Round-asynchronous amnesiac flooding

Oluwatobi Alafin[1], George B. Mertzios[2*][0000−0001−7182−585X], and Paul G. Spirakis[1**][0000−0001−5396−3749]

[1] Department of Computer Science, University of Liverpool, UK
o.f.alafin@liverpool.ac.uk, p.spirakis@liverpool.ac.uk
[2] Department of Computer Science, Durham University, UK
george.mertzios@durham.ac.uk

Abstract. We present a comprehensive analysis of Round-Asynchronous Amnesiac Flooding (RAAF), a variant of Amnesiac Flooding that introduces round-based asynchrony through adversarial delays. We establish fundamental properties of RAAF, including termination characteristics for different graph types and decidability results under various adversarial models. Our key contributions include: (1) a formal model of RAAF incorporating round-based asynchrony, (2) a proof that flooding always terminates on acyclic graphs despite adversarial delays, (3) a construction showing non-termination is possible on any cyclic graph, (4) a demonstration that termination is undecidable with arbitrary computable adversaries, and (5) the introduction of Eventually Periodic Adversaries (EPA) under which termination becomes decidable. These results enhance our understanding of flooding processes in asynchronous settings and provide insights for designing robust distributed protocols.

Keywords: flooding protocol · amnesiac flooding · asynchronous protocol

1 Introduction

Flooding algorithms [2] serve as fundamental primitives in distributed computing for information dissemination, with applications ranging from network discovery to emergency broadcast systems. While traditional flooding maintains message histories to prevent redundant transmissions [3], such approaches become impractical in resource-constrained environments like sensor networks, IoT devices, or networks with high churn rates where maintaining consistent state is challenging.

Amnesiac Flooding (AF) addresses these limitations by eliminating message history, requiring nodes to make forwarding decisions based solely on current information [6]. However, existing AF analyses assume perfect synchrony—an unrealistic assumption in practical networks where delays, failures, and asynchrony are the norm rather than the exception.

* Supported by the EPSRC grant EP/P020372/1.
** Supported by the EPSRC grant EP/P02002X/1.

This paper introduces *Round-Asynchronous Amnesiac Flooding (RAAF)*, which bridges the gap between theoretical AF models and practical network conditions. RAAF maintains the memory-efficiency of amnesiac approaches while incorporating realistic asynchronous behaviour through adversarial delays; thus, our model can be also termed *Round-Delayed Amnesiac Flooding (RDAF)*. Our key insight is that even with minimal state and adverse conditions, we can characterise precise conditions under which flooding terminates, providing both positive results (guaranteed termination in acyclic networks) and fundamental limitations (undecidability with arbitrary adversaries).

The significance of our results extends beyond flooding protocols. By establishing when termination analysis becomes undecidable and identifying restricted adversary models (Eventually Periodic Adversaries) where it remains decidable, we contribute to the broader understanding of computability limits in asynchronous distributed systems.

While our model makes specific assumptions (such as nodes detecting blocked edges), these capture realistic scenarios and enable rigorous analysis of fundamental limits. The dichotomy between acyclic and cyclic graphs, and between arbitrary and periodic adversaries, reveals deep structural properties that inform the design of practical flooding protocols.

1.1 Model Context and Novelty

RAAF occupies a unique position in the spectrum of distributed system models. Unlike classical asynchronous models that allow arbitrary message delays and reorderings [3], or partially synchronous models that impose eventual bounds on communication delays [5], RAAF maintains a synchronous round structure while allowing adversarial edge-level asynchrony within rounds.

Adversarial Model Justification A key aspect of our model is that nodes are aware of which outgoing edges are currently delayed by the adversary. While this may initially seem like a strong assumption, it captures several practical scenarios:

- **Failed transmission detection**: In many network protocols, nodes receive acknowledgments or can detect transmission failures through timeout mechanisms or carrier sensing.
- **Scheduled maintenance**: In managed networks, nodes may be informed of temporary link unavailability due to scheduled maintenance or known congestion patterns.
- **Visible network conditions**: In wireless networks, nodes can often detect poor channel conditions or interference that prevents successful transmission.

This modeling choice allows us to study the fundamental limits of flooding under adversarial conditions while maintaining some feedback about the network state. Alternative models where nodes lack this information would require additional mechanisms (such as acknowledgments or timeouts) that would fundamentally change the nature of the flooding protocol.

Memory Model and Amnesiac Nature The term “amnesiac” in our context requires careful interpretation. Traditional amnesiac flooding assumes that nodes retain *no state* between rounds. Our variant relaxes this to what we call *structured amnesia*: nodes forget all message history but maintain a bounded amount of state (destination sets) that is recomputed based on current round information. Specifically:

- Nodes do not remember which nodes they have received messages from in previous rounds.
- Nodes only maintain destination sets that are *functionally determined* by the current round’s receipts and delays.

This structured amnesia is motivated by resource-constrained environments where maintaining full message history is infeasible, but nodes can afford $O(|N(v)|)$ memory for immediate forwarding decisions, where $N(v)$ denotes the set of neighbours of node v. This represents a middle ground between full amnesia and traditional flooding.

1.2 Our Contributions

This paper makes several significant contributions:

1. **Formal Model**: A comprehensive mathematical framework for RAAF, including precise definitions for system state, adversarial delay functions, and termination conditions.
2. **Termination Analysis**: Proof that RAAF always terminates on acyclic graphs with provable bounds, and demonstration that any cyclic graph admits non-termination.
3. **Decidability Results**: Proof of undecidability for arbitrary computable adversaries via reduction from the halting problem, and introduction of the Eventually Periodic Adversary (EPA) model under which termination becomes decidable.

1.3 Organisation

Section 2 discusses related work. Section 3 presents our formal model and defines key properties. Section 4 demonstrates non-termination in cyclic graphs. Section 5 develops the theory of periodic infinite schedules, providing a framework for analysing recurrent behaviour. Section 6 proves undecidability for arbitrary computable adversaries. Section 7 introduces Eventually Periodic Adversaries (EPA), establishes decidability and provides complexity bounds. In the full version of the paper we show that all non-terminating schedules in the EPA model are periodic infinite schedules.

2 Related Work

Amnesiac Flooding originated with Hussak and Trehan's work [7], establishing fundamental properties in synchronous settings. Their analysis proved termination bounds—exactly e rounds for bipartite and between e and $e + d + 1$ rounds for non-bipartite graphs (where e is source eccentricity, d is graph diameter)—demonstrating AF's asymptotic time optimality against the $\Omega(d)$ broadcast lower bound.

Turau [11] revealed deeper complexity aspects through the (k, c)-flooding problem: finding k nodes to guarantee termination within c rounds under concurrent flooding. Its NP-completeness highlighted inherent optimisation challenges. Sharp bounds showed significant disparities between bipartite and non-bipartite graphs, introducing a behaviour-preserving construction mapping between them.

Hussak and Trehan extended their analysis [8] to multi-source scenarios, proving $e(I)$-round termination for I-bipartite graphs with source set I. Their fixed-delay analysis showed termination by round $2d+\tau-1$ for single-edge delays of duration τ in bipartite graphs and established termination for multiple-edge fixed delays in cycles.

Bayramzadeh et al. [4] proved termination for multiple-message AF in the unranked full-send case, previously conjectured non-terminating, showing $D \cdot (2k-1)$ rounds for bipartite and $(2D+1)\cdot(2k-1)$ rounds for non-bipartite graphs (k messages). Their introduction of graph diameter knowledge as a parameter suggested new model variants.

Comparison with Asynchronous Flooding Models. Hussak and Trehan [7] briefly discuss an asynchronous variation of amnesiac flooding, and they demonstrate with a small example that this variation does not guarantee termination, in contrast to their synchronous model. In this model of [7], the adversary can decide a delay of message delivery on any link. Once a node sends a message, this message will definitely be delivered at some future round. In contrast, in our model, if a message from node u to node v is delayed by the adversary, the initiator node u keeps a note of this and tries to re-send the message to v again and again, until either (i) the message is delivered to v, or (ii) u receives the message from v, in which case u stops trying to send the message to v.

The possibility of a message not being delivered, due to the last case, makes non-termination in our model much less trivial, compared to [7]. We comprehensively investigate this model and we provide a periodic-schedule normal form for it: every infinite execution can be compressed into an ultimately periodic delay pattern, which in turn enables decidability and undecidability results. Summarizing, our model is complementary to the model of [7], and our work provides a rigorous framework that delineates the exact boundary between terminating and non-terminating behaviour.

Comparison with Stateless Flooding Approaches. The stateless flooding algorithm by Adamek et al. [1] achieves statelessness through a different mechanism than our structured amnesia. Their algorithm uses send queues where messages are discarded upon encountering "mates"—pairs of messages with swapped sender/receiver addresses. Crucially, their model assumes fair scheduling where every queued message is eventually transmitted or removed, without adversarial interference. This synchronous assumption fundamentally differs from our round-asynchronous model where an adversary controls edge availability.

While Adamek et al. prove termination under fair scheduling, we establish when termination remains decidable despite adversarial delays (EPA model) or becomes undecidable (arbitrary computable adversaries). Our structured amnesia—recomputing destination sets based on current round information—provides a framework for analysing flooding under hostile scheduling conditions that previous stateless approaches did not consider.

3 Model, Preliminaries and Notation

We first present the computational model for round-based asynchronous systems, then describe the RAAF protocol that operates within this model.

3.1 Computational Model

Network Structure. We consider a simple, finite, connected graph $G = (V, E)$ with a distinguished source node $g_0 \in V$ possessing initial message m_0. For every node v we denote by $N(v)$ the set of neighbours of v.

Round-Based Asynchrony. The system proceeds in synchronous rounds, but an adversary can selectively make edges unavailable for message transmission. Formally:

Definition 1 (Adversarial Delay Function). *A delay function $d : \mathbb{N} \times V \times E \to \{0, 1\}$ specifies for each round j, node v, and incident edge $e = \{v, u\}$ whether transmission from v along e is blocked ($d(j, v, e) = 1$) or allowed ($d(j, v, e) = 0$).*

Definition 2 (Finite Delay Property). *A delay function d satisfies the finite delay property if for every node-edge pair (v, e), any sequence of consecutive rounds where $d(j, v, e) = 1$ is finite. Formally, for all $v \in V$ and incident edges e, if $d(j, v, e) = 1$ for $j \in [t_1, t_2]$, then $t_2 - t_1$ is finite.*

Key Model Assumption. Nodes are aware of which of their incident edges are currently blocked. This models scenarios where transmission failures are detectable (e.g., through carrier sensing, acknowledgments, or network management protocols).

3.2 The RAAF Protocol

Within the above model, we define the Round-Asynchronous Amnesiac Flooding protocol.

Node State. Each node v maintains:

– $M(j,v) \in \{0,1\}$: whether v possesses the message in round j
– $s(j,v) \subseteq V$: source set - neighbours that successfully delivered the message to v in round j
– $\text{dest}(j,v) \subseteq V$: destination set - neighbours to which v intends to forward the message

Structured Amnesia. The protocol is called "amnesiac" because:

– Nodes do not maintain history: i.e. nodes don't remember which nodes they received messages from (or sent messages to) in previous rounds
– The destination set is functionally determined by recent receptions and current delays
– When destination sets empty and no delayed transmissions remain, nodes return to their initial state

The key insight is that while nodes maintain destination sets across rounds (not strictly amnesiac), this state is *recomputed* based on current information rather than accumulated history. When a node receives the message from new sources, it completely recomputes its forwarding strategy.

Protocol Operation. Each round j proceeds as follows:

1. **Delay Phase**: The adversary specifies $d(j,v,e)$ for all node-edge pairs
2. **Transmission Phase**: Each node v with $M(j-1,v)=1$ attempts to transmit to all $u \in \text{dest}(j-1,v)$ where $d(j,v,\{v,u\})=0$
3. **Reception Phase**: Nodes receive messages from successful transmissions
4. **State Update Phase**: Nodes update their state according to the following rules:

State Update Rules. For node u transitioning from round j to $j+1$:
Message Possession:

$$M(j+1,u) = \begin{cases} 1 & \text{if } u \text{ has pending delayed transmissions from round } j \\ 1 & \text{if } u \text{ receives the message in round } j+1 \\ 0 & \text{otherwise} \end{cases}$$

Source Set:

$$s(j+1,u) = \{v \in V : \{v,u\} \in E, u \in \text{dest}(j,v), d(j+1,v,\{v,u\}) = 0\}$$

Destination Set: The update rule for destination sets captures the "structured amnesia":

$$\text{dest}(j{+}1,u) = \begin{cases} \{v \in \text{dest}(j,u) : d(j+1,u,\{u,v\}) = 1\} & \text{if continuing delayed transmission} \\ N(u) \setminus s(j+1,u) & \text{if newly receiving message} \\ \emptyset & \text{if no message possessed} \end{cases}$$

Observation 1 (Persistence of Destination Sets (PDS)) *Let $\{u,v\} \in E$. If u receives in round j from $w \neq v$ and does not receive from v in the same round, then v belongs to dest(u) until u delivers to v or receives from v.*

A key property of the protocol is that when u receives the message from new sources, it *recomputes* its destination set as all neighbours except those that just delivered the message, effectively "forgetting" its previous forwarding intentions.

3.3 System Evolution and Termination

State Function. The system state is captured by $S : \mathbb{N} \times V \to \text{StateRecord}$ where $S(j,u) = (M(j,u), s(j,u), \text{dest}(j,u))$.

Round Function. The transmission function $r(j)$ records actual message transmissions in round j:

$$r(j) = \{(v, \{v,w\}) : v \in V, w \in \text{dest}(j-1,v), d(j,v,\{v,w\}) = 0\}$$

State Update Function. The state update function defines how node states evolve. For node v in round j, computing the next state requires:

– Current graph state $S(j,\cdot) : V \to \text{StateRecord}$
– Current delay decisions $d(j,\cdot,\cdot) : V \times E \to \{0,1\}$
– Node's local state $S(j,v)$ and incident delays $d(j,v,\cdot)$

While updates depend on graph-wide state and delays, these parameters remain fixed when computing individual node updates in a given round. Thus, we can express the state update as:

$S(j+1,v) := u(v, S(j,v), d(j,v,\cdot))$

where u implicitly references the global state $S(j,\cdot)$ and delays $d(j,\cdot,\cdot)$ fixed for round j.

Initial Configuration. At round zero:

– Source: $S(0,g_0) = (1, \emptyset, N(g_0))$
– Others: $S(0,u) = (0,\emptyset,\emptyset)$ for $u \neq g_0$

Termination. We say that flooding has terminated "by" (at or before) round $t \in \mathbb{N}$ if $M(t,v) = 0$ for every $v \in V$, i.e. no node v has the message at round t. Clearly, this is equivalent with saying that $M(j,v) = 0$ for every $j \geq t$ and for every $v \in V$, i.e. if flooding has terminated by round t then no node has the message in any round after round t.

Definition 3. *The termination round t_{min} is defined as:*

$$t_{min} = \min\{t \in \mathbb{N} : M(k,u) = 0, \textit{ for every } u \in V\}$$

Observation 2 (Persistence of Empty Destination Sets (PEDS)) *If a node's destination set is empty at round t, it remains empty in all subsequent rounds until the node receives the message.*

Lemma 1 (Empty Destination Sets and Termination). *All destination sets are empty at round t if and only if flooding has terminated by round t.*

4 Termination Dichotomy

It is not hard to establish that RAAF always terminates on acyclic graphs, regardless of the adversarial strategy. In the remainder of this section we focus on graphs that contain at least one cycle, where we prove that every such graph admits a non-terminating strategy under RAAF by constructing a periodic infinite schedule (Definition 7)[3] that maintains message circulation within the cycle.

[3] When restricting schedule information only to the cycle.

Let $G = (V, E)$ be an arbitrary graph containing a cycle $C = (V_C, E_C)$ where $V_C = \{v_1, \ldots, v_n\}$ and $E_C = \{\{v_i, v_{i+1 \bmod n}\} : 1 \leq i \leq n\}$. Let c be the earliest round where a node in V_C receives the message. Among nodes receiving the message in round c, designate one as v_1 and number remaining cycle nodes sequentially[4]. We define the following delay function d:

$$d(j, u, e) = \begin{cases} 0 & \text{if } j - c \equiv i \pmod{n} \text{ and } e = \{v_i, v_{i+1 \bmod n}\} \\ 1 & \text{otherwise} \end{cases}$$

Property 1 (Cyclic Propagation Pattern). If, for every $i \in \mathbb{N}^+$, we have that node $v_{i \bmod n}$ of the cycle C transmits to node $v_{(i+1) \bmod n}$ of the cycle C at round round $c + i$, then we say that the *cyclic propagation pattern* is satisfied for cycle C.

Lemma 2. *The delay function d ensures cyclic propagation.*

Lemma 3 (Characterisation of Cyclic Pattern Disruption). *The cyclic propagation pattern (Property 1) is disrupted if and only if for some k, node $v_{k+1 \bmod n}$ transmits to $v_{k \bmod n}$ while $v_{k \bmod n}$ has the message but before $v_{k \bmod n}$ transmits to $v_{k+1 \bmod n}$.*

Lemma 4 (Non-disruption of Cyclic Pattern). *The cyclic propagation pattern cannot be disrupted by message flow in the reverse direction.*

Lemma 5 (External Message Preservation). *Receiving messages from nodes outside the cycle does not disrupt cyclic propagation.*

Theorem 1 (Cyclic Non-termination). *For any graph $G = (V, E)$ containing a cycle, there exists a valid delay function d such that flooding does not terminate.*

5 Periodic Infinite Schedules

We introduce *Periodic Infinite Schedules (PIS)* as a framework for analysing certain non-terminating behaviours in RAAF systems, serving as a bridge between finite state descriptions and infinite executions.

Definition 4 (Configuration). *A* configuration $\sigma(j)$ *for round j is an ordered pair $(S(j, \cdot), r(j))$ where S and r are the state and round functions, capturing complete system state and message transmissions.*

Definition 5 (Schedule). *A* schedule σ *maps each round $j \in \mathbb{N}$ to its configuration. Given a valid delay function d, the schedule σ_d* induced by d *is as follows:*

[4] We require that $v_2 \neq g_0$. This is because we assume that $v_2 \notin s(c, v_1)$, and this condition would not be satisfied if $v_2 = g_0$

1. $\sigma_d(0) = (S(0, \cdot), r(0))$
2. *For* $j > 0$, $\sigma_d(j) = (S(j, \cdot), r(j))$ *where:*
   – $S(j, v) = u(v, (S(j-1, v), d(j-1, v, \cdot)))$
   – $r(j) = \{(v, \{v, w\}) \in V \times E : M(j-1, v) = 1 \wedge w \in \mathit{dest}(j-1, v) \wedge d(j, v, \{v, w\}) = 0\}$

In the above definition, the schedule is "induced" as states and transmissions arise deterministically from applying the delay function according to our state evolution rules.

Definition 6 (Eventually Periodic Delay Function). *A* delay function d *is eventually periodic with period* p *if, for some* $c \in \mathbb{N}$*, we have that* $d(i, v, \{v, u\}) = d(i + p, v, \{v, u\})$*, for every round* $i \geq c$ *and for every node* $v \in V$ *and every edge* $\{v, u\}$*.*

Definition 7 (Periodic Infinite Schedule). *A schedule* σ *is* periodic infinite *if there exist natural numbers* c *(stabilisation round) and* l *(cycle length) where:*

1. $\sigma(j) = \sigma(j + l)$ *for every* $j \geq c$*,*
2. $r(c + k) \neq \emptyset$ *for at least one* $k \in \{0, 1, \ldots, l-1\}$*.*

In the above definition, the first condition establishes repeating behaviour after the stabilisation round c, while the second guarantees genuine non-termination through guaranteed transmissions. Now we establish three fundamental results characterising PIS behaviour:

Theorem 2 (IPIS: Identification of PIS). *Given a graph* G *and delay function* d*, the induced schedule* σ_d *is periodic infinite if:*

1. d *is eventually periodic (Definition 6) with period* p*,*
2. $\exists c, l \in \mathbb{N}_{>0} : \forall u \in V : S(c, u) = S(c + l, u)$*,*
3. $l \bmod p = 0$*,*
4. $\exists j \in \{0, \ldots, l-1\} : r(c + j) \neq \emptyset$*.*

Theorem 3 (NTPIS: Non-Termination of PIS). *Any periodic infinite schedule is non-terminating.*

Theorem 4 (EPIS: Existence of PIS). *If a graph admits a non-terminating schedule, it admits a periodic infinite schedule.*

Proof. Suppose that d *induces a non-terminating schedule* σ_d *on* G*. The configuration space is finite, as it has at most* $(2 \cdot 2^{|V|} \cdot 2^{|V|})^{|V|} \cdot 2^{2|E|}$ *configurations. Therefore, as* σ_d *is a non-terminating schedule, there exists at least one configuration* C *which repeats infinitely often.*

As the adversary respects the finite delay property, it follows that for every pair $(v, \{v, u\})$ *there exists an infinite sequence of rounds, in which* $(v, \{v, u\})$ *is not delayed. Let* j_1 *be the first round in* σ_d *where configuration* C *appears. Due to the finite delay property, there exists some round* $j_2 > j_1$ *such that (i) the configuration* C *appears also at round* j_2 *and (ii) every pair* $(v, \{v, u\})$

was allowed to transmit (i.e. it was not delayed by the adversary) at least once between rounds j_1 and j_2.

We now define a new delay function d', which is periodic with period $j_2 - j_1$ after round j_2, as follows:

- *if $j \leq j_2$ then $d'(j, u, e) = d(j, u, e)$, for every (u, e),*
- *if $j > j_2$ then $d'(j, u, e) = d'(j - j_2 + j_1), u, e)$, for every (u, e).*

Then the schedule $\sigma_{d'}$ induced by this new delay function d' is periodic after round j_2, with period $j_2 - j_1$.

These results establish that PIS capture the fundamental structure of non-termination in RAAF systems. While not all non-terminating schedules are periodic, any graph admitting non-termination must also admit a periodic infinite schedule. This insight reduces termination analysis to the study of periodic behaviours, bridging finite state descriptions and infinite executions.

6 Undecidability with Arbitrary Computable Adversaries

We prove that determining flooding termination in RAAF is undecidable when the adversary is an arbitrary computable function via reduction from the Halting problem [9]. The decision problem for the termination of RAAF is defined as follows:

RAAF-TERMINATION

- **Input:** A graph $G = (V, E)$, source node $g_0 \in V$, and computable delay function d.
- **Question:** Does flooding terminate on G with source g_0 under adversary d?

We first establish a basic non-terminating strategy on the triangle graph $G = (V, E)$ where:

- $V = \{\text{Source}, A, B\}$
- $E = \{\{\text{Source}, A\}, \{A, B\}, \{B, \text{Source}\}\}$

We now define the basic delay function d_0:

$$d_0(j, u, e) = \begin{cases} 1 & \text{if } j \bmod 3 = 0 \text{ and } e \neq \{\text{Source}, B\} \\ 1 & \text{if } j \bmod 3 = 1 \text{ and } e \neq \{\text{Source}, A\} \\ 1 & \text{if } j \bmod 3 = 2 \text{ and } e \neq \{A, B\} \\ 0 & \text{otherwise} \end{cases}$$

Lemma 6 (Basic Strategy Nonterminating). *The basic delay strategy d_0 creates a periodic infinite schedule.*

Proof. We demonstrate the basic strategy induces a periodic infinite schedule by verifying the conditions of Theorem 2:

1. *The delay function d is eventually periodic with period $p = 3$. For all $i \geq 1$, $u \in V$, $e \in E$: $d(i, u, e) = d(i + 3, u, e)$. This follows from the three-round delay pattern in the strategy definition.*
2. *Taking $c = 3$ and $l = 6$, we observe from the state evolution in Table 1 that $S(3, u) = S(9, u)$ for all $u \in V$. The complete state evolution demonstrating this equality is shown below:*

Table 1. State Evolution under Basic Strategy

Round 0	**Round 1**	**Round 2**	**Round 3**	**Round 4**
$S : (1, \emptyset, \{A, B\})$	$S : (1, \emptyset, \{B\})$	$S : (1, \emptyset, \{B\})$	$S : (1, \{B\}, \{A\})$	$S : (0, \emptyset, \emptyset)$
$A : (0, \emptyset, \emptyset)$	$A : (1, \{S\}, \{B\})$	$A : (0, \emptyset, \emptyset)$	$A : (0, \emptyset, \emptyset)$	$A : (1, \{S\}, \{B\})$
$B : (0, \emptyset, \emptyset)$	$B : (0, \emptyset, \emptyset)$	$B : (1, \{A\}, \{S\})$	$B : (1, \{S\}, \{A\})$	$B : (1, \emptyset, \{A\})$
Round 5	**Round 6**	**Round 7**	**Round 8**	**Round 9**
$S : (0, \emptyset, \emptyset)$	$S : (1, \{B\}, \{A\})$	$S : (1, \{A\}, \{B\})$	$S : (1, \emptyset, \{B\})$	$S : (1, \{B\}, \{A\})$
$A : (1, \{B\}, \{S\})$	$A : (1, \emptyset, \{S\})$	$A : (1, \{S\}, \{B\})$	$A : (0, \emptyset, \emptyset)$	$A : (0, \emptyset, \emptyset)$
$B : (1, \{A\}, \{S\})$	$B : (0, \emptyset, \emptyset)$	$B : (0, \emptyset, \emptyset)$	$B : (1, \{A\}, \{S\})$	$B : (1, \{S\}, \{A\})$

3. *$l \bmod p = 0$ as $6 \bmod 3 = 0$.*
4. *In each cycle $[3, 9)$, transmission occurs. For instance, at round 4, we have $r(4) \neq \emptyset$.*

Therefore, by Theorem 2, the schedule is periodic infinite with $c = 3$ and $l = 6$. By Theorem 3, we conclude it is non-terminating.

Theorem 5 (Undecidability). *RAAF-TERMINATION is undecidable.*

Proof. Let M be an arbitrary Turing Machine M, and let x be an arbitrary input to M. Then we construct the following delay function on the triangle graph $G = (V, E)$:

$$d(j, u, e) = \begin{cases} 0 & \text{if } M \text{ halts on } x \text{ within } j \text{ steps} \\ d_0(j, u, e) & \text{otherwise} \end{cases}$$

where d_0 is the basic delay function defined above. We will prove that M halts on x if and only if flooding terminates under the delay function d.

(⇒) Suppose that M halts on x after exactly t steps. Then $d(j, u, e) = 0$ for every $j \geq t$ and every u and e. Then flooding terminates by round $t + 2$.

(⇐) Suppose that flooding terminates at round t, and assume for the sake of contradiction that M does not halt after any finite number of steps. Then $d(j, u, e) = d_0(j, u, e)$ for every j and every u and e, and thus d creates a periodic infinite schedule by Lemma 6. Therefore flooding does not terminate after any finite number of rounds, which is a contradiction.

This undecidability persists even under severe computational constraints:

Theorem 6 (Resilient Undecidability). *For any unbounded computable $f : \mathbb{N} \to \mathbb{N}$ with $\lim_{n \to \infty} f(n) = \infty$, an adversary with $O(f(n))$ time and space in round n can simulate $\lfloor f(n) \rfloor$ Turing machine steps, preserving undecidability.*

Proof. *Let the adversary in round n simulate $\lfloor f(n) \rfloor$ Turing machine steps, applying no delays if halted, else d_0. Then the Turing machine halts after k steps if and only if flooding terminates after $f^{-1}(k)$ rounds. The theorem holds even for extremely slow-growing f like inverse Ackermann $\alpha(n) = \min\{m : A(m,m) > n\}$ where A is the Ackermann function [10].*

This fundamental limitation motivates restricting adversary behaviour rather than computational power, leading to the EPA model.

7 Eventually Periodic Adversaries

To bridge the gap between undecidability and practical analysis, we introduce Eventually Periodic Adversaries (EPA), which restrict adversaries to eventually periodic behaviour while maintaining significant expressive power.

Definition 8 (Eventually Periodic Adversary). *An* Eventually Periodic Adversary *is a triple (d, c, l) where the delay function d is a computable function, $c \in \mathbb{N}$ is the stabilisation round, $l \in \mathbb{N}^+$ is the cycle length, and $d(i, u, e) = d(i + l, u, e)$ for every $i \geq c$, $u \in V$, and $e \in E$.*

Theorem 7 (EPA Decidability). *The termination problem for EPA-RAAF systems is decidable with time complexity $O(2^{2|V|^2+2|E|}(c + l))$.*

Theorem 8 (Lower Bound). *Any decision procedure for EPA-RAAF termination has time complexity $\Omega((c + l) \cdot |E|)$.*

The EPA model demonstrates that restricting adversary behaviour to eventual periodicity yields decidable termination while preserving significant expressive power, bridging the gap between undecidability for arbitrary computable adversaries and practical analysis needs. A non-trivial lower bound remains open.

8 Conclusions and Open Problems

We have established fundamental properties of Round-Asynchronous Amnesiac Flooding through rigorous mathematical characterisation. Our analysis yields a complete structural dichotomy: acyclic graphs guarantee termination with $O((B + 1) \cdot e(g_0))$ bound for B-bounded delays, while cyclic graphs admit non-terminating adversarial strategies. This extends to a sharp complexity separation - termination is undecidable for arbitrary computable adversaries, but it becomes decidable for eventually periodic adversaries with time complexity at least $\Omega((c + l)|E|)$ and at most $O(2^{2|V|^2+2|E|}(c + l))$.

Several theoretical challenges remain open. The complexity landscape invites tighter bounds for EPA-RAAF, while specific graph classes may admit improved parameterisation by structural properties. Natural model extensions include multiple messages and dynamic topologies with bounded modification rates.

Our dichotomy results establish fundamental limits on automated verification of asynchronous flooding protocols: while acyclic networks allow termination analysis, verification becomes undecidable in the presence of cycles unless the adversary exhibits eventual periodicity. This framework precisely characterises when efficient algorithmic analysis of asynchronous flooding behaviour is possible.

References

1. Adamek, J., Nesterenko, M., Robinson, J.S., Tixeuil, S.: Stateless Reliable Geocasting. In: Proceedings of SRDS 2017. IEEE Computer Society, Hong Kong, China (Sep 2017), `https://hal.sorbonne-universite.fr/hal-01549915`
2. Aspnes, J.: Flooding. Online (February 2019), `http://www.cs.yale.edu/homes/aspnes/pinewiki/Flooding.html`
3. Attiya, H., Welch, J.: Distributed Computing: Fundamentals, Simulations and Advanced Topics. John Wiley & Sons (2004)
4. Bayramzadeh, Z., Kshemkalyani, A.D., Molla, A.R., Sharma, G.: Weak amnesiac flooding of multiple messages. In: Proceedings of the 9th International Conference on Networked Systems (NETYS). vol. 12879, pp. 1–17 (2021)
5. Dwork, C., Lynch, N.A.: Consensus in the presence of partial synchrony. Journal of the ACM **35**(2), 288–323 (1988)
6. Hussak, W., Trehan, A.: On the termination of a flooding process. arXiv preprint arXiv:1907.07078 (2019), `https://arxiv.org/abs/1907.07078`
7. Hussak, W., Trehan, A.: Terminating cases of flooding. arXiv preprint arXiv:2009.05776 [cs.DC] (September 2020), `https://arxiv.org/abs/2009.05776`
8. Hussak, W., Trehan, A.: Termination of amnesiac flooding. Distributed Computing **36**(2), 193–207 (May 2023). `https://doi.org/10.1007/s00446-023-00448-y`
9. Lucas, S.: The origins of the halting problem. Journal of Logical and Algebraic Methods in Programming **121**, 100687 (June 2021)
10. Matos, A.B.: Total recursive functions that are not primitive recursive. Unpublished manuscript (June 21 2016), `https://www.dcc.fc.up.pt/~acm/definitions.pdf`, accessed: Sep. 18, 2024
11. Turau, V.: Analysis of amnesiac flooding. CoRR **abs/2002.10752** (2020), `https://arxiv.org/abs/2002.10752`