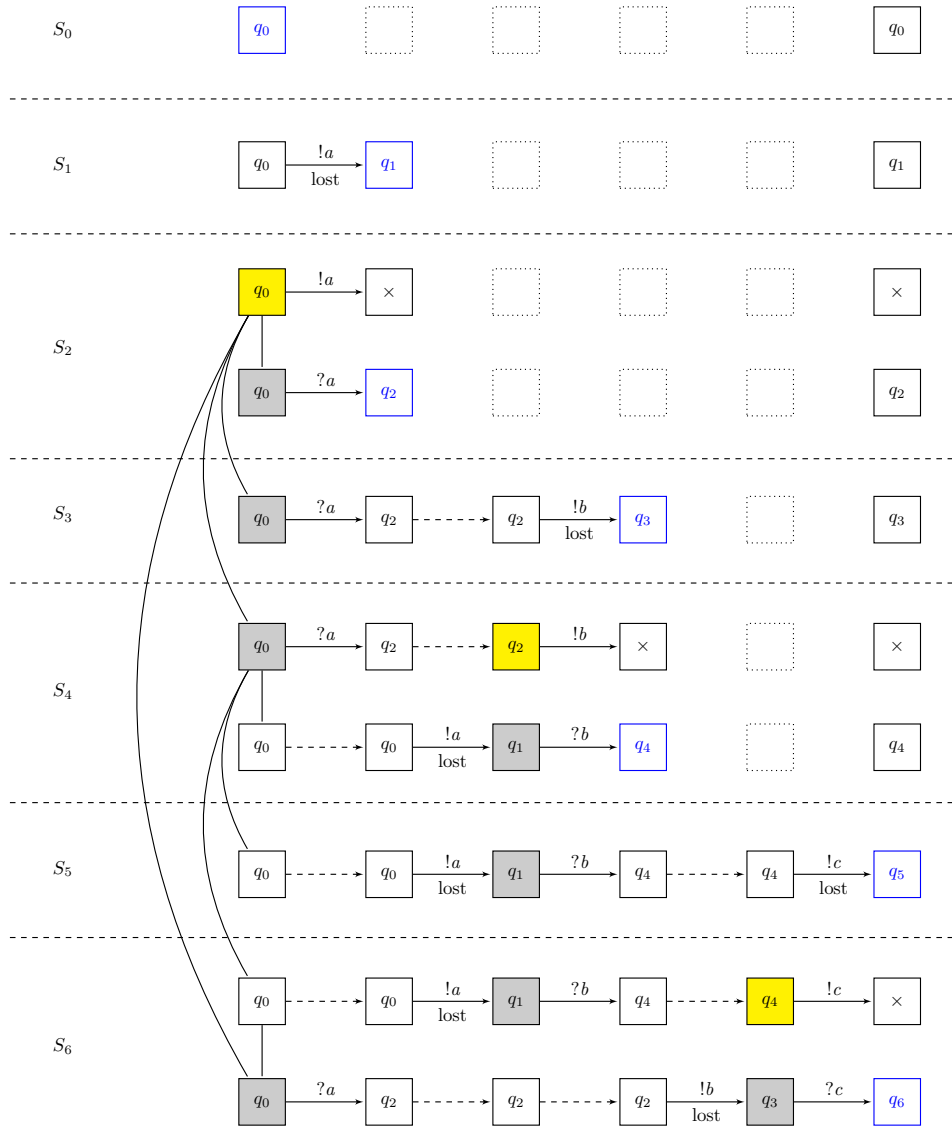
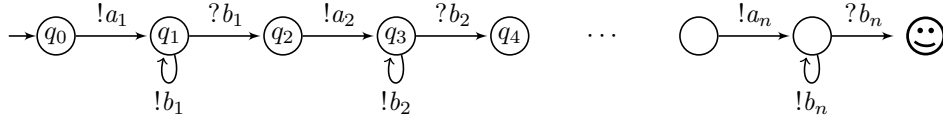


Figure 4 Applying saturation algorithm on protocol in Figure 3 in lossy semantics. Configurations are represented vertically; for readability, macrosteps merge several broadcasts.

406 node n_q^{main} , we build an execution $\rho' : \gamma'_0 \xrightarrow{*} \gamma'$ such that $\gamma' = (N, E, L')$ with $|N'| = |N| + 2$,
 407 and an appropriate injection ι . The two fresh nodes n_{fresh} and n'_{fresh} are only connected
 408 to \times -nodes in γ' (by induction hypothesis on n_q^{main} and n_q^{main} respectively). We transform
 409 γ'_0 into γ''_0 by connecting the two nodes n_{fresh} and n'_{fresh} . By Proposition 9, we know that
 410 those two nodes don't perform any real sending (*i.e.*, $\#\text{nonlost_steps}(\rho', n_{\text{fresh}}) = 0$ and
 411 $\#\text{nonlost_steps}(\rho', n'_{\text{fresh}}) = 0$), hence this new connection will not affect the labels of the
 412 nodes, and we can safely apply the same transitions as in ρ' from γ''_0 to get an execution



■ **Figure 5** Broadcast protocol with linear cutoff and quadratic covering length.

413 $\rho'' : \gamma_0'' \xrightarrow{*} \gamma_1''$, where γ_1'' coincides with γ_1' , with an extra connection between nodes n_{fresh}
 414 and n'_{fresh} . Then, we extend ρ'' with $\gamma_1'' \xrightarrow{n'_{\text{fresh}}, !a} \gamma_2''$. We assume it is a real sending, hence:
 415 node n_{fresh} can progress from state q to q_{i+1} , and node n'_{fresh} can progress from q' to q'' . All
 416 other nodes which are connected to n'_{fresh} are labelled by \times in γ_2'' , hence cannot be really
 417 affected by that sending. We relabel n'_{fresh} to \times , and declare $n_{q_{i+1}}^{\text{main}} = n_{\text{fresh}}$. The expected
 418 conditions of the statement are easily checked to be satisfied by this new execution. ◀

419 Bounds are then obtained similarly to the reconfigurable case, see page 8.

420 3.3 Matching lower bounds for reconfigurable and lossy networks

421 In this section, we show that the linear bound on the cutoff and the quadratic bound on the
 422 length of witness executions are tight, both for the reconfigurable and the lossy broadcast
 423 networks.

424 ► **Theorem 11.** *There exists a family of broadcast protocols $(\mathcal{P}_n)_n$ with $\mathcal{P}_n = (Q_n, I_n, \Sigma_n, \Delta_n)$,
 425 and target states $F_n \subseteq Q_n$ with $|Q_n| \in O(n)$, such that for every n , $\text{COVER}_r(\mathcal{P}_n, F_n) \neq \emptyset$,
 426 $\text{COVER}_l(\mathcal{P}_n, F_n) \neq \emptyset$, and any witness reconfigurable or lossy execution has size $O(n)$ and
 427 length $O(n^2)$.*

428 **Proof.** Consider \mathcal{P}_n , as depicted in Figure 5 with $2n+1$ states and $F_n = \{\ominus\}$. Any covering
 429 reconfigurable execution involves at least $n+1$ nodes, and has at least $\frac{n^2+5n}{2}$ steps. Indeed,
 430 intuitively, the process responsible for broadcasting b_i is blocked in q_{2i-1} , so that n such
 431 processes are needed, plus one process in \ominus ; moreover, $n+2-i$ broadcasts of a_i and one
 432 broadcast of each b_i happen. ◀

433 4 Succinctness of reconfigurations compared to losses

434 In this section, we show that reconfigurable executions can be linearly more succinct than
 435 lossy executions, in terms of number of nodes. Given the tight linear bound on cutoff, this is
 436 somehow optimal.

437 ► **Theorem 12.** *There exists a family of broadcast protocols $(\mathcal{P}_n)_n$ with $\mathcal{P}_n = (Q_n, I_n, \Sigma_n, \Delta_n)$
 438 and target states $F_n \subseteq Q_n$ such that for every n :*

- 439 ■ *there exists a reconfigurable covering execution in \mathcal{P}_n with 3 nodes; and*
- 440 ■ *any lossy covering execution in \mathcal{P}_n requires $O(n)$ nodes.*

441 **Proof.** \mathcal{P}_n is depicted in Figure 6. It has $3n+2$ states and we let $F_n = \{\ominus\}$. A covering
 442 reconfigurable execution of size 3 is given in Figure 7. Colored nodes broadcast a message
 443 in the step leading to the next configuration. Along that execution, the top node always
 444 remains at q_0 and alternatively broadcasts a to the middle node and disconnects; the middle
 445 node follows the chain of q_i states and alternatively broadcasts b_i 's to the bottom node which
 446 gradually progresses along the chain of states r_i and reaches \ominus .

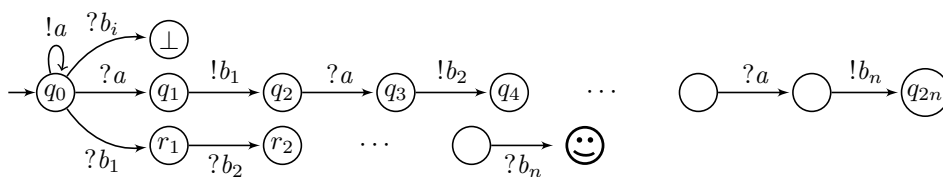


Figure 6 Example where reconfigurable semantics needs less nodes than lossy semantics.

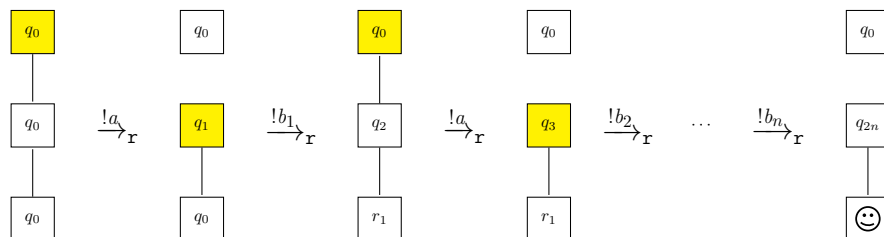


Figure 7 Covering reconfigurable execution with 3 nodes on the protocol from Figure 6.

Let us argue that in the lossy semantics, $O(n)$ nodes are needed to cover \odot . Obviously, one node, say n_{\odot} , is needed to reach the target state, after having received sequentially all the b_i 's (which should then correspond to real broadcasts). Towards a contradiction, assume there is a node n which makes n_{\odot} progress twice, that is, n is connected to n_{\odot} and performs at least two real broadcasts, say $!b_i$ and $!b_j$ with $i < j$. Node n needs to receive $j - i > 0$ times the message a after the real $!b_i$ has occurred, hence there must be at least one node in state q_0 connected to n after the real $!b_i$ by n . This is not possible, since this node has received the real $!b_i$ while being in q_0 , leading to \perp if $i > 1$, otherwise \perp or r_1 . Hence, each broadcast $!b_i$ needs to be sent by a different node. This requires at least $n+1$ nodes, say $\{n_i \mid 1 \leq i \leq n\} \cup \{n_{\odot}\}$: node n_i is responsible for broadcasting (with no loss) b_i and n_{\odot} progresses towards \odot . Notice that n_{\odot} might be the node responsible for broadcasting all the a 's. We conclude that $n+1$ is a lower bound on the number of nodes needed to cover \odot in the lossy semantics.

To complete this example, observe that $n+1$ nodes do actually suffice in lossy semantics to cover \odot . Let $\mathbf{N} = \{n_i \mid 1 \leq i \leq n\} \cup \{n_{\odot}\}$ and consider the static communication topology defined by $n_i \sim n_{\odot}$ for every i . In the covering lossy execution, node n_{\odot} initially broadcasts a 's, so that all its neighbours, the n_i 's can move to q_{2i-1} , using lost sendings. Then the each node n_i broadcasts its message b_i to n_{\odot} , starting with n_1 until n_n , so that n_{\odot} reaches \odot . ◀

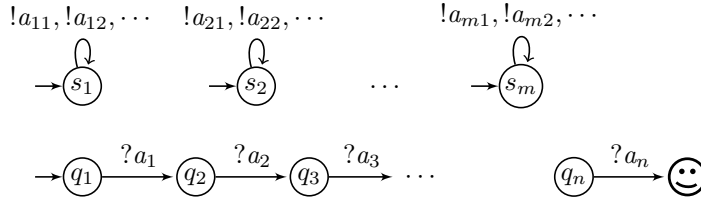
5 Complexity of deciding the size of minimal witnesses

We now consider the following decision problem of determining the minimal size of coverability witnesses for both the reconfigurable and lossy semantics.

MINIMUM NUMBER OF NODES FOR COVERABILITY (MINCOVER)

Input: A broadcast protocol \mathcal{P} , a set of states $F \subseteq Q$, and $k \in \mathbb{N}$.

Question: Does there exist a reconfigurable/lossy execution ρ covering some state in F , and with $\#\text{nodes}(\rho) = k$?



■ **Figure 8** Illustration of the reduction to prove NP-hardness of MINCOVER.

470 By the copycat properties (for both semantics), if there is a covering execution of size less
 471 than k , then there is one of size exactly k .

472 ► **Theorem 13.** *MINCOVER is NP-complete for both reconfigurable and lossy broadcast*
 473 *networks.*

474 The NP-hardness of MINCOVER is proved by reduction from SETCOVER, which is known
 475 to be NP-complete [9]. Recall that SETCOVER takes as input a finite set of elements \mathcal{U} , a
 476 collection \mathcal{S} of subsets of \mathcal{U} and an integer k , and returns yes iff there exists a subcollection
 477 of \mathcal{S} of size at most k that covers \mathcal{U} .

478 Given an instance of the SETCOVER problem $(\mathcal{U}, \mathcal{S}, k)$ with $\mathcal{U} = \{a_1, a_2, \dots, a_n\}$ and
 479 $\mathcal{S} = \{S_1, S_2, \dots, S_m\}$, we build a protocol $\mathcal{P} = (Q, I, \Sigma, \Delta)$ as depicted in Figure 8, where
 480 we assume $S_i = \{a_{i1}, a_{i2}, \dots\}$ for every i .

481 We can then show that \mathcal{U} has a cover using \mathcal{S} of size k if and only if there exists a
 482 reconfigurable/lossy execution for \mathcal{P} covering F and with $k+1$ nodes.

483 For the NP-membership, it suffices to observe that the length of a minimal covering
 484 execution is polynomially bounded, thanks to Theorem 5 and 8. Moreover, configurations
 485 and updates of configurations by given transitions can be represented in and computed in
 486 a compact way. It is thus possible to implement a guess-and-check NP-algorithm for the
 487 MinCover problem, that non deterministically guesses an execution with k nodes of maximal
 488 length that is polynomially bounded in the size of the broadcast protocol.

489 6 Conclusion

490 In this paper, we have given a tight linear bound on the cutoff and a tight quadratic bound
 491 on the covering length for reconfigurable broadcast networks. We have also proposed a new
 492 semantics for broadcast networks with a static topology, where messages can be lost at
 493 sending. Similar tight bounds can be proven for that new semantics. Proofs are based on a
 494 refinement of the saturation algorithm of [2], and on fine analysis of copycat lemmas. As a
 495 side result of these constructions, we get that the set of states which can be covered by the
 496 two semantics is actually the same, but that the reconfigurable semantics can be linearly
 497 more succinct (in terms of number of nodes). We also prove the NP-completeness for the
 498 existence of a witness execution with the minimal number of nodes.

499 As future work, we want to pursue the study of the model with stochastic losses, and
 500 design analysis algorithms for various quantitative questions. Also, in this work we have
 501 not studied the tradeoff between number of nodes and length of covering computation. The
 502 precise interplay between number of nodes and length of covering execution is a possible
 503 direction for future work.

504 — **References** —

- 505 **1** Roderick Bloem, Swen Jacobs, Ayrat Khalimov, Igor Konnov, Sasha Rubin, Helmut
506 Veith, and Josef Widder. *Decidability of Parameterized Verification*. Synthesis Lectures
507 on Distributed Computing Theory. Morgan & Claypool Publishers, 2015. doi:10.2200/
508 S00658ED1V01Y201508DCT013.
- 509 **2** Giorgio Delzanno, Arnaud Sangnier, Riccardo Traverso, and Gianluigi Zavattaro. On the
510 complexity of parameterized reachability in reconfigurable broadcast networks. In *Proceedings of*
511 *the 32nd Conference on Foundations of Software Technology and Theoretical Computer Science*
512 *(FSTTCS'12)*, volume 18 of *Leibniz International Proceedings in Informatics*, pages 289–300.
513 Leibniz-Zentrum für Informatik, December 2012. doi:10.4230/LIPIcs.FSTTCS.2012.289.
- 514 **3** Giorgio Delzanno, Arnaud Sangnier, and Gianluigi Zavattaro. Parameterized verification of
515 ad hoc networks. In *Proceedings of the 21st International Conference on Concurrency Theory*
516 *(CONCUR'10)*, volume 6269 of *Lecture Notes in Computer Science*, pages 313–327. Springer,
517 September 2010. doi:10.1007/978-3-642-15375-4_22.
- 518 **4** Giorgio Delzanno, Arnaud Sangnier, and Gianluigi Zavattaro. Verification of ad hoc networks
519 with node and communication failures. In *Proceedings of the 32nd International Conference on*
520 *Formal Techniques for Distributed Systems (FMODS/FORTE'12)*, volume 7273 of *Lecture*
521 *Notes in Computer Science*. Springer, 2012.
- 522 **5** E. Allen Emerson and A. Prasad Sistla. Symmetry and model checking. *Formal Methods in*
523 *System Design*, 9(1-2):105–131, August 1996. doi:10.1007/BF00625970.
- 524 **6** Javier Esparza. Keeping a crowd safe: On the complexity of parameterized verification
525 (invited talk). In *Proceedings of the 31st Symposium on Theoretical Aspects of Computer*
526 *Science (STACS'14)*, volume 25 of *Leibniz International Proceedings in Informatics*, pages
527 1–10. Leibniz-Zentrum für Informatik, March 2014. doi:10.4230/LIPIcs.STACS.2014.1.
- 528 **7** Javier Esparza, Pierre Ganty, and Rupak Majumdar. Parameterized verification of asyn-
529 chronous shared-memory systems. In *Proceedings of the 25th International Conference on*
530 *Computer Aided Verification (CAV'13)*, volume 8044 of *Lecture Notes in Computer Science*,
531 pages 124–140. Springer, July 2013. doi:10.1007/978-3-642-39799-8_8.
- 532 **8** Steven M. German and A. Prasad Sistla. Reasoning about systems with many processes.
533 *Journal of the ACM*, 39(3):675–735, July 1992. doi:10.1145/146637.146681.
- 534 **9** Richard M. Karp. Reducibility among combinatorial problems. In *Complexity of Computer*
535 *Computations*, The IBM Research Symposia Series, pages 85–103, 1972.
- 536 **10** Marvin Minsky. *Computation: Finite and Infinite Machines*. Prentice Hall International, 1967.
- 537 **11** Ichiro Suzuki. Proving properties of a ring of finite-state machines. *Information Processing*
538 *Letters*, 28(4):213–214, 1988. doi:10.1016/0020-0190(88)90211-6.