

# Amnesiac Flooding: Easy to Break, Hard to Escape

Henry Austin 


Durham University, United Kingdom

Maximillien Gadouleau 

Durham University, United Kingdom

George B. Mertzios 

Durham University, United Kingdom

Amitabh Trehan 

Durham University, United Kingdom

---

## Abstract

Broadcast is a central problem in distributed computing. Recently, Hussak and Trehan [PODC'19/STACS'20/DC'23] proposed a stateless broadcasting protocol (Amnesiac Flooding), which was surprisingly proven to terminate in asymptotically optimal time (linear in the diameter of the network). However, it remains unclear: (i) Are there other stateless terminating broadcast algorithms with the desirable properties of Amnesiac Flooding, (ii) How robust is Amnesiac Flooding with respect to *faults*?

In this paper we make progress on both of these fronts. Under a reasonable restriction (obliviousness to message content) additional to the fault-free synchronous model, we prove that Amnesiac Flooding is the *only* strictly stateless deterministic protocol that can achieve terminating broadcast. We achieve this by identifying four natural properties of a terminating broadcast protocol that Amnesiac Flooding uniquely satisfies. In contrast, we prove that even minor relaxations of *any* of these four criteria allow the construction of other terminating broadcast protocols.

On the other hand, we prove that Amnesiac Flooding can become non-terminating or non-broadcasting, even if we allow just one node to drop a single message on a single edge in a single round. As a tool for proving this, we focus on the set of all *configurations* of transmissions between nodes in the network, and obtain a *dichotomy* characterizing the configurations, starting from which, Amnesiac Flooding terminates. Additionally, we characterise the structure of sets of Byzantine agents capable of forcing non-termination or non-broadcast of the protocol on arbitrary networks.

**2012 ACM Subject Classification** Mathematics of computing → Discrete mathematics; Mathematics of computing → Graph algorithms; Theory of computation → Distributed algorithms; Theory of computation → Graph algorithms analysis

**Keywords and phrases** Amnesiac flooding, Terminating protocol, Algorithm state, Stateless protocol, Flooding algorithm, Network algorithms, Graph theory, Termination, Communication, Broadcast.

**Digital Object Identifier** 10.4230/LIPIcs.DISC.2025.17

**Related Version** Full Version

*Full Version:* <https://arxiv.org/abs/2502.06001> [2]

**Funding** George B. Mertzios: Supported by the EPSRC grant EP/P020372/1.

Amitabh Trehan: Supported by the EPSRC grant EP/P021247/1.

**Acknowledgements** We would like to thank Danial Maqbool (Durham University) for his insightful work with a precursor to the uniqueness result.



© Henry Austin, Maximillien Gadouleau, George B. Mertzios and Amitabh Trehan;  
licensed under Creative Commons License CC-BY 4.0

39th International Symposium on Distributed Computing (DISC 2025).

Editor: Dariusz R. Kowalski; Article No. 17; pp. 17:1–17:23

Leibniz International Proceedings in Informatics



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

## 1 Introduction

The dissemination of information to disparate participants is a fundamental problem in both the construction and theory of distributed systems. A common strategy for solving this problem is to “broadcast”, i.e. to transmit a piece of information initially held by one agent to all other agents in the system [1, 18, 21, 24, 25]. In fact, broadcast is not merely a fundamental communication primitive in many models, but also underlies solutions to other fundamental problems such as leader election and wake-up. Given this essential role in the operation of distributed computer systems and the potential volume of broadcasts, an important consideration is simplifying the algorithms and minimizing the overhead required for each broadcast.

Within a synchronous setting, *Amnesiac Flooding* as introduced by Hussak and Trehan in 2019 [13, 14] eliminates the need of the standard flooding algorithm to store historical messages. The algorithm terminates in asymptotically optimal  $O(D)$  time (for  $D$  the diameter of the network) and is stateless as agents are not required to hold any information between communication rounds. The algorithm in the fault-free synchronous message passing model is defined as follows:

► **Definition 1. Amnesiac flooding algorithm.** (adapted from [16]) Let  $G = (V, E)$  be an undirected graph, with vertices  $V$  and edges  $E$  (representing a network where the vertices represent the nodes of the network and edges represent the connections between the nodes). Computation proceeds in synchronous ‘rounds’ where each round consists of nodes receiving messages sent from their neighbours. A receiving node then sends messages to some neighbours which arrive in the next round. No messages are lost in transit. The algorithm is defined by the following rules:

- (i) All nodes from a subset of sources or initial nodes  $I \subseteq V$  send a message  $M$  to all of their neighbours in round 1.
- (ii) In subsequent rounds, every node that received  $M$  from a neighbour in the previous round, sends  $M$  to all, and only, those nodes from which it did not receive  $M$ . Flooding terminates when  $M$  is no longer sent to any node in the network.

These rules imply several other desirable properties. Firstly, the algorithm only requires the ability to forward the messages, but does not read the content (or even the header information) of any message to make routing decisions. Secondly, the algorithm only makes use of local information and does not require knowledge of a unique identifier. Thirdly, once the broadcast has begun, the initial broadcaster may immediately forget that they started it.

However, extending Amnesiac Flooding and other stateless flooding algorithms (such as those proposed in [27, 29, 4]) beyond synchronous fault-free scenarios is challenging. This is due to the fragility of these algorithms and their inability to build in complex fault-tolerance due to the absence of state and longer term memory. It has subsequently been shown that no stateless flooding protocol can terminate under moderate asynchrony, unless it is allowed to perpetually modify a super-constant number (i.e.  $\omega(1)$ ) of bits in each message [27]. Yet, given the fundamental role of broadcast in distributed computing, the resilience of these protocols is extremely important even on synchronous networks.

Outside of a partial robustness to crash failures, the fault sensitivity of Amnesiac Flooding under synchrony has not been explored in the literature. This omission is further compounded by the use of Amnesiac Flooding as an underlying subroutine for the construction of other broadcast protocols. In particular, multiple attempts have been made to extend Amnesiac Flooding to new settings (for example routing multiple concurrent broadcasts [4] or flooding networks without guaranteed edge availability [29]), while maintaining its desirable properties.

However, none have been entirely successful, typically requiring some state-fulness. It has not in fact been established that *any other* protocol can retain all of Amnesiac Flooding's remarkable properties even in its original setting. These gaps stem fundamentally from the currently limited knowledge of the dynamics of Amnesiac Flooding beyond the fact of its termination and its speed to do so. In particular, all of the existing techniques (e.g. parity arguments such as in [16] and auxiliary graph constructions such as in [28]) used to obtain termination results for Amnesiac Flooding are unable to consider faulty executions of the protocol and fail to capture the underlying structures driving terminating behaviour.

We address these gaps through the application of novel analysis and by considering the structural properties of Amnesiac Flooding directly. By considering the sequence of message configurations, we are able to identify the structures underlying Amnesiac Flooding's termination and use these to reason about the algorithm in arbitrary configurations. The resulting dichotomy gives a comprehensive and structured understanding of termination in Amnesiac Flooding. For example, we apply this to investigate the sensitivity of Amnesiac Flooding with respect to several forms of fault and find it to be quite fragile. Furthermore, we show that under reasonable assumptions on the properties of a synchronous network, any strictly stateless deterministic terminating broadcast algorithm oblivious to the content of messages, must produce the exact same sequence of message configurations as Amnesiac Flooding on any network from any initiator. We therefore argue that Amnesiac Flooding is unique. However, we show that if any of these restrictions are relaxed, even slightly, distinct terminating broadcast algorithms can be obtained. As a result of this uniqueness and simplicity, we argue that Amnesiac Flooding represents a prototypical broadcast algorithm. This leaves open the natural question: do there exist fundamental stateless algorithms underlying solutions to other canonical distributed network problems? Though memory can be essential or naturally useful in certain scenarios [5, 7, 8, 10, 17, 20], understanding what we can do with statelessness can help us push fundamental boundaries.

## 1.1 Our Contributions

In this work, we investigate the existence of other protocols possessing the following four desirable properties of Amnesiac Flooding:

1. *Strict Statelessness*: Nodes maintain no information other than their port labellings between rounds. This includes whether or not they were in the initiator set.
2. *Obliviousness*: Routing decisions may not depend on the contents of received messages.
3. *Determinism*: All decisions made by a node must be deterministic.
4. *Unit Bandwidth*: Each node may send at most one message per edge per round.

Our main technical results regarding the existence of alternative protocols to Amnesiac Flooding are given in the next two theorems (reworded in Section 4).

► **Theorem 2** (Uniqueness of Amnesiac Flooding). *Any terminating broadcast algorithm possessing all of Strict Statelessness, Obliviousness, Determinism and Unit Bandwidth behaves identically to Amnesiac Flooding on all graphs under all valid labellings for all source nodes.*

Note that the last theorem allows, but does not require, that nodes have access to unique identifiers labelling themselves and their ports. However, we enforce the condition that these identifiers, should they exist, may be drawn adversarially from some super set of  $[n + \kappa]$  where  $n$  is the number of nodes on the networks and  $\kappa = R(9, 8)$  where  $R(9, 8)$  is a Ramsey

## 17:4 Amnesiac Flooding: Easy to Break, Hard to Escape

133 Number ( $\leq 2662$ )[22]. It is important to stress here that Theorem 2 holds even if the space  
134 of unique identifiers is only greater than  $n$  by an additive constant. In contrast to the last  
135 theorem, the next one does not assume that agents have access to unique identifiers.

136 ► **Theorem 3** (Existence of relaxed Algorithms). *There exist terminating broadcast algorithms*  
137 *which behave distinctly from Amnesiac Flooding on infinitely many networks possessing any*  
138 *three of: Strict Statelessness, Obliviousness, Determinism and Unit Bandwidth.*

139 We derive four of these relaxed algorithms which all build upon Amnesiac Flooding:

- 140 1. NEIGHBOURHOOD-2 FLOODING: *Strict Statelessness* is relaxed and agents are given  
141 knowledge of their neighbours' neighbours. This allows for distinct behaviour on networks  
142 of radius one as some agents are aware of the entire network topology.
- 143 2. RANDOM FLOODING: *Determinism* is relaxed and agents are given access to a random  
144 coin. Agents randomly choose in each round whether to use Amnesiac Flooding or to  
145 forward messages to all of their neighbours.
- 146 3. 1-BIT FLOODING (MESSAGE DEPENDENT): *Obliviousness* is relaxed and the source is  
147 allowed to include one bit of read-only control information in the messages. The source  
148 records in the message whether they are a leaf vertex, and if so agents perform Amnesiac  
149 Flooding upon receiving the message. Otherwise, agents perform a modified version  
150 where leaf vertices return messages.
- 151 4. 1-BIT FLOODING (HIGH BANDWIDTH): *Unit Bandwidth* is relaxed and agents are allowed  
152 to send either one or two messages over an edge. This permits the same algorithm as the  
153 previous case by encoding the control information in the number of messages sent.

154 We note that despite being a non-deterministic algorithm RANDOM FLOODING achieves  
155 broadcast with certainty and terminates almost surely in finite time.

156  
157 We also perform a comprehensive investigation of the fault sensitivity of Amnesiac  
158 Flooding in a synchronous setting. Through the use of a method of invariants, we obtain  
159 stronger characterizations of termination than were previously known, for both Amnesiac  
160 Flooding, and a subsequently proposed Stateless Flooding protocol [27]. This allows us to  
161 provide precise characterizations of the behaviour of Amnesiac Flooding under the loss of  
162 single messages, uni-directional link failure, and time bounded Byzantine failures. The above  
163 invariants may be of independent interest, beyond fault sensitivity, as they provide strong  
164 intuition for how asynchrony interferes with the termination of both Amnesiac Flooding  
165 and the Stateless Flooding proposed in [27]. The main technical result concerning fault  
166 sensitivity is a *dichotomy* characterizing the configurations, starting from which, Amnesiac  
167 Flooding terminates. As the rigorous statement of the result requires some additional  
168 notation and terminology, we will state it only informally here. We show in Theorem 15  
169 that, whether Amnesiac Flooding terminates when begun from a configuration or not, is  
170 determined exclusively by the parity of messages distributed around cycles and so-called  
171 faux-even cycles (FEC) (essentially pairs of disjoint odd cycles connected by a path, see  
172 section 5 for a full definition) within the graph. It follows from this characterisation that  
173 Amnesiac Flooding terminates from a configuration if and only if the configuration can  
174 be reached from some sequence of multi-casts. Theorem 15 also implies the following three  
175 theorems which demonstrate the fragility of Amnesiac Flooding under three increasingly  
176 strong forms of fault. We give our fault model explicitly in Section 5.2.

177 ► **Theorem 4** (Single Message Failure). *If single-source Amnesiac Flooding experiences a*  
178 *single message drop failure for the message  $(u, v)$  then it fails to terminate if and only if*  
179 *either or both of the following hold:*

180 ■ *uv is not a bridge*

181 ■ *uv lies on a path between vertex disjoint odd cycles*

182 *Moreover, it fails to broadcast if and only if this is the first message sent along uv, uv is a*  
 183 *bridge, and the side of the cut containing u does not contain an odd cycle.*

184 We also consider the possibility of a link/edge failing in one direction (or if we consider  
 185 an undirected edge as two directed links in opposite directions, only one of the directed links  
 186 fails).

187 ► **Theorem 5 (Uni-directional link failure).** *For any graph  $G = (V, E)$  and any initiator set*  
 188  *$I \subsetneq V$  there exists an edge  $e \in E$  such that a uni-directional link failure at  $e$  will cause*  
 189 *Amnesiac Flooding to either fail to broadcast or fail to terminate when initiated from  $I$  on*  
 190  *$G$ . Furthermore, for any non-empty set of uni-directional link failures there exists  $v \in V$*   
 191 *such that, when Amnesiac Flooding is initiated at  $v$ , it will either fail to broadcast or fail to*  
 192 *terminate.*

193 We, finally, consider a set of Byzantine nodes, who know the original message, are free to  
 194 collude among themselves and may decide to forward this message in any arbitrary pattern  
 195 to their neighbours. However, for a discussion of termination to be meaningful, we require  
 196 that the nodes have Byzantine behaviour for only a finite number of rounds.

197 ► **Theorem 6 (Byzantine Failure).** *If Amnesiac Flooding on  $G = (V, E)$  initiated from  $I \subsetneq V$*   
 198 *experiences a weak Byzantine failure at  $J \subseteq V \setminus I$ , then the adversary can force:*

199 ■ *Failure to broadcast if and only if  $J$  contains a cut vertex set.*

200 ■ *Non-termination if and only if at least one member of  $J$  lies on either a cycle or a path*  
 201 *between odd-cycles.*

202 Two natural corollaries of the Theorem 4 we wish to highlight here are: (i) on any network  
 203 from any initial node there exists a single message, the dropping of which, will produce either  
 204 non-termination or non-broadcast and (ii) dropping any message on a bipartite network will  
 205 cause either non-termination or non-broadcast. The latter requires the additional observation  
 206 that each edge is traversed by a message precisely once in a bipartite graph under Amnesiac  
 207 Flooding. Similarly, Theorem 6 implies that any Byzantine set containing a non-leaf node  
 208 on any network can force either non-termination or non-broadcast for Amnesiac Flooding  
 209 with any initial set.

210

## 211 1.2 Organisation of the paper

212 The initial part of the paper presents the required technical and motivational background,  
 213 statements of our results and a technical outline of the more interesting proofs. Due to  
 214 space limitations detailed proofs are deferred to the appendices and the full version [2].  
 215 Related work is presented in Section 2. Section 3 presents the model and notation required  
 216 for the following technical sections. *Uniqueness* of the algorithm is discussed in Section 4.  
 217 In subsection 4.1 we discuss the conditions under which the algorithm is unique and in  
 218 subsection 4.2 we relax the conditions individually to derive additional algorithms. The proof  
 219 of the uniqueness result is given in Appendix A. We present our work on the *termination*  
 220 *dichotomy* for Amnesiac Flooding and its applications to *fault sensitivity* in Section 5 with  
 221 the proof of the dichotomy given in Appendix B. We end with our conclusions and pointers  
 222 to future work in Section 6.

## 223 2 Related Work

224 The literature surrounding both the broadcast problem and fault sensitivity is vast, and a  
 225 summary of even their intersection is well beyond the scope of this work. Instead, we shall  
 226 focus on work specifically concerning stateless, or nearly stateless broadcast.

227 The termination of Amnesiac Flooding and its derivatives has been the focus of several  
 228 works. In combination they provide the following result that Amnesiac Flooding terminates  
 229 under any sequence of multi-casts, i.e.

230 ► **Theorem 7** (Termination of Amnesiac Flooding (adapted from [15, 16, 28])). *For  $G = (V, E)$   
 231 a graph and  $I_1, \dots, I_k \subseteq V$  a sequence of sets of initiator nodes, Amnesiac Flooding on  $G$   
 232 terminates when initiated from  $I_1$  in round 1, then  $I_2$  in round 2 and so forth.*

233 Two independent proofs of the algorithm's termination have been presented, using either  
 234 parity arguments over message return times [16] or axillary graph constructions [28]. The  
 235 latter technique has further been used to establish tight diameter independent bounds on  
 236 the termination of multicast using Amnesiac Flooding, complementing the eccentricity based  
 237 bounds of [16]. The techniques we develop in this work, however, are more closely aligned  
 238 with those of [15], as we exploit a similar notion to their "even flooding cycles" in our message  
 239 path argument. In contrast to that work we focus on arbitrary configurations of messages,  
 240 rather than just those resulting purely from a correct broadcast which allows us to obtain a  
 241 stronger characterisation. Combining our techniques with the dual "reverse" flood introduced  
 242 by [15], we are able to show the complement of Theorem 7, that only those configurations  
 243 reachable from a sequence of multi-casts lead to termination.

244 There have, further, been multiple variants of Amnesiac Flooding introduced. It was  
 245 observed by [27] in a result reminiscent of the BASIC protocol proposed by [12], that  
 246 sending a second wave of messages from a subset of the initial nodes reduces the worst  
 247 case  $2D + 1$  termination time to the optimal  $D + 1$  in all but a specific subset of bipartite  
 248 graphs. We note that our fault sensitivity results extend naturally to this algorithm as  
 249 well, as the same invariants apply to this setting. Beyond this, there have been several  
 250 approaches to deal with the flooding of multiple messages simultaneously. In [15], the original  
 251 authors of [13] show that under certain conditions termination can be retained, even when  
 252 conflicting floods occur. Since then, two partially stateless algorithms have been proposed,  
 253 both making use of message buffers and a small amount of local memory [29, 4]. We will  
 254 not be directly concerned with these approaches, however, as we assume a single concurrent  
 255 broadcast throughout. However, the mechanism employed in [29] should be highlighted as it  
 256 rather cleverly exploits the underlying parity properties we identify as driving termination.  
 257 Furthermore, as reduction to Amnesiac Flooding is used as a technique for proofs in many of  
 258 these works, the comprehensive understanding of its termination we present here could prove  
 259 a powerful tool for future work in these areas.

260 While the robustness of Amnesiac flooding and its variants have been previously studied,  
 261 this has been focused on two forms of fault. The first is the disappearance and reappearance  
 262 of nodes and links. The termination of Amnesiac Flooding is robust to disappearance and  
 263 vulnerable to reappearance [16]. We will observe that this is a necessary consequence of the  
 264 invariants driving termination and their relation to cyclicity. In particular, the disappearance  
 265 of nodes and links cannot form new cycles violating the invariant, whereas their reappearance  
 266 can. A pseudo-stateless extension to Amnesiac Flooding has been proposed to circumvent  
 267 this [29], implicitly exploiting the parity conditions of [16]. The second are faults that  
 268 violate synchrony. Under a strong form of asynchrony, truly stateless and terminating  
 269 broadcast is impossible [27]. However, the landscape under a weaker form of asynchrony



(namely, the case of fixed delays on communication links) is more fine-grained. Although termination results have been obtained for cycles, as well as the case of single delayed edges in bipartite graphs [16], there is no clear understanding of the impact of fixed channel delays. While we do not directly address this, we believe that techniques mirroring our invariant characterizations may prove fruitful in this area. To our knowledge, this work is the first to consider both the uniqueness of Amnesiac Flooding, as well as its fault sensitivity beyond node/link unavailability in a synchronous setting.

Beyond Amnesiac Flooding and its extensions, the role of memory in information dissemination is well studied in a variety of contexts. Frequently, stateful methods obtain faster termination time, such as in the phone-call model where the ability to remember one's communication partners and prevent re-communication dramatically improves termination time and message efficiency [10, 5, 8]. Similarly, for bit dissemination in the passive communication model the addition of only  $\log \log n$  bits of memory is sufficient to break the near linear time convergence lower bound of [7] and achieve polylogarithmic time [17]. Even more strongly, a recent work [20] has shown that in the context of synchronous anonymous dynamic networks, stabilizing broadcast from an idle start is impossible with  $O(1)$  memory and even with  $o(\log n)$  memory if termination detection is required. Despite this, low memory and even stateless broadcasts remain desirable [12]. The possibility of solving other canonical distributed computing problems beyond broadcast, in a stateless manner, remains intriguing. In this direction, various low memory stateful models have been proposed to handle more complex distributed problems e.g. the compact local streaming (CLS) model in [6] with deterministic solutions (routing, self-healing fault-tolerance etc.) and randomised solutions for distributed colouring in a similar model [11].

Stateless broadcast schemes have been studied in a variety of contexts. To give an example, they are used in mobile ad hoc networks, which, due to the lower power and rapid movement of devices, see diminishing returns from maintaining information about the network [19]. However, given a lack of synchronisation as well as the wish to avoid so-called broadcast storms [26], these techniques typically rely on either some form of global knowledge (such as the direction or distance to the initiator) or the ability to sample network properties by eavesdropping on communications over time [3, 23]. It should be noted however, that in contrast to many models, such as anonymous dynamic networks, radio networks and many mobile ad hoc networks, the typical framework for studying stateless flooding ("true statelessness" as defined by [27] restricting the model of [9]) permits the knowledge and distinguishing of neighbours in both broadcasting and receiving.

### 3 Model and Notation

Throughout this work we consider only finite, connected graphs on at least two nodes. We denote the set  $\{1, \dots, x\}$  by  $[x]$  and  $R(r, s)$  the Ramsey number such that any graph on  $R(r, s)$  vertices contains either a clique on  $r$  vertices or an independent set on  $s$  vertices. In this work, we make use of a generic synchronous message passing model (as described in definition 1) with several additional assumptions based on the truly stateless model of [27]. Our agents are the nodes of a network and are able to communicate via messages of arbitrary size sent over the edges of this network. Computation occurs in synchronous rounds, consisting of three phases: (i) nodes receive messages sent in the previous round, (ii) nodes perform computation and (iii) nodes send messages to be delivered in the next round. Unless stated otherwise, agents do not suffer faults and no messages are lost. In addition to these standard assumptions, we enforce that the model is stateless, i.e. nodes

## 17:8 Amnesiac Flooding: Easy to Break, Hard to Escape

cannot maintain any additional information between rounds (such as routing information, previous participation in the flood or even a clock value), cannot hold onto messages and can only forward, not modify the messages.

Unless stated otherwise we do not assume that nodes have access to unique identifiers, however they have locally labelled ports that are distinguishable and totally orderable for both receiving and sending messages. When we do work with identifiers these identifiers are assumed to be unique, drawn from  $[|V| + \kappa]$  for  $\kappa > 0$  a constant and assigned adversarially. We refer to such an assignment of IDs to a network as a labelling of the network and we identify the port label of a link leading to a node with the ID of that node. We further assume that individual nodes have access to arbitrarily powerful computation on information they do have.

We are principally interested in the problem of *broadcast*, although we will occasionally consider the related *multicast* problem i.e. there are multiple initiators who may potentially wake up in different rounds with the same message to be broadcast. For a graph  $G = (V, E)$  and an *initiator* set  $I \subseteq V$  we say that a node is informed if it has ever received a message from a previously informed node (where initiators are assumed to begin informed). An algorithm *correctly* solves broadcast (resp. multicast) on  $G$  if for all singleton (resp. non-empty) initiator sets there exists a finite number of rounds after which all nodes will be informed. Unless specified otherwise, we assume that initiator nodes remain aware of their membership for only a single round. We say that an algorithm *terminates* on  $G = (V, E)$  if, for all valid initiator sets, there exists a finite round after which no further messages are sent (i.e. the communication network quiesces).

Formally, for the message  $M$ , we denote a *configuration* of Amnesiac Flooding as follows:

► **Definition 8.** A *configuration* of Amnesiac Flooding on graph  $G = (V, E)$  is a collection of messages/edges  $S \subseteq \{(u, v) | uv \in E\}$  where  $(u, v) \in S$  implies that in the current round  $u$  sent a message to  $v$ .

Further, for  $H$  a subgraph of  $G$ , we denote by  $S_H$ ,  $S$  restricted to  $H$ . Below, we define the operator  $A_{I,G} : 2^{V^2} \rightarrow 2^{V^2}$  to implement one round of Amnesiac Flooding on the given configuration where agents in  $I$  receive the message from outside the network:

► **Definition 9.** The operator  $A_{I,G} : 2^{V^2} \rightarrow 2^{V^2}$  is defined as follows: The set  $I$  of nodes initiate the broadcast. The message  $(u, v) \in A_{I,G}(S)$  if  $uv \in E$ ,  $(v, u) \notin S$  and either  $\exists w \in V : (w, u) \in S$  or  $u \in I$ .

We will drop the subscript when  $G$  is obvious from context and  $I = \emptyset$ . Further, we adopt the standard convention of using  $A_{I,G}^k$  to mean  $A_{I,G}$  applied  $k$  times. Theorem 7 in this notation can be expressed as follows:

► **Theorem 10** (Theorem 7 restated). For any graph  $G = (V, E)$ , and any finite sequence  $I_1, \dots, I_k \subseteq V$ , there exists  $m \in \mathbb{N}$  such that

$$A_{\emptyset,G}^m(A_{I_k,G}(\dots A_{I_1,G}(\emptyset))) = \emptyset.$$

### 4 Uniqueness

In this section, we investigate broadcast protocols similar to Amnesiac Flooding and establish four desirable properties that Amnesiac Flooding uniquely satisfies in combination. On the other hand, we show that this result is sharp and that by relaxing any of these conditions one can obtain similar terminating broadcast protocols.



## 4.1 Uniqueness

Our first major result concerns the uniqueness of Amnesiac Flooding. Given the algorithms surprising properties, a natural question is whether other broadcast algorithms exist maintaining these properties. Specifically, does there exist a terminating protocol for broadcast which obeys all of the following for all graphs and valid port labellings:

1. *Strict Statelessness*: Nodes maintain no information other than their port labellings between rounds. This includes whether or not they were in the initiator set.
2. *Obliviousness*: Routing decisions may not depend on the contents of received messages.
3. *Determinism*: All decisions made by a node must be deterministic.
4. *Unit Bandwidth*: Each node may send at most one message per edge per round.

The answer is negative. We will actually prove the slightly stronger case, that this holds even if agents are provided with unique identifiers, are aware of the identifiers of their neighbours and that these identifiers have bounded size. Intuitively, the *Strict Statelessness* condition forces any broadcast protocol to make its forwarding decisions based only on the messages it receives in a given round. The combination of *Obliviousness* and *Unit Bandwidth* forces any protocol meeting the conditions to view messages as atomic. Finally, *Determinism* forces the protocol to make identical decisions every time it receives the same set of messages. Formally, any broadcast protocol meeting the four conditions must be expressible in the following form:

► **Definition 11.** A protocol  $P = (b, f)$  is a pair of functions, an initial function  $b$  and a forwarding function  $f$ , where  $b : \mathbb{N} \times 2^{\mathbb{N}} \rightarrow 2^{\mathbb{N}}$  and  $f : \mathbb{N} \times 2^{\mathbb{N}} \times 2^{\mathbb{N}} \rightarrow 2^{\mathbb{N}}$ . The protocol is implemented as follows. On the first round the initiator node  $s$  with neighbourhood  $N(s)$  sends messages to every node with a label in  $b(s, N(s))$ . On future rounds, each node  $u$  sends messages to every node with a label in  $f(u, N(u), S)$  where  $S$  is the set of labels of nodes  $u$  received messages from in the previous round. Further, we require that for any  $B \subseteq A \subset \mathbb{N}$   $b(u, A), f(u, A, B) \subseteq A$  and that  $f(u, A, \emptyset) = \emptyset$ , enforcing that agents can only communicate over edges of the graph and can only forward messages they have actually received respectively.

In this setting, achieving broadcast is equivalent to every node receiving a message at least once and terminating in finite time corresponds to there existing a finite round after which no messages are sent. For example, Amnesiac Flooding is defined by the following functions:

► **Definition 12 (Amnesiac Flooding Redefinition).** Amnesiac Flooding is defined by  $P_{AF} = (b_{AF}, f_{AF})$  where, for all  $T \subseteq S \subset \mathbb{N}$ ,  $b(u, S) = S$  and  $f(u, S, T) = S \setminus T$  if  $T \neq \emptyset$  and  $\emptyset$  otherwise.

In order, to argue that Amnesiac Flooding is unique we require a notion of what it means for two broadcast algorithms to be distinct.

► **Definition 13.** Let  $G = (V, E)$  be a graph and  $L \subseteq \mathbb{N}$  a set of labels. We say that the pair  $(G, L)$  distinguishes the protocols  $P$  and  $Q$  if there exists a labelling of  $G$  using only labels from  $L$  such that for some initial vertex  $s$ ,  $P$  and  $Q$  send messages over different sets of edges in the same round when implementing broadcast on  $G$  initiated from  $s$ . If there exists some pair  $(G, L)$  which distinguishes  $P$  and  $Q$ , we describe  $P$  and  $Q$  as distinct. Otherwise, we consider them the same protocol.

For a protocol  $P = (b, f)$ , a set  $S \subseteq \mathbb{N}$  and a number  $k \in \mathbb{N}$  we describe  $P$  as AF up to degree  $k$  on  $S$  if there is no graph  $G$  of maximum degree  $k$  such that  $(G, S)$  distinguishes  $P$  from  $P_{AF}$ . From here on we will assume that all unique labels are drawn from  $[n + \kappa]$

where  $\kappa$  is a sufficiently large constant and  $n$  is the number of nodes in the graph. As this only eliminates possible pairs distinguishing protocols from Amnesiac Flooding, this only strengthens the result. We obtain the following result, discussion and sketching the proof of which makes up the remainder of this section, provided  $\kappa \geq R(9, 8)$ .

► **Theorem 14.** (*Restatement of Theorem 2*) *Let  $P = (b, f)$  be a correct and terminating broadcast protocol defined according to definition 11, then  $P$  is not distinct from Amnesiac Flooding.*

**Proof sketch for Theorem 14.** The basic argument is to show that any correct and terminating broadcast protocol meeting the criteria is identical to Amnesiac Flooding. Our core technique is to construct a set of network topologies such that any policy distinct from Amnesiac Flooding fails on at least one of them. However, the behaviour of these alternative protocols can be quite complex, as can the relationship between the constraints enforced by different networks under different labellings. In order to circumvent this, we find very simple networks, and labellings there of, where any algorithm must behave like Amnesiac Flooding and then modify them to obtain new instances, with the property that only a small number of vertices may ever behave distinctly from Amnesiac Flooding. In these more manageable cases, we are able to then show that any distinct behaviour leads to an incorrect algorithm. More precisely, for any given protocol we derive a directed graph (separate from the network topology) describing its behaviour and demonstrate via a forbidden subgraph argument that any set of IDs of size  $R(x, 8)$  must contain a subset  $T$  of size at least  $x$  such that  $P$  is AF up to degree 1 on  $T$ . We take  $\kappa = R(9, 8)$  and show that there must then exist  $U \subseteq T$  containing at least 6 identifiers such that  $P$  is AF up to degree 2 on  $U$ . By constructing a set of small sub-cubic graphs, we are able to extend this to degree 3. These form the base case of a pair of inductive arguments. First, we construct a progression of sub-cubic graphs which enforce that if  $P$  is AF on  $[m]$  up to degree 3 it must be AF on  $[m + 1]$  up to degree 3. We then construct a family of graphs which have a single node of high-degree, while all other nodes have a maximum degree of 3 and so must behave as though running Amnesiac Flooding. These graphs permit a second inductive argument showing that this unique high degree node must also behave as if running Amnesiac Flooding. In combination, these two constructions enforce that  $P$  behaves like Amnesiac Flooding in all possible cases. The proof is given in appendix A. ◀

## 4.2 Relaxing the constraints

Despite the uniqueness established in the previous subsection, we are able to derive four relaxed algorithms distinct from Amnesiac Flooding each obeying only three of the four conditions.

► **Theorem 3** (Existence of relaxed Algorithms). *There exist terminating broadcast algorithms which behave distinctly from Amnesiac Flooding on infinitely many networks possessing any three of: Strict Statelessness, Obliviousness, Determinism and Unit Bandwidth.*

The algorithms we obtain all build upon Amnesiac Flooding, and we believe illuminate the role of each of the four conditions in the uniqueness result by showing what they prevent. The algorithms for *Obliviousness* and *Unit Bandwidth* are presented together, as they differ only in how control information is encoded.

■ **Strict Statelessness:** Several relaxations of this already exist, such as Stateless Flooding (the initiator retains information for one round) or even classical non-Amnesiac Flooding (nodes are able to retain 1-bit for one round). We present NEIGHBOURHOOD-2 FLOODING.

Nodes know the ID of their neighbours' neighbours. The protocol behaves distinctly on star graphs, as the hub can determine the entire network topology.

■ *Obliviousness and Unit Bandwidth:* 1-BIT FLOODING. Nodes are allowed to send a single bit of read-only control information (in the message header or encoded in the number of messages sent) communicating whether the initiator is a leaf vertex. If it is, nodes implement Amnesiac Flooding, otherwise they use a different mechanism called PARROT FLOODING (leaves bounce the message back) which always terminates when begun from a non-leaf vertex.

■ *Determinism:* RANDOM-FLOODING. Nodes have access to one bit of randomness per round. Each round every node randomly chooses to implement Amnesiac Flooding or to forward to all neighbours. RANDOM-FLOODING is correct with certainty and terminates almost surely in finite time.

Since each of these constitutes only a minor relaxation of the restrictions, we argue that the uniqueness of Amnesiac Flooding is in some sense sharp. We present the protocols fully and demonstrate their correctness and termination for each of these cases independently in the full version [2].

## 5 Termination Dichotomy

With the uniqueness of Amnesiac Flooding established, a greater understanding of its properties is warranted. In this section, we study the configuration space of Amnesiac Flooding and obtain an exact characterisation of terminating configurations. We then apply this to investigate the algorithm's fault sensitivity.

### 5.1 Obtaining a termination dichotomy

In order to consider the fault sensitivity of Amnesiac Flooding, we need to be able to determine its behaviour outside of correct broadcasts. Unfortunately, neither of the existing termination proofs naturally extend to the case of arbitrary message configurations. Fortunately, we can derive an invariant property of message configurations when restricted to subgraphs that exactly captures non-termination, which we will call "balance" (see Definition 20).

► **Theorem 15.** *Let  $S$  be a configuration on  $G = (V, E)$  then there exists  $k \geq 0$  such that  $A_G^k(S) = \emptyset$  if and only if  $S$  is balanced on  $G$ .*

In fact we obtain that not only do balanced configurations terminate, they terminate quickly.

► **Corollary 16.** *Let  $S$  be a balanced configuration on  $G = (V, E)$  then there exists  $k \leq 2|E|$  such that  $A_G^k(S) = \emptyset$ .*

Intuitively, for the protocol not to terminate, we require that a message is passed around forever and since it is impossible for a message to be passed back from a leaf node the message must traverse either a cycle or system of interconnected cycles. As we will demonstrate in the rest of the section we need only consider systems of at most two cycles. Specifically, we introduce an invariant property determined by parity constraints on the number of messages travelling in each direction and their spacing around: odd-cycles, even cycles and what we will dub, faux-even cycles.

► **Definition 17.** *A faux-even cycle (FEC) is a graph comprised of either two node disjoint odd cycles connected by a path or two odd cycles sharing only a single node. We denote by*

## 17:12 Amnesiac Flooding: Easy to Break, Hard to Escape

487  $FEC_{x,y,z}$  the FEC with one cycle of length  $2x+1$ , one of length  $2z+1$  and a path containing  
 488  $y$  edges between them. We emphasize that if  $y = 0$  the two cycles share a common node and  
 489 if  $y = 1$  the two cycles are connected by a single edge.

490 FECs get their name from behaving like even cycles with respect to the operator  $A$ . In order  
 491 to capture this we can perform a transformation to convert them into an equivalent even  
 492 cycle.

493 ► **Definition 18.** Let  $F = (V, E)$  be  $FEC_{x,y,z}$ . Then the even cycle representation of  $F$   
 494 denoted  $F_2$  is the graph constructed by splitting the end points of the interconnecting path in  
 495 two, and duplicating the path to produce an even cycle. Formally, if the two cycles are of the  
 496 form  $a_0 \dots a_{2x} a_0$  and  $c_0 \dots c_{2z} c_0$ , with  $a_0$  and  $c_0$  connected by the path  $b_1 \dots b_{y-1}$ , we construct  
 497 the following large even cycle from four paths:  $a_0 \dots a_{2x} a_{-1} d_1 \dots d_{y-1} c_{-1} c_{2z} \dots c_0 b_{y-1} \dots b_1 a_0$ .  
 498 Here  $a_{-1}$  is a copy of  $a_0$ ,  $c_{-1}$  is a copy of  $c_0$ , and the path  $d_1 \dots d_{y-1}$  is a copy of the path  
 499  $b_1 \dots b_{y-1}$  (See figure 1). Note that if  $y = 0$ ,  $a_0 = c_0$  and so we do not include any nodes from  
 500  $b$  or  $d$ . Similarly, if  $y = 1$ ,  $a_0$  and  $c_0$  are connected by a single edge, as are  $a_{-1}$  and  $c_{-1}$ .

501 There then exists a corresponding message configuration over the even cycle representation.  
 502 Essentially (other than a few technical exceptions), this new configuration is the same as the  
 503 old configuration but with two copies of each message on the path, one on each corresponding  
 504 edge of the even cycle representation. Formally,

505 ► **Definition 19.** Let  $F = (V, E)$  be  $FEC_{x,y,z}$  and  $S$  a configuration of Amnesiac Flooding  
 506 on  $F$ , the even cycle representation of  $S$  on  $F$  denoted  $S_{2,F}$  is determined as follows. For  
 507 each  $m \in S$

- 508 ■ If  $m = (a_{2x}, a_0)$  (resp.  $(a_0, a_{2x})$ ) we add  $(a_{2x}, a_{-1})$  (resp.  $(a_{-1}, a_{2x})$ ) to  $S_{2,F}$ .
- 509 ■ If  $m = (b_i, b_j)$  for some  $i, j \in \{1, \dots, y-1\}$ , we add both  $(b_i, b_j)$  and  $(d_i, d_j)$  to  $S_{2,F}$ .
- 510 ■ If  $m = (c_{2z}, c_0)$  (resp.  $(c_0, c_{2z})$ ) we add  $(c_{2z}, c_{-1})$  (resp.  $(c_{-1}, c_{2z})$ ) to  $S_{2,F}$ .
- 511 ■ If  $m = (a_0, b_1)$  (resp.  $(b_1, a_0)$ ) we add both  $(a_0, b_1)$  and  $(a_{-1}, d_1)$  (resp.  $(b_1, a_0)$  and  
 512  $(d_1, a_{-1})$ ) to  $S_{2,F}$ .
- 513 ■ If  $m = (c_0, b_{y-1})$  (resp.  $(b_{y-1}, c_0)$ ) we add both  $(c_0, b_{y-1})$  and  $(c_{-1}, d_{y-1})$  (resp.  $(b_{y-1}, c_0)$   
 514 and  $(d_{y-1}, c_{-1})$ ) to  $S_{2,F}$ .
- 515 ■ If  $m = (a_0, c_0)$  (resp.  $(c_0, a_0)$ ) we add both  $(a_0, c_0)$  and  $(a_{-1}, c_{-1})$  (resp.  $(c_0, a_0)$  and  
 516  $(c_{-1}, a_{-1})$ ) to  $S_{2,F}$ .
- 517 ■ Otherwise we add  $m$  to  $S_{2,F}$ .

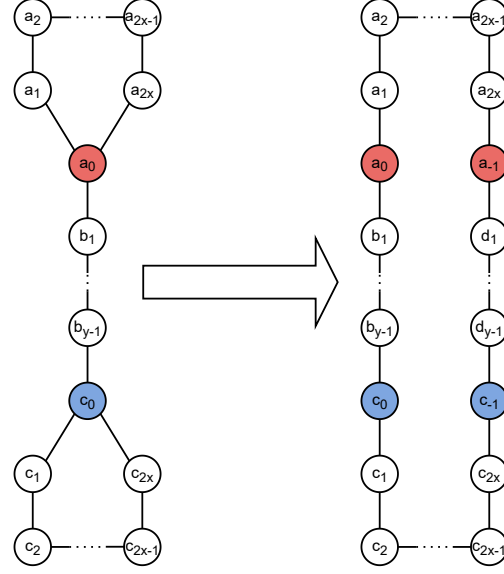
518 With this established we can now define the notion of balance.

519 ► **Definition 20.** A configuration  $S$  is balanced on  $G = (V, E)$  if for all subgraphs  $H$  of  $G$   
 520 one of the following holds:

- 521 ■  $H$  is not a cycle or FEC.
- 522 ■  $H$  is an odd cycle and  $S_H$  contains an equal number of messages travelling clockwise and  
 523 anti-clockwise on  $H$ .
- 524 ■  $H$  is an even cycle and for any given message  $m$  in  $S_H$ , there is an equal number of  
 525 messages travelling clockwise and anti-clockwise on  $H$  such that their heads are an even  
 526 distance from  $m$ 's.
- 527 ■  $H$  is an FEC and  $S_{2,H}$  is balanced on  $H_2$  (i.e. obeys the previous condition).

528 With these definitions established, we can present the intuition behind the proof of Theo-  
 529 rem 15.

530 **Sketch of the proof of Theorem 15.** We first establish that balance, and therefore imbal-  
 531 ance, is conserved by Amnesiac flooding and, as the empty configuration is balanced, Amnesiac



■ **Figure 1** Left: An  $FEC_{x,y,z}$ . Right: The corresponding even cycle representation. Please note that this depiction only holds for  $y \geq 2$ . For  $y = 1$ :  $a_0$  and  $c_0$  are connected directly by an edge in both sub figures (as are  $a_{-1}$  and  $c_{-1}$ ). For  $y = 0$ :  $a_0 = c_0$  and  $a_{-1} = c_{-1}$ .

532 Flooding cannot terminate from any imbalanced configuration. For Amnesiac Flooding not  
 533 to terminate it requires that some message travels around the communication graph and  
 534 returns to the same edge, in the same direction. We show that if a configuration is balanced,  
 535 the trajectory of any message can spend only a bounded number of consecutive steps on any  
 536 given cycle or FEC. However, we can also show that any message's trajectory which crosses  
 537 the same edge twice in the same direction, must have spent a large number of consecutive  
 538 steps on some cycle or FEC, and therefore could not have begun in a balanced configuration.  
 539 Thus, Amnesiac Flooding started from any balanced configuration must terminate. The  
 540 proof is given in appendix B. ◀

## 541 5.2 Applying the termination dichotomy

542 In this section, we apply the dichotomy to obtain a number of results.

### 543 5.2.1 Extended Dichotomy

544 While Theorem 15 provides a full dichotomy over the configuration space of Amnesiac  
 545 Flooding and is easier to reason about than previous results, the definition is somewhat  
 546 unwieldy. In this subsection, we demonstrate the effectiveness of the dichotomy and unify it  
 547 with the existing results [15, 16, 28]. It has previously been observed that running Amnesiac  
 548 Flooding backwards obtains another instance of multi-cast Amnesiac Flooding [15]. Formally,  
 549

550 ► **Definition 21.** Let  $G = (V, E)$  be a graph and  $S$  be a configuration of messages on  $G$ .  
 551 Then  $\bar{S} = \{(v, u) | (u, v) \in S\}$ .

552 The following lemma is similar to the argument made in Corollary 4.6. of [15].

## 17:14 Amnesiac Flooding: Easy to Break, Hard to Escape

► **Lemma 22.** *Let  $G = (V, E)$  be a graph,  $S$  a configuration of messages on  $G$  and  $T = \{u \in V \mid \forall v \in N(u) : (u, v) \in S\}$  the set of source vertices. Then  $A_{T,G}(\overline{A_{\emptyset,G}(\bar{S})}) = S$*

Which gives the following immediately via induction,

► **Lemma 23.** *Let  $G = (V, E)$  be a graph and  $S$  a configuration of messages on  $G$ . Then for any  $k \in \mathbb{N}$  there exists a sequence  $I_1, \dots, I_k \subseteq V$  such that  $A_{I_1,G}(\dots(A_{I_k,G}(\overline{A_{\emptyset,G}^k(\bar{S})}))\dots) = S$ .*

Intuitively, this means that given any configuration  $S$  of Amnesiac Flooding, we can run it backwards through time to some earlier configuration  $S'$ . Further we obtain a sequence of vertex sets  $I_1, \dots, I_k$  that were sinks in the time-reversed process and therefore sources in the forwards process. We can therefore reconstruct  $S$  beginning from  $S'$  via some sequence of fresh multi-casts from  $I_1, \dots, I_k$ . We will use this fact to obtain all configurations from which Amnesiac Flooding terminates (i.e. balanced configurations) from the empty configuration. The following lemma is immediate from the definition of balance (definition 20), as reversing the direction of all messages in a configuration does not affect its balance.

► **Lemma 24.**  *$\bar{S}$  is balanced on  $G$  if and only if  $S$  is balanced on  $G$*

Putting it all together, we obtain the following extension of the dichotomy result, as well as the complement to Theorem 7.

► **Theorem 25.** *Let  $G = (V, E)$  be a graph and  $S$  a configuration of  $G$ , the following are all equivalent:*

1.  $\exists k \in \mathbb{N} : A_{\emptyset,G}^k(S) = \emptyset$
2.  $\exists k \in \mathbb{N}, I_1, \dots, I_k \subseteq V : A_{I_k,G}(\dots(A_{I_1,G}(\emptyset))\dots) = S$
3.  $S$  is balanced on  $G$

**Proof.** The equivalence of (1) and (3) follow immediately from Theorem 15. Further, we have that (2) implies (1) from Theorem 7. Now assume  $S$  is balanced, then by lemma 24, so is  $\bar{S}$ . Thus by Theorem 15 there exists a finite  $k$  such that after  $k$  rounds Amnesiac flooding started from  $\bar{S}$  must terminate, i.e.  $A_{\emptyset,G}^k(\bar{S}) = \emptyset$ . Therefore, by lemma 23 we have a sequence  $I_1, \dots, I_k \subseteq V$  such that  $S = A_{I_1,G}(\dots(A_{I_k,G}(\overline{A_{\emptyset,G}^k(\bar{S})}))\dots) = A_{I_1,G}(\dots(A_{I_k,G}(\emptyset))\dots)$ . Thus, we have (3) implies (2) and the result follows. ◀

### 5.2.2 Fault Sensitivity

In this work we consider three key forms of fault of increasing severity: message dropping, uni-directional link failure and weak-Byzantine failures. Intuitively, these correspond to a set of messages failing to send in a specific round, a link failing in one direction creating a directed edge and a set of nodes becoming transiently controlled by an adversary.

More precisely, let  $\mathbf{S} = (S_i)_{i \in \mathbb{N}}$  be the sequence of actual message configurations on our network. We say that  $\mathbf{S}$  is *fault free* for  $G = (V, E)$  if  $S_{i+1} = A_G(S_i)$  for all  $i \in \mathbb{N}$ . Otherwise, we say it experienced a *fault*. In this case we say  $\mathbf{S}$  has suffered from,

- Message dropping, if there exists  $T \subseteq V^2$  and  $k \geq 1$  such that  $S_{k+1} = A(S_k) \setminus T$  and for all  $i \neq k$ ,  $S_{i+1} = A(S_i)$ . This corresponds to all messages in  $T$  being dropped on round  $k$ .
- Uni-directional link failure, if there exists  $X \subseteq V^2$  such that for all  $i \geq 1$ ,  $S_{i+1} = A(S_i) \setminus X$ . This corresponds to all oriented links in  $X$  failing.



593 ■ Weak-Byzantine failure, if there exists  $Y \subseteq V$  such that for some  $k$  at least twice the  
 594 diameter, for all  $i < k$ ,  $S_{i+1} \setminus \{(u, v) | u \in Y\} = A(S_i) \setminus \{(u, v) | u \in Y\}$ . This corresponds  
 595 to a possible failure where an adversary determines the forwarding decisions of the nodes  
 596 in  $Y$  until round  $k$ .

597 Note that we refer to the Byzantine failures as weak, since they are transient and only  
 598 interfere with the forwarding of the message, not its content. It is obvious to see that in a  
 599 stateless setting there is no way to deal with a Byzantine fault that changes the message  
 600 as there is no method to verify which message is authentic. Intuitively, in our setting,  
 601 Weak-Byzantine agents may choose to send messages to an arbitrary set of neighbours in  
 602 each round and they are all controlled by a single coordinated adversary. We say that a  
 603 Weak-Byzantine adversary with control of a given set of nodes can force some behaviour if  
 604 there exists any weak byzantine failure on that set of nodes producing the forced behaviour.  
 605 We can now express our fault sensitivity results, the proofs of which we defer to the full  
 606 version [2], and begin with an extreme case of single message dropping.

607 ► **Theorem 4** (Single Message Failure). *If single-source Amnesiac Flooding experiences a*  
 608 *single message drop failure for the message  $(u, v)$  then it fails to terminate if and only if*  
 609 *either or both of the following hold:*

610 ■  *$uv$  is not a bridge*

611 ■  *$uv$  lies on a path between vertex disjoint odd cycles*

612 *Moreover, it fails to broadcast if and only if this is the first message sent along  $uv$ ,  $uv$  is a*  
 613 *bridge, and the side of the cut containing  $u$  does not contain an odd cycle.*

614 Thus, Amnesiac Flooding is extremely fault-sensitive with respect to message dropping.  
 615 Secondly, considering uni-directional link failures we obtain the following.

616 ► **Theorem 5** (Uni-directional link failure). *For any graph  $G = (V, E)$  and any initiator set*  
 617  *$I \subsetneq V$  there exists an edge  $e \in E$  such that a uni-directional link failure at  $e$  will cause*  
 618 *Amnesiac Flooding to either fail to broadcast or fail to terminate when initiated from  $I$  on*  
 619  *$G$ . Furthermore, for any non-empty set of uni-directional link failures there exists  $v \in V$*   
 620 *such that, when Amnesiac Flooding is initiated at  $v$ , it will either fail to broadcast or fail to*  
 621 *terminate.*

622 Finally for the weak-Byzantine case.

623 ► **Theorem 6** (Byzantine Failure). *If Amnesiac Flooding on  $G = (V, E)$  initiated from  $I \subsetneq V$*   
 624 *experiences a weak Byzantine failure at  $J \subseteq V \setminus I$ , then the adversary can force:*

625 ■ *Failure to broadcast if and only if  $J$  contains a cut vertex set.*

626 ■ *Non-termination if and only if at least one member of  $J$  lies on either a cycle or a path*  
 627 *between odd-cycles.*

## 628 6 Conclusions and Future Work

629 In this paper, we prove a *uniqueness* result: Under standard synchronous message passing  
 630 assumptions, any strictly stateless deterministic algorithm oblivious to the message content  
 631 which solves terminating broadcast is indistinguishable from *Amnesiac Flooding*. We therefore  
 632 argue due to both its uniqueness and simplicity, that Amnesiac Flooding is a fundamental  
 633 or prototypical broadcast algorithm. We formalise the four properties required for this  
 634 uniqueness to hold, and show that by relaxing each individually one can obtain other correct  
 635 and terminating broadcast algorithms, of which we present several. These present the  
 636 following natural questions: To what extent does Amnesiac Flooding represent a “minimal”

637 broadcast algorithm? Are there identifiable families of algorithms solving terminating  
 638 broadcast with a subset of these restrictions? Are any of these independent of (i.e. not  
 639 derivatives of) Amnesiac Flooding? Lastly, we are not aware of any similar uniqueness  
 640 results in algorithms literature, restricting the number of successful algorithms to just one  
 641 (or even to small finite numbers) - can this result stimulate a study into enumerating distinct  
 642 algorithms for solutions to interesting problems? Note that our model of true statelessness  
 643 rules out trivial extensions to algorithms such as delaying algorithm start or holding messages  
 644 for a certain count. Are there reasonable ways to define distinctiveness i.e. discard ‘trivial’  
 645 extensions to algorithms in less restrictive models than ours?

646 We also obtain an understanding of the structural properties of Amnesiac Flooding. In  
 647 particular, we study its sensitivity to single message drops, uni-directional link failures, and  
 648 weak byzantine collusion, showing it can easily become non-terminating or non-broadcasting  
 649 under such conditions. This is perhaps surprising, as statelessness is frequently associated  
 650 with fault tolerance, such as in the self stabilizing setting. A reasonable interpretation of  
 651 Theorem 15, however, is that Amnesiac Flooding, while locally stateless, depends heavily on  
 652 a distributed “meta-state” contained in the configuration of sent messages. This suggests it  
 653 is unlikely that any minor modification of Amnesiac Flooding will resolve its fragility without  
 654 depending on an entirely different mechanism for termination. In support of this, we note  
 655 that of the four alternatives presented in the proof of Theorem 3, only Random-Flooding is  
 656 meaningfully more robust (and will in fact terminate from any configuration in finite time  
 657 almost surely). Nevertheless, we contend that further exploration of stateless algorithms such  
 658 as Amnesiac Flooding, their properties and related models are important for both theory  
 659 and practice of distributed networks.

## 660 — References —

- 661 1 Hagit Attiya and Jennifer Welch. *Distributed Computing: Fundamentals, Simulations and*  
 662 *Advanced Topics*. John Wiley & Sons, 2004.
- 663 2 Henry Austin, Maximilien Gadouleau, George B. Mertzios, and Amitabh Trehan. Amnesiac  
 664 flooding: Easy to break, hard to escape, 2025. URL: <https://arxiv.org/abs/2502.06001>,  
 665 [arXiv:2502.06001](https://arxiv.org/abs/2502.06001).
- 666 3 Abhik Banerjee, Chuan Heng Foh, Chai Kiat Yeo, and Bu-Sung Lee. Performance improvements  
 667 for network-wide broadcast with instantaneous network information. *J. Netw. Comput.*  
 668 *Appl.*, 35(3):1162–1174, 2012. URL: <https://doi.org/10.1016/j.jnca.2012.01.008>, doi:  
 669 10.1016/J.JNCA.2012.01.008.
- 670 4 Zahra Bayramzadeh, Ajay D. Kshemkalyani, Anisur Rahaman Molla, and Gokarna Sharma.  
 671 Weak amnesiac flooding of multiple messages. In Karima Echihabi and Roland Meyer, editors,  
 672 *Networked Systems - 9th International Conference, NETYS 2021, Virtual Event, May 19-21,*  
 673 *2021, Proceedings*, volume 12754 of *Lecture Notes in Computer Science*, pages 88–94. Springer,  
 674 2021. doi:10.1007/978-3-030-91014-3\_6.
- 675 5 Petra Berenbrink, Robert Elsässer, and Thomas Sauerwald. Randomised broadcasting: Memory  
 676 vs. randomness. *Theor. Comput. Sci.*, 520:27–42, 2014. URL: [https://doi.org/10.1016/j.](https://doi.org/10.1016/j.tcs.2013.08.011)  
 677 [tcs.2013.08.011](https://doi.org/10.1016/j.tcs.2013.08.011), doi:10.1016/J.TCS.2013.08.011.
- 678 6 Armando Castañeda, Jonas Lefèvre, and Amitabh Trehan. Fully compact routing in low  
 679 memory self-healing trees. In Nandini Mukherjee and Sriram V. Pemmaraju, editors, *ICDCN*  
 680 *2020: 21st International Conference on Distributed Computing and Networking, Kolkata, India,*  
 681 *January 4-7, 2020*, pages 21:1–21:10. ACM, 2020. doi:10.1145/3369740.3369786.
- 682 7 Niccolò D’Archivio and Robin Vacus. On the Limits of Information Spread by Memory-Less  
 683 Agents. In Dan Alistarh, editor, *38th International Symposium on Distributed Comput-*  
 684 *ing (DISC 2024)*, volume 319 of *Leibniz International Proceedings in Informatics (LIPIcs)*,

- pages 18:1–18:21, Dagstuhl, Germany, 2024. Schloss Dagstuhl – Leibniz-Zentrum für Informatik. URL: <https://drops.dagstuhl.de/entities/document/10.4230/LIPIcs.DISC.2024.18>, doi:10.4230/LIPIcs.DISC.2024.18.
- 8 Benjamin Doerr, Mahmoud Fouz, and Tobias Friedrich. Social networks spread rumors in sublogarithmic time. In *Proceedings of the Forty-Third Annual ACM Symposium on Theory of Computing*, STOC '11, page 21–30, New York, NY, USA, 2011. Association for Computing Machinery. doi:10.1145/1993636.1993640.
  - 9 Danny Dolev, Michael Erdmann, Neil Lutz, Michael Schapira, and Adva Zair. Stateless computation. In Elad Michael Schiller and Alexander A. Schwarzmann, editors, *Proceedings of the ACM Symposium on Principles of Distributed Computing, PODC 2017, Washington, DC, USA, July 25–27, 2017*, pages 419–421. ACM, 2017. doi:10.1145/3087801.3087854.
  - 10 Robert Elsässer and Thomas Sauerwald. The power of memory in randomized broadcasting. In Shang-Hua Teng, editor, *Proceedings of the Nineteenth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2008, San Francisco, California, USA, January 20–22, 2008*, pages 218–227. SIAM, 2008. URL: <http://dl.acm.org/citation.cfm?id=1347082.1347107>.
  - 11 Maxime Flin, Mohsen Ghaffari, Magnús M. Halldórsson, Fabian Kuhn, and Alexandre Nolin. Coloring fast with broadcasts. In Kunal Agrawal and Julian Shun, editors, *Proceedings of the 35th ACM Symposium on Parallelism in Algorithms and Architectures, SPAA 2023, Orlando, FL, USA, June 17–19, 2023*, pages 455–465. ACM, 2023. doi:10.1145/3558481.3591095.
  - 12 Ajei S. Gopal, Inder S. Gopal, and Shay Kutten. Fast broadcast in high-speed networks. *IEEE/ACM Trans. Netw.*, 7(2):262–275, 1999. doi:10.1109/90.769773.
  - 13 Walter Hussak and Amitabh Trehan. On termination of a flooding process. In Peter Robinson and Faith Ellen, editors, *Proceedings of the 2019 ACM Symposium on Principles of Distributed Computing, PODC 2019, Toronto, ON, Canada, July 29 - August 2, 2019*, pages 153–155. ACM, 2019. doi:10.1145/3293611.3331586.
  - 14 Walter Hussak and Amitabh Trehan. On the termination of flooding. In Christophe Paul and Markus Bläser, editors, *37th International Symposium on Theoretical Aspects of Computer Science, STACS 2020, March 10–13, 2020, Montpellier, France*, volume 154 of *LIPIcs*, pages 17:1–17:13. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2020. URL: <https://doi.org/10.4230/LIPIcs.STACS.2020.17>, doi:10.4230/LIPIcs.STACS.2020.17.
  - 15 Walter Hussak and Amitabh Trehan. Terminating cases of flooding. *CoRR*, abs/2009.05776, 2020. URL: <https://arxiv.org/abs/2009.05776>, arXiv:2009.05776.
  - 16 Walter Hussak and Amitabh Trehan. Termination of amnesiac flooding. *Distributed Comput.*, 36(2):193–207, 2023. URL: <https://doi.org/10.1007/s00446-023-00448-y>, doi:10.1007/S00446-023-00448-Y.
  - 17 Amos Korman and Robin Vacus. Early adapting to trends: self-stabilizing information spread using passive communication. volume 37, pages 335–362, 2024. URL: <https://doi.org/10.1007/s00446-024-00462-8>, doi:10.1007/S00446-024-00462-8.
  - 18 Nancy A. Lynch. *Distributed Algorithms*. Morgan Kaufmann, 1996.
  - 19 Victoria Manfredi, Mark Crovella, and Jim Kurose. Understanding stateful vs stateless communication strategies for ad hoc networks. In Parmesh Ramanathan, Thyaga Nandagopal, and Brian Neil Levine, editors, *Proceedings of the 17th Annual International Conference on Mobile Computing and Networking, MOBICOM 2011, Las Vegas, Nevada, USA, September 19–23, 2011*, pages 313–324. ACM, 2011. doi:10.1145/2030613.2030649.
  - 20 Garrett Parzych and Joshua J. Daymude. Memory Lower Bounds and Impossibility Results for Anonymous Dynamic Broadcast. In Dan Alistarh, editor, *38th International Symposium on Distributed Computing (DISC 2024)*, volume 319 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 35:1–35:18, Dagstuhl, Germany, 2024. Schloss Dagstuhl – Leibniz-Zentrum für Informatik. URL: <https://drops.dagstuhl.de/entities/document/10.4230/LIPIcs.DISC.2024.35>, doi:10.4230/LIPIcs.DISC.2024.35.
  - 21 David Peleg. *Distributed Computing: A Locality Sensitive Approach*. SIAM, 2000.

- 736 22 Stanisław Radziszowski. Small ramsey numbers. *The electronic journal of combinatorics*,  
737 pages DS1–Sep, 2012.
- 738 23 Patricia Ruiz and Pascal Bouvry. Survey on broadcast algorithms for mobile ad hoc networks.  
739 *ACM Comput. Surv.*, 48(1):8:1–8:35, 2015. doi:10.1145/2786005.
- 740 24 Andrew Tanenbaum. *Computer networks*. Pearson Prentice Hall, Boston, 2011.
- 741 25 Gerard Tel. *Introduction to distributed algorithms*. Cambridge University Press, New York,  
742 NY, USA, 1994.
- 743 26 Yu-Chee Tseng, Sze-Yao Ni, Yuh-Shyan Chen, and Jang-Ping Sheu. The broadcast storm  
744 problem in a mobile ad hoc network. volume 8, pages 153–167, 2002. doi:10.1023/A:  
745 1013763825347.
- 746 27 Volker Turau. Stateless information dissemination algorithms. In Andréa Werneck Richa and  
747 Christian Scheideler, editors, *Structural Information and Communication Complexity - 27th*  
748 *International Colloquium, SIROCCO 2020, Paderborn, Germany, June 29 - July 1, 2020,*  
749 *Proceedings*, volume 12156 of *Lecture Notes in Computer Science*, pages 183–199. Springer,  
750 2020. doi:10.1007/978-3-030-54921-3\_11.
- 751 28 Volker Turau. Amnesiac flooding: Synchronous stateless information dissemination. In Tomás  
752 Bures, Riccardo Dondi, Johann Gamper, Giovanna Guerrini, Tomasz Jurdzinski, Claus Pahl,  
753 Florian Sikora, and Prudence W. H. Wong, editors, *SOFSEM 2021: Theory and Practice*  
754 *of Computer Science - 47th International Conference on Current Trends in Theory and*  
755 *Practice of Computer Science, SOFSEM 2021, Bolzano-Bozen, Italy, January 25-29, 2021,*  
756 *Proceedings*, volume 12607 of *Lecture Notes in Computer Science*, pages 59–73. Springer, 2021.  
757 doi:10.1007/978-3-030-67731-2\_5.
- 758 29 Volker Turau. Synchronous concurrent broadcasts for intermittent channels with bounded  
759 capacities. In Tomasz Jurdzinski and Stefan Schmid, editors, *Structural Information and*  
760 *Communication Complexity - 28th International Colloquium, SIROCCO 2021, Wrocław,*  
761 *Poland, June 28 - July 1, 2021, Proceedings*, volume 12810 of *Lecture Notes in Computer*  
762 *Science*, pages 296–312. Springer, 2021. doi:10.1007/978-3-030-79527-6\_17.

## APPENDIX

In this section we present the proof of our main technical results, i.e. theorems 14 and 15, as well as their constituent lemmas. Due to space constraints, we defer the proofs of all such lemmas as well as the correctness of all other results to the full version. Additionally, we will refer to Amnesiac Flooding as simply *AF* throughout.

### A Proof of Uniqueness (Theorem 14)

We begin with the following observations, which follow from case analysis on paths:

#### ► Observation 26.

1. For all  $\emptyset \neq S \subset \mathbb{N}$  and  $u \in \mathbb{N}$ ,  $b(u, S) \neq \emptyset$ . Otherwise no new nodes will be informed after the first round, violating correctness.
2. For  $u, v, w \in \mathbb{N}$ ,  $f(v, \{u, w\}, \{u\}) \in \{\{w\}, \{u, w\}\}$ . Otherwise, if  $uvw$  is a subsequence of a path the message started from  $u$  will not reach  $w$ .
3. For  $u, v \in [\kappa]$ , if  $f(u, \{v\}, \{v\}) = v$  then  $f(v, \{u\}, \{u\}) = \emptyset$ . Otherwise  $P$  will not terminate on the two node path labelled with  $u$  and  $v$ .
4. For  $u, v, w \in [\kappa]$  if  $f(u, \{v\}, \{v\}) = \{v\}$  and  $f(w, \{v\}, \{v\}) = \{v\}$  then  $f(v, \{u, w\}, \{u\}) = \{u, w\}$  and/or  $f(v, \{u, w\}, \{w\}) = \{u, w\}$ . Additionally,  $f(v, \{u, w\}, \{u, w\}) = \emptyset$ . Otherwise the protocol would not terminate on the three node path with labels  $uvw$ .

Via a combinatorial argument followed by case analysis, we show the existence of a small set of labels which when restricted to,  $P$  behaves identically to *AF* for subcubic graphs.

► **Lemma 27.** *There exists  $T \subseteq [\kappa]$  such that  $|T| \geq 6$  and  $P$  is *AF* on  $T$  up to degree 3.*

With this established we can extend the result to all labels via an inductive argument. Essentially, we are able to construct a sequence of graph-labelling pairs that grow to include all labels with all possible neighbourhoods. At each state, by our constructions and the induction hypothesis, at most four vertices may behave distinctly from *AF*. By careful construction, however, any distinct behaviour will lead to an incorrect algorithm.

► **Lemma 28.**  *$P$  is *AF* on  $\mathbb{N}$  up to degree 3.*

The main proof makes use of a very similar argument. However, we now construct graphs with one high degree vertex, while the rest all have degree at most three and so behave indistinguishably from *AF*.

**Proof of Theorem 14.** By lemma 28 we have that  $P$  is *AF* up to degree 3 on  $\mathbb{N}$ . It is immediate that for all  $u \in \mathbb{N}$ ,  $S \subset \mathbb{N}$   $b(u, S) = S$  as  $u$  could be at the centre of a star with one of its leaves replaced by a long path. In this case,  $u$  will never receive a message again and so must send to all of its neighbours in the first round.

For an arbitrary label  $u \in \mathbb{N}$  and degree  $k \in \mathbb{N}$  we can show that  $u$  must behave as though it is implementing *AF* if it is at a node of degree  $k$ . Take  $S \subset \mathbb{N}$  such that  $|S| = k$ , we will show that for all non-empty  $T \subseteq S$ ,  $f(u, S, T) = S \setminus T$  via induction on the size of  $T$ .

Beginning with our base case. If  $u$  receives from a single neighbour we can construct a pair of graphs (special cases of our general construction) that enforce the *AF* policy. Consider the tree in figure 2a, if  $u$  receives a message from  $v$  it must send to at least  $x_1, \dots, x_{k-1}$  as  $u$  will receive messages in only a single round. This follows as the rest of tree will use *AF* policies and since the graph is bipartite each node will be active only once. Thus, the only

two options for  $f(u, S, \{v\})$  are  $S$  or  $S \setminus \{v\}$ . Now consider the second graph from figure 2a and a broadcast initiated at  $u$ . We can see that if  $f(u, S, \{v\}) = S$  then the cycle and  $u$  will simply pass a message back and forth forever. Thus,  $f(u, S, \{v\}) = S \setminus \{v\}$ .

We now generalize this construction and perform our induction. Assume that for any non-empty subset  $T$  of  $S$  of size at most  $q$ ,  $f(u, S, T) = S \setminus T$ . For the sake of contradiction assume that this is not true for  $V$  where  $T \subset V \subseteq S$  with  $|V| = q + 1$ . Thus either  $f(u, S, V) \cap V \neq \emptyset$  or  $f(u, S, V) \cup V \neq S$ .

In the first, case let  $W = V \cap f(u, S, V)$ , and for some ordering label the elements of  $W$ :  $w_1, \dots, w_r$  and the elements of  $V \setminus W$ :  $v_1, \dots, v_{q+1-r}$ .

If  $W = V$ , then we construct a communication graph as in figure 2b and consider a broadcast initiated at  $u$ . The messages will travel in only one direction through the binary tree and so when  $w^*$  receives a message it will be the only message on the graph. This message will then be passed onto the cycle where it will circulate, before being passed back onto the binary tree in the opposite direction. Again the policy of all nodes other than  $u$  is indistinguishable from AF and so when  $u$  next receives a message, it receives from all of  $V$  and these are the only messages (outside of the path from  $c$  to  $d$ ). We therefore have a repeating sequence as  $u$  will forward the message back to every node of  $W$ .

If  $W \subset V$ , then we construct a slightly different communication graph (see figure 2c). This graph instead partitions  $S$  into  $W$ ,  $V \setminus W$  and  $S \setminus V$ , with separate binary trees for  $W$  and  $V \setminus W$ . Here we consider a broadcast initiated at  $a$ . When  $u$  first receives a message it will receive the message from all identifiers in  $V$  and there will be no other messages in the body of the graph. Then by assumption  $u$  will send a message to some portion of  $\{l_1, \dots, l_{k-q-1}\}$  as well as all of  $W$ . The leaves will not respond and messages will travel only upwards in the binary tree with  $W$  as leaves and  $w^*$  as its root. Thus,  $a$  will next receive a message only from  $w^*$  and will send to  $v^*$  and  $c$ . Until  $u$  receives a message the only messages in the body of the graph will be those travelling down the binary tree with  $V \setminus W$  as its leaves and they will all arrive at  $u$  simultaneously. Thus,  $u$  will then receive only from  $V \setminus W$ . Since  $|V \setminus W| < |V| = q + 1$  by assumption  $u$  must make the same decision as AF and so will send messages to all of  $W$  and its leaves. This creates a repeating sequence and so we have non-termination. Thus, since  $f(u, S, V) \cap V \neq \emptyset$  always allows us to construct a communication graph with a non-terminating broadcast, we must have that  $f(u, S, V) \cap V = \emptyset$ . Now consider the communication graph from figure 2c again but with  $W$  an arbitrary strict subset of  $V$ . Since  $f(u, S, V)$  does not contain any id from  $V$  and none of  $\{l_1, \dots, l_{k-q-1}\}$  will send a message back to  $u$ ,  $u$  sends messages in only a single round. Therefore,  $u$  must send to all of  $\{l_1, \dots, l_{k-q-1}\}$  otherwise some would not receive the message (and so the protocol would not implement broadcast correctly). Thus,  $f(u, S, V) = S \setminus V$  and so we have our contradiction.

Therefore, if  $f(u, S, T) = S \setminus T$  for  $S \subset \mathbb{N}$  where  $|S| = k$  and all  $T \subseteq S$  such that  $0 < |T| \leq q$ , then  $f(u, S, V) = S \setminus V$  for  $V \subset S$  such that  $|V| = q + 1$ . Thus, by induction since we know this to be true for all  $u$  and  $S$  when  $q = 1$  it must hold for all  $q \leq |S|$ .

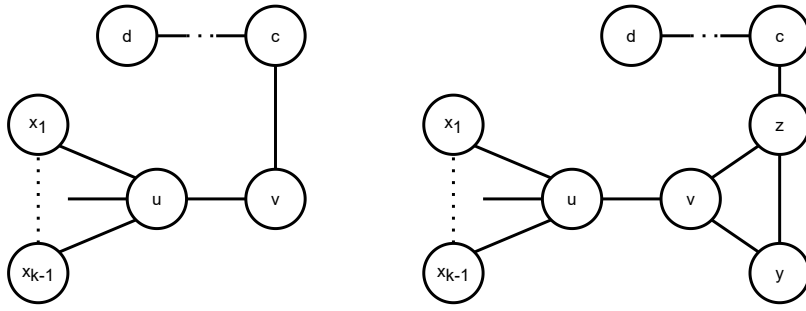
This gives our claim, as for every  $u$  and  $k$  we can apply this argument and show that for any  $k \in \mathbb{N}$ ,  $P$  is AF up to degree  $k$ . ◀

## **B Proof of the Dichotomy (Theorem 15)**

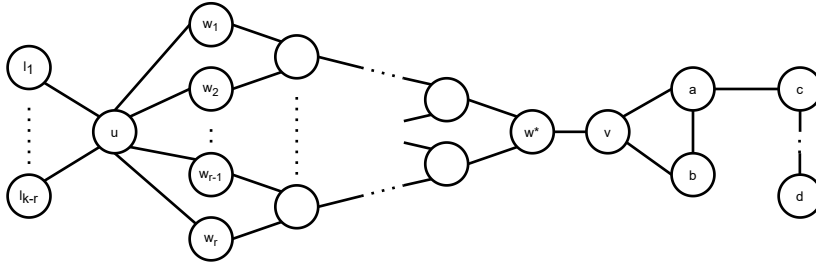
We begin by showing that (im)balance is preserved by the operation of AF,

► **Lemma 29.** *For any set of initiators  $I \subseteq V$  and configuration  $S$  on  $G = (V, E)$ ,  $A_{I,G}(S)$  is balanced if and only if  $S$  is balanced.*

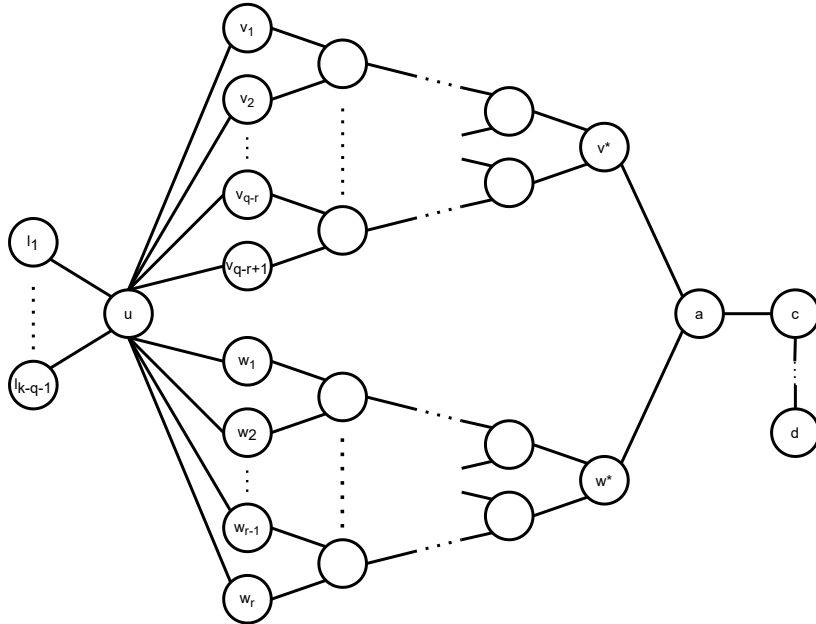




(a) Two graphs used in the proof of theorem 14 to determine identifiers' response to receiving only a single message. Left: A tree that forbids sending to too small a subset of neighbours. Right: A graph that forbids sending a message to all neighbours.



(b) A graph used in the proof of theorem 14. The graph consists of a star centred at  $u$  with its leaves partitioned into two sets of size  $r$  and  $k - r$ . The leaves in the set of size  $r$  are connected by a binary tree of depth  $\lceil \log_2 r \rceil$  with root  $w^*$  to a cycle, which in turn is connected to a path.



(c) A graph used in the proof of Theorem 14. The graph consists of a star centred at  $u$  with its leaves partitioned into three sets of size  $r$ ,  $q - r + 1$  and  $k - q - 1$ . The leaves in the first two sets are then each joined to a single node labelled by  $w^*$  and  $v^*$  respectively by binary trees of depth  $\lceil \log_2 q \rceil$ . The single nodes are connected to a node labelled  $a$  which is the start of a path  $ac...d$ .

■ **Figure 2** Note that in all figures the path  $c...d$  contains all identifiers in  $[m]$  not used in labelling the body of the graph, where  $m$  is the largest id in the whole labelling.

## 17:22 Amnesiac Flooding: Easy to Break, Hard to Escape

850 This gives the following immediate corollary, as  $\emptyset$  is trivially balanced.

851 ► **Corollary 30.** *If  $S$  is imbalanced on  $G = (V, E)$ , then for all  $k > 0$ ,  $A_G^k(S) \neq \emptyset$ .*

852 This gives us the forward direction of Theorem 15. For the other direction, we need the  
853 notion of message paths and their recurrence.

854 ► **Definition 31.** *A message  $m = (v_0, v_1)$  in configuration  $S \subset V^2$  has a message path  $v_0 v_1$ .  
855 We define the rest of its paths recursively, i.e.  $m = (v_0, v_1)$  has a message path  $v_0 \dots v_{k+1}$  from  
856  $S$  on  $G$  if:*

857 ■  $v_0 v_1 \dots v_k$  is a message path of  $m$  in  $S$

858 ■ The message  $(v_k, v_{k+1})$  exists in  $A_G^k(S)$

859 We say that a message  $m = (v_0, v_1)$  is recurrent on  $G = (V, E)$  from  $S$  if  $m$  has a message  
860 path of the form  $v_0 v_1 \dots v_0 v_1$  on  $G$  from  $S$ .

861 We obtain the following property relating message paths and termination immediately.

862 ► **Lemma 32.** *Let  $S$  be a non-empty configuration on  $G = (V, E)$  such that  $A_G^k(S) = S$ ,  
863 then  $S$  contains a recurrent message on  $G$ .*

864 The following theorem connects the notions of imbalance and recurrence.

865 ► **Theorem 33.** *Let  $S$  be a configuration on  $G$ ,  $S$  is imbalanced if and only if it contains a  
866 recurrent message.*

867 Since, all non-terminating configurations must either contain a recurrent message or eventually  
868 reach a configuration with a recurrent message, we have that all balanced configurations  
869 must terminate. Thus, Theorem 15 follows immediately from Theorem 33. The forward  
870 direction is itself immediate from Corollary 30. For the reverse we take  $G = (V, E)$  to  
871 be a communication graph and  $S \subseteq V^2$  to be a balanced configuration of messages. For  
872 contradiction we assume that  $S$  contains a recurrent message  $m$  which has a message path  $W$   
873 performing exactly one excursion and return to  $m$ . We will view  $W = w_0 w_1 \dots w_0 w_1$  as both  
874 a walk on  $G$  and a word. The key observation we require is that the number of consecutive  
875 steps that can be spent on certain subgraphs by a message path is bounded from a balanced  
876 configurations.

877 ► **Lemma 34.**  *$W$  must obey the following rules:*

- 878 1.  $W$  cannot return to the node it just came from, i.e. no sequence of the form  $uvu$ .
- 879 2.  $W$  cannot take  $2x + 2$  consecutive steps around an odd cycle of length  $2x + 1$ .
- 880 3.  $W$  cannot take  $x + 1$  consecutive steps around an even cycle of length  $2x$ .
- 881 4. If  $W$  takes more than  $x + y + z + 2$  consecutive steps on an FEC it remains on one cycle.
- 882 5. If  $W$  takes  $x$  steps around an even cycle of length  $2x$ , then there exists  $W'$  which takes  
883 the opposite path of equal length and is otherwise identical.

884 The fifth of these has the following useful interpretation, if the existence of  $W'$  implies  
885 imbalance then the existence of  $W$  implies imbalance. Therefore, we will use this rule as a  
886 substitution allowing us to "modify"  $W$  to take the alternate path. By application of these  
887 rules we can obtain the following further conditions:

888 ► **Lemma 35.**  *$W$  must contain the following four rules:*

- 889 1. If  $W$  contains a factor  $u \dots u$  then that factor contains an odd cycle as a subfactor.
- 890 2. There can only be one factor of  $W$  that forms an odd cycle.
- 891 3. Every node appears at most twice in  $W$ .

892 4. *There exists at most one node that is both a member of a consecutive odd cycle and appears*  
 893 *twice. Furthermore, such a node must be the start and end of the cycle.*

894 No matter how we construct  $W$ , it will violate one of these rules.

895 **Proof of the reverse direction of Theorem 33.** Since  $W$  must have the form  $w_0w_1\dots w_0w_1$ ,  
 896 it follows from Lemma 35 (3) that there exists a cycle  $C$  containing  $m$  such that  $W$  fully  
 897 traverses  $C$  before returning to  $m$ , with possible excursions. Specifically, there exists a  
 898 sequence of pairs  $(w_0, w_1), (w_1, w_2), \dots, (w_k, w_0), (w_0, w_1)$  such that  $C = w_0w_1\dots w_kw_0$  is a cycle  
 899 of  $G$ , each pair appears in  $W$  in consecutive order (i.e.  $W = w_0w_1\dots w_1w_2\dots w_kw_0\dots w_0w_1$ ).

900 ▷ **Claim 36.**  $C$  is an odd cycle

901 **Proof.** If  $C$  is of even length (say  $2k$ ), then there must be an excursion from  $C$  otherwise  
 902 Lemma 34 (3) would be violated. However, by Lemma 35 (1) and (2), there can be at most  
 903 one such excursion as it must contain a consecutive odd cycle. Thus, the two subsequences  
 904 on either side must be factors of  $W$ . Therefore, since some subsequence of  $W$  traverses  $C$   
 905 fully with one additional step, one of the two factors must take  $k + 1$  steps around  $C$ . This  
 906 also violates Lemma 34 (3) and so  $C$  must be an odd cycle. ◁

907 ▷ **Claim 37.**  $W$  consists of two odd cycles  $C$  and  $\hat{C}$  connected by a path.

908 **Proof.** If  $C$  is an odd cycle there must be an excursion from it or Lemma 34 (2) would be  
 909 violated. By Lemma 35 (1) and (2) the excursion must contain exactly one consecutive odd  
 910 cycle which we denote by  $\hat{C}$ . If  $\hat{C}$  does not share its starting node with  $C$ ,  $W$  either forms  
 911 a path between  $C$  and  $\hat{C}$  or some chain of cycles. We can use Lemma 34 (5) to eliminate  
 912 all even cycles of this chain, after which any odd cycles in the chain correspond to a fully  
 913 traversed  $FEC$  when paired with  $\hat{C}$  and so violate Lemma 34 (4). Thus,  $W$  takes a simple  
 914 path from  $C$  to  $\hat{C}$  and back, although possibly intersecting  $C$  along the way. ◁

915 ▷ **Claim 38.**  $C$  and  $\hat{C}$  intersect with each other but not the path between them.

916 **Proof.** If the path to  $\hat{C}$  does intersect  $C$ , since we are taking the same path in both directions  
 917 any node shared between the path and  $C$  appears in  $W$  three times. This violates Lemma 35  
 918 (3) and so  $C$  must be disjoint from the path to  $\hat{C}$ . Similarly  $\hat{C}$  must be disjoint from the  
 919 path otherwise it would violate the same lemma. If  $C$  is disjoint from  $\hat{C}$  the pair would form  
 920 a fully traversed  $FEC$ , thereby violating Lemma 34 (4). Thus,  $C$  and  $\hat{C}$  must intersect. ◁

921 ▷ **Claim 39.** Claim:  $C$  and  $\hat{C}$  do not intersect.

922 **Proof.** Let  $W = uv\dots wx_1, \dots, x_kw\dots uv$  where  $C = u\dots w\dots uv$  and the excursion to  $\hat{C}$  is given  
 923 by  $wx_1\dots x_kw$ . Now assume that  $\hat{C}$  contains a node from  $C$  which occurs strictly before  $w$  in  
 924  $W$ . This node is on a consecutive odd cycle and appears twice. There must exist a latest  
 925 such node in the ordering of  $C$ , we denote it  $y$ . Since  $y$  is on a consecutive odd cycle and  
 926 appears twice in  $W$  it must be the start and end point of  $\hat{C}$  by Lemma 35 (4). However,  
 927 then since  $y \neq w$  it must appear three times, violating Lemma 35 (3). The same argument  
 928 holds taking the earliest node shared by  $C$  and  $\hat{C}$  strictly after  $w$ . Thus, the only node that  
 929 can be shared by  $C$  and  $\hat{C}$  is  $W$ , implying that  $W$  forms two odd cycles sharing a single  
 930 node. However, this is an  $FEC$  which is fully traversed and so violates Lemma 34 (4). ◁

931 Thus,  $W$  cannot exist and so there can be no recurrent message in  $S$  ◀