Memory Lower Bounds and Impossibility Results for Anonymous Dynamic Broadcast

Garrett Parzych
School of Computing and Augmented Intelligence, Biodesign Center for Biocomputing, Security and Society Arizona State University, Tempe, AZ, USA

Joshua J. Daymude
School of Computing and Augmented Intelligence, Biodesign Center for Biocomputing, Security and Society Arizona State University, Tempe, AZ, USA

Abstract

Broadcast is a ubiquitous distributed computing problem that underpins many other system tasks. In static, connected networks, it was recently shown that broadcast is solvable without any node memory and only constant-size messages in worst-case asymptotically optimal time (Hussak and Trehan, PODC'19/STACS'20/DC'23). In the dynamic setting of adversarial topology changes, however, existing algorithms rely on identifiers, port labels, or polynomial memory to solve broadcast and compute functions over node inputs. We investigate *space-efficient, terminating broadcast algorithms for anonymous, synchronous, 1-interval connected dynamic networks* and introduce the first memory lower bounds in this setting. Specifically, we prove that broadcast with *termination detection* is impossible for idle-start algorithms (where only the broadcaster can initially send messages) and otherwise requires $\Omega(\log n)$ memory per node, where n is the number of nodes in the network. Even if the termination condition is relaxed to *stabilizing termination* (eventually no additional messages are sent), we show that any idle-start algorithm must use $\omega(1)$ memory per node, separating the static and dynamic settings for anonymous broadcast. This lower bound is not far from optimal, as we present an algorithm that solves broadcast with stabilizing termination using $\mathcal{O}(\log n)$ memory per node in worst-case asymptotically optimal time. In sum, these results reveal the necessity of non-constant memory for nontrivial terminating computation in anonymous dynamic networks.

2012 ACM Subject Classification Theory of computation → Distributed algorithms; Networks → Network algorithms

Keywords and phrases Dynamic networks, anonymity, broadcast, space complexity, lower bounds, termination detection, stabilizing termination

Digital Object Identifier 10.4230/LIPIcs.DISC.2024.35

Funding This work is supported in part by National Science Foundation award CCF-2312537.

1 Introduction

Distributed algorithms for *dynamic networks* enable processes to coordinate even as the communication links between them change over time, often rapidly and adversarially [1, 8]. When once a process disconnecting from a distributed system was viewed as a rare crash fault to tolerate, research over the last two decades has come to view dynamics as natural or even necessary to a system's function. Application domains such as self-stabilizing overlay networks [2, 19], blockchains [4, 21], and swarm robotics [20, 22] are defined by their rapidly changing or physically moving components, forcing algorithms to achieve their goals by leveraging – or more often operating in spite of – these dynamics.

Despite the challenges, many fundamental problems have been addressed under adversarial dynamics, including broadcast, consensus, and leader election (see [3, 5] for complementary surveys). However, many of these algorithms endow their nodes with unique identifiers, port labels for locally distinguishing among their neighbors, (approximate) knowledge of

© Garrett Parzych and Joshua J. Daymude;
licensed under Creative Commons License CC-BY 4.0
38th International Symposium on Distributed Computing (DISC 2024).
Editor: Dan Alistarh; Article No. 35; pp. 35:1–35:18
Leibniz International Proceedings in Informatics
Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

the number of nodes in the network, or superlinear memories. Taking inspiration from collective behavior in biological complex systems of computationally weak entities – such as foraging in ant colonies [10, 32], aggregation in slime mold spore migrations [36], and energy distribution in microbiomes [28, 33] – we question to what extent these additional capabilities are necessary. Specifically, we consider dynamic networks of *anonymous* nodes (lacking both unique identifiers and port labels) with limited memory. As an aside, anonymity is a desirable feature in its own right for engineering privacy-sensitive applications, such as Bluetooth-based contact tracing [35].

We consider the fundamental problem of synchronous *broadcast*, in which all nodes in a network must eventually be informed of some information originating at a single node. In static, connected networks, broadcast can be solved without any persistent memory at all: the *amnesiac flooding* algorithm [23–25] – in which nodes forward copies of any message they receive to any neighbor that did not send them the message in the last round – informs all nodes (correctness) and allows them to eventually stop sending additional messages (stabilizing termination) within worst-case asymptotically optimal time. But even when adversarial dynamics are constrained to maintain network connectivity in every round, a longstanding conjecture states that dynamic broadcast with stabilizing termination is impossible in $\mathcal{O}(\log n)$ memory without node identifiers or knowledge of n [31], let alone without any memory at all. We summarize our contributions as follows.

Our Contributions. All results are proven with respect to deterministic algorithms run by anonymous nodes (lacking identifiers and port labels) in a dynamic network whose topology can change arbitrarily but remains connected in each synchronous round. In this setting:

- Broadcast with *termination detection* – i.e., the broadcaster must eventually decide broadcast is complete – is impossible for idle-start algorithms where only the broadcaster can initially send messages (Section 2) and otherwise requires $\Omega(\log n)$ space (Section 3).
- Any idle-start algorithm solving broadcast with *stabilizing termination* – i.e., eventually no additional messages are sent – must have $\omega(1)$ space complexity (Section 4). We then present an algorithm solving broadcast with stabilizing termination in $\mathcal{O}(\log n)$ space and worst-case asymptotically optimal time (Section 5).

We note that although synchronous systems are typically seen as less general than asynchronous ones, it is actually the opposite for the purposes of lower bounds. Since an asynchronous adversary can always simulate a synchronous one, any impossibility results or lower bounds proven w.r.t. synchrony will apply to both types of systems.

1.1 Model

Dynamic Networks. We consider a synchronous dynamic network comprising a fixed set of nodes V. Nodes communicate with each other via message passing over a communication graph whose topology changes over time. We model this topology as a *time-varying graph* $\mathcal{G} = (V, E, T, \rho)$ where V is the set of nodes, E is the static set of undirected edges that may appear in the graph, $T = \mathbb{N}$ is the lifetime of the graph, and $\rho : E \times T \to \{0, 1\}$ is the presence function indicating whether an edge exists at a given time [8]. We refer to the set of edges present at time $t \in T$ as $E_t = \{e \in E : \rho(e, t) = 1\}$ and the undirected graph $G_t = (V, E_t)$ as the *snapshot* of $\mathcal{G}$ at time $t \in T$. We assume an adversary controls the presence function ρ and that E is the complete set of edges on V; i.e., we do not limit which edges the adversary can introduce. We do, however, follow the majority of dynamic broadcast literature (e.g., [8, 12, 26, 27, 31]) in assuming *1-interval-connectivity* (also called "always-connected snapshots"); i.e., the adversary may make arbitrary topological changes at each time $t \in T$ so long as each snapshot G_t is connected.

Node Capabilities. Motivated by computationally weak individuals in biological collectives (e.g., cells, microbes, social insects, etc.), we consider nodes that are *anonymous*, lacking unique identifiers, and have *no knowledge or approximation of any global measure*, including the number of nodes n. We further assume that nodes have *no port labels*; i.e., they cannot count or locally distinguish among their neighbors. Consequently, when a node communicates with its neighbors via *message passing*, it does so using a broadcast mechanism, sending the same message to all its current neighbors.

Algorithms and Execution. Each node in the time-varying graph $\mathcal{G}$ synchronously executes the same distributed algorithm $\mathcal{A}$. All nodes are initialized at time $t = 0$, and each synchronous round t starting at time t proceeds as follows:

1. The adversary fixes the network topology G_t for round t.
2. Each node may send a message to its neighbors in G_t according to algorithm $\mathcal{A}$ as a function of its current state.
3. Each node may perform a state transition according to algorithm $\mathcal{A}$ as a function of its current state and the multiset of messages it (reliably) receives from its neighbors in G_t.

Memory. In this paper, we are primarily concerned with an algorithm's *space complexity*, the maximum number of bits a node uses to store its state between rounds. As usual for distributed systems, we are interested in the asymptotic growth of this measure as a function of $n = |V|$, the number of nodes. We emphasize that even if nodes have $\Omega(\log n)$ memory – sufficient for storing unique identifiers – they are anonymous and are not assigned such identifiers a priori. We also note that we do not analyze *message complexity* directly, as our execution model specifies nodes that send messages based only on their states; thus, there are at most as many message types as states.

Broadcast. In the *broadcast* problem, every node starts in the same state except for a single node known as the *broadcaster* that is trying to deliver some information to every other node in the network. We say that a node is *informed* if it is the broadcaster or has previously received a message from an informed node. All other nodes are *uninformed*. A broadcast is *complete* when every node in the network is informed. A distributed algorithm $\mathcal{A}$ *solves* the broadcast problem in t rounds if, for any time-varying graph whose nodes all execute $\mathcal{A}$, broadcast is completed by the end of round t.

Idle-Start. A node is *idle* if it will not send a message in the subsequent round, and will not change its state if it does not receive a message (this has also been called "quiescent" [29], though that term is overloaded in this context). Some broadcast algorithms critically rely on initializing all nodes as non-idle at time $t = 0$; conversely, an *idle-start* algorithm initializes all nodes except the broadcaster as idle.

Termination. A simple solution to broadcast is to make every informed node continuously send messages. Since we assume 1-interval connectivity, there is always at least one uninformed node receiving a message from an informed node in each round, so this algorithm solves broadcast in $\mathcal{O}(n)$ rounds. This runtime bound is worst-case asymptotically optimal, but a smoothed analysis reveals significant improvements on more "typical" topologies [17, 18]. This algorithm also achieves $\Theta(1)$ space complexity, since nodes need only remember whether they're informed. However, nodes sending messages forever creates undesirable congestion and precludes the system from advancing to further tasks, e.g., starting a new broadcast

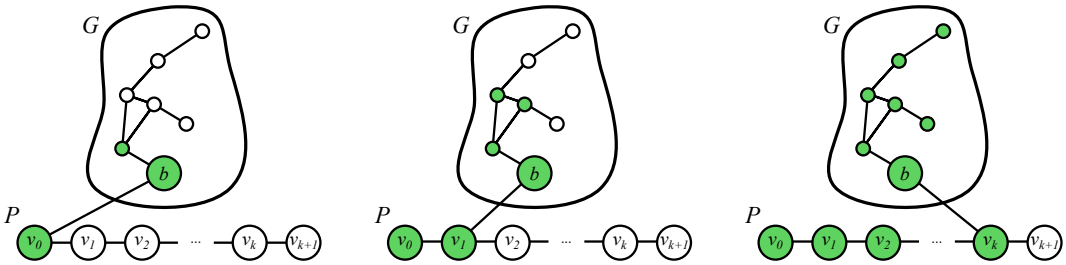
or using the broadcast information as part of a larger algorithm. Thus, we seek algorithms meeting some kind of termination conditions. An algorithm achieves *stabilizing termination* if every node becomes idle within finite time. An algorithm achieves the stronger condition of *termination detection* if the broadcaster correctly and irrevocably decides that broadcast is complete (i.e., by entering a terminating state) within finite time.

1.2 Related Work

Broadcast is a ubiquitous and well-studied distributed computing problem, often appearing as a building block in more complex tasks. In static, connected networks, broadcast with stabilizing termination is solvable without any node memory (and thus without identifiers) and only $\Theta(1)$ message complexity in worst-case asymptotically optimal time, though port labels are required to distinguish among neighbors [23, 24]. In a recent extension of this work, the same algorithm was proven correct under node and edge deletion dynamics [25], but breaks down under more general adversarial dynamics. With this inspiration, our focus is space-efficient, terminating algorithms for broadcast in anonymous dynamic networks.

Early works on dynamic broadcast typically assumed stronger node capabilities. A series of works on shortest, fastest, and foremost broadcast assumed local identifiers enabling a node u to maintain a consistent label for any neighbor v, even if v disconnected from and later reconnected to u [6–8]. This assumption enables the construction of time-invariant logical structures like spanning trees, which – when combined with the assumption of recurrent dynamics (edges will eventually reappear) – reduces dynamic broadcast to static routing on these structures. Similar techniques are used when assuming both unique node identifiers and shared knowledge of n, the number of nodes in the network [34]. Among these early works, only O'Dell and Wattenhofer [31] share our focus on anonymous nodes and space complexity. They conjectured that no algorithm can solve broadcast with stabilizing termination in $\mathcal{O}(\log n)$ space when nodes are anonymous and have no knowledge of n. Interestingly, we prove this is solvable in our model (Section 5), though we do not disprove their conjecture since our model's synchrony is not directly comparable to their combination of asynchronous time, bounded message latency, and disconnection detection for re-broadcasting messages.

A parallel line of work investigated what functions a dynamic network can deterministically compute over its nodes' inputs [27]. In the context of anonymous dynamic networks, most results focus on the *(exact) counting problem* [9, 11, 14, 15, 26, 30], which terminating broadcast reduces to in $\mathcal{O}(n)$ time and $\mathcal{O}(\log n)$ space: once the broadcaster knows the number of nodes n, it need only wait n rounds before every other node must have been informed (since the dynamic network is 1-interval connected), at which point it can terminate. These works recently culminated in the exact characterization by Di Luna and Viglietta [12, 13] showing that anonymous dynamic networks with at least one leader can compute only the *multi-aggregate functions* – those for which a node's output depends only on its own input and the multiset of all nodes' inputs – and do so in optimal (linear) time. However, their algorithm uses $\Theta(n^3 \log n)$ space in the worst case [16], leaving open what memory is necessary. Our impossibility results and memory lower bounds for terminating broadcast (Sections 2–4) shed light on this question, as many nontrivial multi-aggregate functions require information from at least one node to be communicated to all other nodes (e.g., minimums/maximums, averages, exact and generalized counting, etc.).



(a) End of Round 0. **(b)** End of Round 1. **(c)** End of Round k.

Figure 1 The time-varying graph $\mathcal{G}$ used in the proof of Theorem 1. In each round t, the broadcaster b is connected to node v_t in the path P. Informed nodes are shown in green. When b declares broadcast to be complete in round k, node v_{k+1} is still uninformed.

2 Impossibility Results for Termination Detection

We begin by proving that there is no idle-start algorithm – i.e., one in which only the broadcaster can initially send messages – that solves broadcast with termination detection in our setting. The idea behind the proof is as follows. Supposing to the contrary that such an algorithm $\mathcal{A}$ exists, it must solve broadcast and detect termination on any static, connected network G. So we consider an execution of $\mathcal{A}$ on an extension of G as a time-varying graph $\mathcal{G}$ that we carefully construct to achieve two goals: (1) the execution of $\mathcal{A}$ on G in the time-varying graph $\mathcal{G}$ is identical to its execution on G alone, and (2) there is an idle node in $\mathcal{G}$ that is sequestered from ever being informed. This drives a contradiction: the broadcaster must detect termination in $\mathcal{G}$ because it does so on G alone, but will do so incorrectly because there is still an uninformed node.

▶ **Theorem 1.** *No deterministic idle-start algorithm can solve broadcast with termination detection for anonymous, synchronous, 1-interval connected dynamic networks.*

Proof. Suppose to the contrary that there exists an idle-start algorithm $\mathcal{A}$ solving broadcast with termination detection in our setting. Let G be any static, connected graph and let b be any node in G. If $\mathcal{A}$ is executed on G with b as the broadcaster, there must exist some round k during which b correctly and irrevocably declares broadcast to be complete.

Construct a time-varying graph $\mathcal{G}$ as follows (see Figure 1). First, $\mathcal{G}$ contains the static graph G as a fixed part; i.e., all edges of G will remain present throughout the lifetime of $\mathcal{G}$. Additionally, $\mathcal{G}$ contains a fixed path $P = v_0 v_1 \cdots v_{k+1}$ of $k+2$ nodes. Finally, in each round $t \in \{0, \ldots, k\}$, there is a single edge $\{b, v_t\}$ connecting the broadcaster b to the path P.

Consider the execution of $\mathcal{A}$ on $\mathcal{G}$ with b as the broadcaster. We argue by induction on $t \in \{0, \ldots, k\}$ that in round t, (1) all nodes in G send the same messages and perform the same state transitions as they did in the execution of $\mathcal{A}$ on G, and (2) the nodes $\{v_{t+1}, \ldots, v_{k+1}\}$ remain idle and uninformed. In round $t = 0$, only the broadcaster b potentially sends messages and changes state since $\mathcal{A}$ is an idle-start algorithm. Recall that nodes have no port labels and no knowledge of n. Thus, b must send the same messages and perform the same state transition as in the execution of $\mathcal{A}$ on G. The only neighbor of b in the path P is v_0, so nodes $\{v_1, \ldots, v_{k+1}\}$ receive no messages and remain idle and uninformed.

Now suppose the claim holds up to and including some round $0 \leq t < k$. By the induction hypothesis, all nodes in G have the same states at the start of round $t+1$ as in the execution of $\mathcal{A}$ on G, and thus send the same messages. However, since $\{b, v_{t+1}\}$ is the only edge between G and P in round $t+1$, b could in principle make a divergent state transition if it

receives a message from v_{t+1}. But by the induction hypothesis, v_{t+1} is idle and thus sends no messages in round $t+1$. For this same reason, nodes $\{v_{t+2}, \ldots, v_{k+1}\}$ receive no messages and remain idle and uninformed in round $t+1$.

Therefore, in round k, the broadcaster b irrevocably declares broadcast to be complete just as it did in the static setting, but v_{k+1} remains uninformed, a contradiction. ◀

A related line of research investigates the computability of functions in dynamic networks with no or multiple leaders. Di Luna and Viglietta have recently shown that if the number of leaders is known, nodes can compute the same functions as systems with a single leader. However, they found that if the number of leaders is unknown, then it is impossible for nodes to compute the size of the network with termination detection [13]. Using similar ideas to our previous proof, we can show that broadcast with termination detection is also impossible without knowing the number of broadcasters. Since broadcast can be reduced to counting the size of the network (simply broadcast for n rounds after receiving the count), our theorem implies the result of Di Luna and Viglietta. However, since we are unaware of any reduction from counting to broadcast, ours seems to be slightly more general.

▶ **Theorem 2.** *Even without an idle start, no deterministic algorithm can solve broadcast with termination detection for anonymous, synchronous, 1-interval connected dynamic networks if nodes have no knowledge of the number of broadcasters.*

Proof. Suppose for contradiction that an algorithm $\mathcal{A}$ solves broadcast with termination detection without giving nodes knowledge of the number of broadcasters. Let the *configuration* of a dynamic network at time t be the multiset of node states at the start of round t. Let $C_0, C_1, \ldots, C_x$ be the sequence of configurations that occur from running $\mathcal{A}$ on the complete graph K_3 with a single broadcaster, where C_x is the first configuration in which the broadcaster declares termination. We will create a new time-varying graph $\mathcal{G}$ on which $\mathcal{A}$ will incorrectly terminate. First create 2^x copies of C_0, with each copy having a single broadcaster and two non-broadcasters. Add a path $p_0 p_1 \ldots p_x$ of $x+1$ nodes and for each copy of C_0, choose one of the non-broadcaster nodes and attach it to p_0; this will be the first snapshot of $\mathcal{G}$.

Suppose that in C_1, the broadcaster was in state β_1 and the non-broadcasters in state α_1. By symmetry of the graphs, after the first round of executing $\mathcal{A}$ on $\mathcal{G}$, each copy of C_0 will have a node in state β_1 and a node in state α_1, while the node attached to p_0 will be in some other unknown state. Thus there are 2^x nodes in state β_1 and 2^x nodes in state α_1. Use all the nodes in state α_1 and half the nodes in state β_1 to create 2^{x-1} copies of the configuration C_1. Attach all of the unused nodes from the copies of C_0 to p_1. Again, choose a single non-broadcaster node from each copy of C_1, and attach them to p_1. Then run $\mathcal{A}$ for one additional round. If β_2 and α_2 are the states of the broadcaster and non-broadcasters in C_2 respectively, then by symmetry, our graph after this round will have 2^{x-1} copies each of β_2 and α_2. We can now use these nodes to create 2^{x-2} copies of C_2.

Continuing on in this way, we can create 2^{x-i} copies of configuration C_i for each $i \in \{0, \ldots, x\}$ while only informing a single node in the path at a time. Thus, after x rounds, we will have $2^{x-x} = 1$ copy of C_x and p_x will still be uninformed. But the broadcaster in the copy of C_x will have declared termination, contradicting the correctness of $\mathcal{A}$. ◀

3 Memory Lower Bound for Termination Detection

In an idle-start algorithm, nodes have no indication of whether they have idle, uninformed neighbors. This drives the indistinguishability result at the center of Theorem 1: a static, connected graph cannot tell if it's the entire network or a subgraph in a larger whole. Without

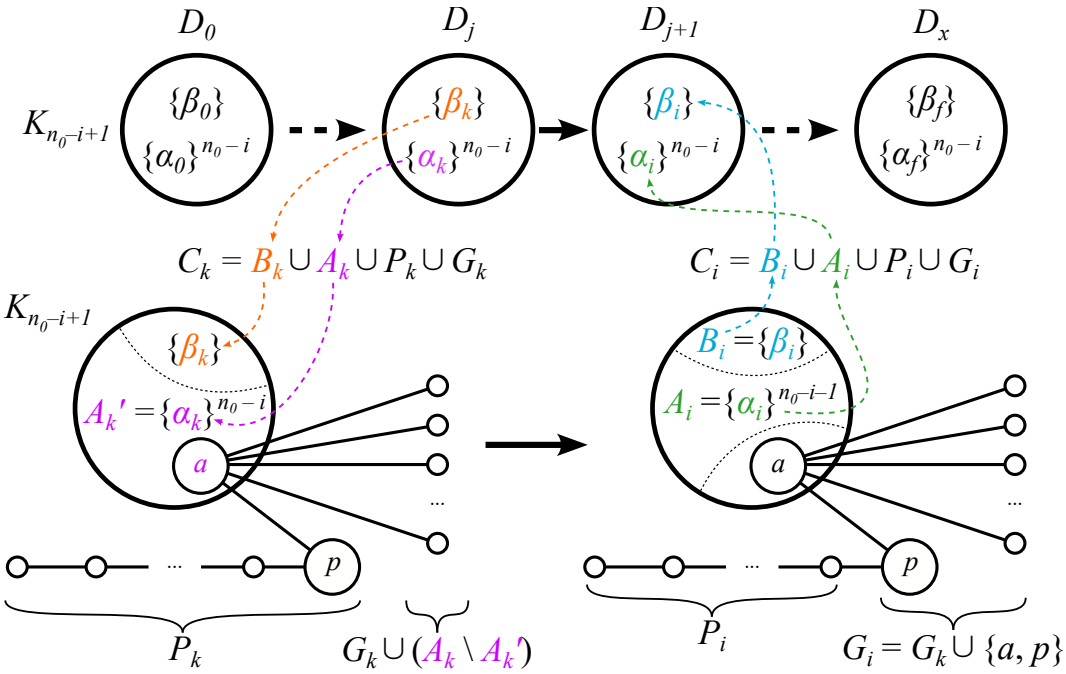


Figure 2 The inductive construction of configuration C_i as described in the proof of Theorem 3. The key idea is to find a new pair of states (β_i, α_i) that is not already in $(\beta_0, \alpha_0), \ldots, (\beta_{i-1}, \alpha_{i-1})$ by arranging a previously identified reachable configuration C_k and extending the corresponding execution of $\mathcal{A}$ by one additional round.

the constraints of an idle-start, however, this particular contradiction – and its corresponding impossibility result – disappears. For example, the history tree algorithm of Di Luna and Viglietta solves broadcast in this setting in linear time and $\Theta(n^3 \log n)$ space [16]. Still, all such algorithms must use at least logarithmic memory, as we now show.

▶ **Theorem 3.** *Any algorithm that solves broadcast with termination detection for anonymous, synchronous, 1-interval connected dynamic networks must have $\Omega(\log n)$ space complexity.*

Proof. Consider any (non-idle-start) algorithm $\mathcal{A}$ that solves broadcast with termination detection. Let $f(n)$ be the maximum number of states that $\mathcal{A}$ uses when run on dynamic networks of at most n nodes. We will show that $f(n) \geq n^{1/2}$ for all $n \geq 1$, implying that $\mathcal{A}$ uses $\log(f(n)) \geq \log(n^{1/2}) = \Omega(\log n)$ space.

Suppose to the contrary that there exists an $n_0 \geq 1$ such that $f(n_0) < n_0^{1/2}$. A configuration C is *reachable* (from an initial configuration) if there exists a time-varying graph $\mathcal{G}$ and time t such that the execution of $\mathcal{A}$ on $\mathcal{G}$ for t rounds results in configuration C. We will find a sequence of n_0 reachable configurations $(C_i)_{i=0}^{n_0-1}$ where each C_i is a disjoint union of multisets[1] of states $B_i \cup A_i \cup P_i \cup G_i$ satisfying:

1. $|C_i| = 2n_0$,
2. $B_i = \{\beta_i\}$, where β_i is the state of the broadcaster,
3. $A_i = \{\alpha_i\}^{n_0-i-1}$, exactly $n_0 - i - 1$ copies of the same state α_i,

[1] A multiset X is a disjoint union of multisets $Y \cup Z$ if the multiplicity of any element $x \in X$ is the sum of multiplicities of x in Y and Z.

4. $|P_i| \geq n_0 - i$ and contains only uninformed states,
5. $G_i = C_i \setminus (B_i \cup A_i \cup P_i)$, and
6. $(\beta_i, \alpha_i) \neq (\beta_j, \alpha_j)$ for all $j \neq i$.

Initially, the broadcaster is in some state β_0 and all other nodes are uninformed in some state $\alpha_0 \neq \beta_0$. Define the initial configuration C_0 by letting $B_0 = \{\beta_0\}$, $A_0 = \{\alpha_0\}^{n_0-1}$, $P_0 = \{\alpha_0\}^{n_0}$, and $G_0 = \varnothing$. Clearly, C_0 is reachable and meets the above conditions.

Now consider any $1 \leq i < n_0$ and suppose that $C_0, \ldots, C_{i-1}$ have already been defined; we define C_i recursively as follows (see Figure 2 for an illustration). Let $\mathcal{D} = (D_0, \ldots, D_x)$ be the sequence of configurations obtained by running $\mathcal{A}$ on K_{n_0-i+1}, the complete graph on $n_0 - i + 1$ nodes, where D_x is the first configuration in which the broadcaster declares termination. Note that, by the symmetry of the complete graph, each configuration in $\mathcal{D}$ has one state for the broadcaster and one state shared by every other node. Let $j \in \{0, \ldots, x\}$ and $k \in \{0, \ldots, i-1\}$ be such that $\beta_k \in D_j$ and $D_j \setminus \{\beta_k\} \subseteq A_k$ and for any other j', k' fulfilling this condition, $j' \leq j$. Note that such a j must exist since, firstly, D_0 is the initial configuration of K_{n_0-i+1} and thus $\beta_0 \in D_0$ and $D_0 \setminus \{\beta_0\} = \{\alpha_0\}^{n_0-i} \subseteq \{\alpha_0\}^{n_0-1} = A_0$; and secondly, the sequence $\mathcal{D}$ is finite since $\mathcal{A}$ terminates in finite time. We also have $j < x$ since otherwise there exists a configuration C_k containing $|P_k| \geq n_0 - k > 0$ uninformed nodes (by induction), but $\beta_k \in D_x$ is a terminating state, contradicting the correctness of $\mathcal{A}$.

Configuration C_k is reachable, so there is some time-varying graph upon which the execution of $\mathcal{A}$ will within finite time be in configuration $C_k = B_k \cup A_k \cup P_k \cup G_k$. We extend this execution by putting this graph into the following topology and executing $\mathcal{A}$ for a single additional round. In a slight abuse of notation, we refer here to configurations as sets of nodes instead of multisets of states since edge dynamics make nodes in the same state interchangeable. Choose any subset $A'_k \subseteq A_k$ of $n_0 - i$ nodes (which is well-defined since $|A_k| = n_0 - k - 1 \geq n_0 - i$ by induction) and arrange them with B_k as a complete graph K_{n_0-i+1}. Arrange the nodes in P_k as a path. Connect these components by attaching an end $p \in P_k$ of the path to some node $a \in A'_k$ and connect every node in $G_k \cup (A_k \setminus A'_k)$ only to this node a. Let C_i be the configuration obtained after executing one round of $\mathcal{A}$ on this topology. Clearly C_i is reachable; we show next that it satisfies the required conditions.

1. $|C_i| = |C_k| = 2n_0$ as desired.
2. Let $B_i = \{\beta_i\}$ be the state of the unique node in B_k after this one additional round.
3. Consider any two nodes $a_1, a_2 \in A'_k \setminus \{a\}$ before execution of the round. Notice that they are both connected to B_k and every node in A'_k except for themselves. By induction, every node in A'_k has the same state α_k. Thus, a_1 and a_2 receive the same sets of messages from their neighbors, so they transition to the same state in the next round. Let α_i be this state shared by all nodes in $A'_k \setminus \{a\}$ after the round's execution and let A_i be the corresponding configuration. Thus $A_i = \{\alpha_i\}^{|A'_k|-1} = \{\alpha_i\}^{n_0-i-1}$ as desired.
4. Consider any node $v \in P_k \setminus \{p\}$ before execution of the round. By induction, v and its neighbors are uninformed. Thus, v receives no message from an informed node in the subsequent round and remains uninformed. Let P_i be the (states of) nodes $P_k \setminus \{p\}$. By induction, we have $|P_i| = |P_k| - 1 \geq n_0 - k - 1 \geq n_0 - i$ as desired.
5. $G_i = C_i \setminus (B_i \cup A_i \cup P_i)$ is simply defined as all the remaining nodes from C_k.
6. Suppose for contradiction that $(\beta_i, \alpha_i) = (\beta_\ell, \alpha_\ell)$ for some $\ell \in \{0, \ldots, i-1\}$. Recall that the configuration $D_j = \{\beta_k\} \cup \{\alpha_k\}^{n_0-i}$ was obtained by running $\mathcal{A}$ on K_{n_0-i+1} and D_j is not the final configuration of that execution. Moreover, observe that when we extended the execution reaching C_k by one round, we reconstructed this exact configuration and topology by arranging $A'_k \cup B_k$ as a complete graph. When running $\mathcal{A}$ for one

additional round on this component, the broadcaster in D_j and B_k transitions from β_k to β_i and the non-broadcasters in D_j and $A'_k \setminus \{a\}$ transition from α_k to α_i. Thus, $D_{j+1} = \{\beta_i\} \cup \{\alpha_i\}^{n_0-i} = \{\beta_\ell\} \cup \{\alpha_\ell\}^{n_0-i}$, by supposition. But this implies $\beta_\ell \in D_{j+1}$ and $D_{j+1} \setminus \{\beta_\ell\} = \{\alpha_\ell\}^{n_0-i} \subseteq \{\alpha_\ell\}^{n_0-\ell} = A_\ell$, contradicting the maximality of j.

Thus, we obtain the desired sequence of n_0 reachable configurations $(C_i)_{i=0}^{n_0-1}$ and their corresponding distinct state pairs (β_i, α_i). The initial state pair (β_0, α_0) appears in every execution of $\mathcal{A}$ and the remaining state pairs (β_i, α_i) for $i \geq 1$ appear in executions of $\mathcal{A}$ on complete graphs K_{n_0-i+1} of at most n_0 nodes. But the maximum number of states $\mathcal{A}$ can use on dynamic networks of at most n_0 nodes is $f(n_0) < n_0^{1/2}$, so there are only $f(n_0)^2 < n_0$ distinct state pairs, a contradiction. ◀

4 Memory Lower Bound for Stabilizing Termination from Idle-Start

We now turn our attention from termination detection to stabilizing termination, requiring only that all nodes are eventually informed and stop sending messages. In this section, we prove that any idle-start algorithm solving broadcast with stabilizing termination must use superconstant memory. This shows that stabilizing broadcast is strictly harder in dynamic networks than in the static setting, which has a trivial constant-memory algorithm (if uninformed and receiving a message, become informed and forward the message to all neighbors) and even permits an algorithm with no persistent memory at all [23, 24].

Our proof will use similar time-varying graphs as in the proofs of Theorems 1 and 3. However, those proofs derived contradictions from the broadcaster declaring termination too early, a condition that cannot be used in the case of stabilizing termination. Instead, we suppose a constant memory algorithm exists and use it to create an infinite sequence of configurations satisfying some very specific properties. We show in Lemma 6, however, that no infinite sequence with these properties exists.

In Lemma 5, we show that the time an algorithm takes to stabilize on a static complete graph, starting from any reachable configuration, depends only on the number of states used and not on the number of nodes in the graph. This will be useful in our proof since we consider algorithms using constant memory, thus implying a fixed bound on the stabilization time for any complete graph, regardless of size. To this end, we define the following function.

▶ **Definition 4.** *Let $f(k)$ be the minimum number of rounds such that for any idle-start algorithm $\mathcal{A}$ solving broadcast with stabilizing termination using at most k states and any configuration C reachable by $\mathcal{A}$, an execution of $\mathcal{A}$ on a static complete graph starting in configuration C stabilizes within $f(k)$ rounds.*

We first prove that this function is well-defined and bounded.

▶ **Lemma 5.** $f(k) \leq 3k!$

Proof. Let $\mathcal{A}$ be an idle-start algorithm solving broadcast with stabilizing termination using only k states and consider its execution on a static complete graph starting from some reachable configuration C. At each round, consider partitioning the nodes into $\ell \leq k$ sets by their current state. In each round, any nodes with the same state receive the same messages from their neighbors and transition to the same next state, so nodes sharing a state continue to do so throughout the execution. Thus, although the number of sets ℓ can decrease over time as some nodes converge to the same state, it can never increase. If there are ℓ sets of nodes with distinct states at one time, there are $\binom{k}{\ell}$ possibilities for what these states are and $\ell!$ ways for these states to be assigned to the sets. Again, since the number of of sets can decrease, ℓ may take on any value in $\{1, \ldots, k\}$ throughout the execution.

Since $\mathcal{A}$ is deterministic, if the same assignment of states to the same sets ever occurs twice in an execution, the algorithm must be in a loop. However, this can not happen since C is reachable and thus $\mathcal{A}$ must stabilize in finite time. Thus, the maximum number of configurations $\mathcal{A}$ can visit before stabilizing is

$$\sum_{\ell=1}^{k} \binom{k}{\ell} \cdot \ell! = \sum_{\ell=1}^{k} \frac{k!}{(k-\ell)!} = k! \cdot \sum_{i=0}^{k-1} \frac{1}{i!} \leq k! \cdot e$$ ◀

Our next lemma sets us up for the contradiction in Theorem 7.

▶ **Lemma 6.** *Let $\mathcal{S} = \{S_1, S_2, \dots\}$ be a collection of multisets S_i which are each the disjoint union of multisets $A_i \cup B_i$ with elements from $[n] = \{1, \dots, n\}$ satisfying $|A_i| = k$, $|B_i|$ is finite, and $\neg(A_j = A_i \wedge B_j \subseteq B_i)$ for all $i \neq j$. Then $|\mathcal{S}|$ is finite.*

Proof. Suppose for contradiction that $\mathcal{S}$ is an infinite collection fulfilling these conditions. Since each A_i contains elements from $[n]$ and $|A_i| = k$, there are only a finite number of possible definitions for the multiset A_i. Since $\mathcal{S}$ is infinite, it must contain some infinite subcollection $\mathcal{S}' = \{S'_1, S'_2, \dots\} \subseteq \mathcal{S}$ such that $A'_i = A'_j$ for all $S'_i, S'_j \in \mathcal{S}'$. Thus, for $\mathcal{S}$ to fulfill the conditions, we must have $B'_i \not\subseteq B'_j$ for all $S'_i \neq S'_j$. We will derive a contradiction by finding multisets in $\mathcal{X}_0 = \{B'_i \mid S'_i \in \mathcal{S}'\}$ such that one is a subset of the other. In fact, we show something much more general: $\mathcal{X}_0$ contains an infinite subcollection of equivalent multisets, i.e., multisets containing the same elements with the same multiplicities.

We will use $\#(S, i)$ to denote the multiplicity of element $i \in [n]$ in multiset S. Choose any multiset $X \in \mathcal{X}_0$. For each $i \in [n]$, let $\mathcal{X}_1^{(i)} = \{B' \in \mathcal{X}_0 \setminus \{X\} \mid \#(B', i) < \#(X, i)\}$ be the collection of multisets of $\mathcal{X}_0$ containing fewer instances of i than X. Each $B' \in \mathcal{X}_0 \setminus \{X\}$ must exist in at least one of these collections since $X \not\subseteq B'$. But there are only n collections $\mathcal{X}_1^{(1)}, \dots, \mathcal{X}_1^{(n)}$, and $\mathcal{X}_0$ is infinite, so there must exist a collection $\mathcal{X}_1^{(i_1)}$ that is infinite. For each $0 \leq j < \#(X, i_1)$, let $\mathcal{X}_1^{(i_1,j)} = \{B' \in \mathcal{X}_1^{(i_1)} \mid \#(B', i_1) = j\}$ be the multisets in $\mathcal{X}_1^{(i_1)}$ containing exactly j instances of element i_1. Once again, there are infinitely many multisets in $\mathcal{X}_1^{(i_1)}$ but only a finite range of multiplicities j, so at least one collection $\mathcal{X}_1^{(i_1,j)}$ is infinite. Call this one $\mathcal{X}_1$.

Next define $\mathcal{X}_2$ in a similar way, but using $\mathcal{X}_1$ in place of $\mathcal{X}_0$. By our construction, every multiset in $\mathcal{X}_1$ contains the same number of instances of element $i_1 \in [n]$. When defining $\mathcal{X}_2$ then, we will have $\mathcal{X}_2^{(i_1)} = \varnothing$. Thus, there will be an element $i_2 \in [n]$ with $i_2 \neq i_1$ such that every multiset in $\mathcal{X}_2$ has the same number of instances of i_2. Since $\mathcal{X}_2 \subset \mathcal{X}_1$, every multiset in $\mathcal{X}_2$ has the same number of instances of both i_1 and i_2. If we continue to define the collections $\mathcal{X}_3, \dots, \mathcal{X}_n$ in this way, all multisets in each $\mathcal{X}_j$ will have the same numbers of instances of $i_1, \dots, i_j$. Thus, for every $X, Y \in \mathcal{X}_n$ and $i \in [n]$, $\#(X, i) = \#(Y, i)$. But then $\mathcal{X}_n$ is an infinite subcollection of equivalent multisets in $\mathcal{X}_0$, a contradiction. ◀

We now prove our superconstant memory lower bound.

▶ **Theorem 7.** *Any idle-start algorithm that solves broadcast with stabilizing termination for anonymous, synchronous, 1-interval connected dynamic networks must use $\omega(1)$ memory.*

Proof. Suppose for contradiction that $\mathcal{A}$ is an idle-start algorithm with $\Theta(1)$ space complexity that solves broadcast with stabilizing termination. Then there is a constant k such that $\mathcal{A}$ never uses more than k states. Note that throughout the rest of this proof, $f(k)$ is well-defined and has finite value by Lemma 5. We will show that the existence of this algorithm $\mathcal{A}$ contradicts Lemma 6 by constructing an infinite sequence of reachable configurations $(C_i)_{i=0}^{\infty}$ where each C_i is a disjoint union of multisets $S_i \cup T_i$ satisfying:

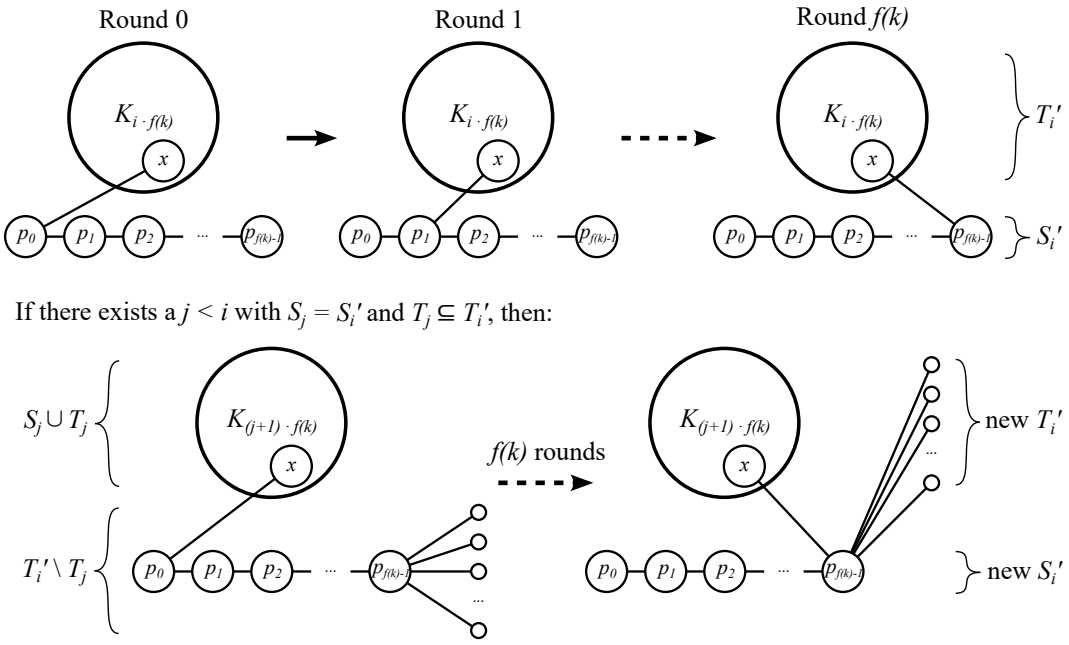


Figure 3 The time-varying graph structures used in the proof of Theorem 7.

1. $|S_i| = f(k)$,
2. $|T_i| = i \cdot f(k)$, and
3. $\neg(S_j = S_i \wedge T_j \subseteq T_i)$ for all $j \neq i$.

Initially, the broadcaster is in some state β_0 and all other nodes are in some other state $\alpha_0 \neq \beta_0$. Define the initial configuration C_0 by letting $S_0 = \{\beta_0\} \cup \{\alpha_0\}^{f(k)-1}$ and $T_0 = \varnothing$. Clearly, C_0 is reachable and satisfies the above conditions.

Now consider any $i \geq 1$ and suppose configurations $C_0, \ldots, C_{i-1}$ have already been defined; we inductively define C_i as follows. Construct a time-varying graph $\mathcal{G}$ from two static components: the complete graph $K_{i \cdot f(k)}$ on $i \cdot f(k)$ nodes (including the broadcaster) and a path $p_0 p_1 \cdots p_{f(k)-1}$ (Figure 3, top). In each round $t \in \{0, \ldots, f(k)-1\}$, these components are connected by a single edge $\{x, p_t\}$, where $x \in K_{i \cdot f(k)}$ is some fixed node. It can be easily shown – as we did by induction in the proof of Theorem 1 – that the nodes in $K_{i \cdot f(k)}$ send the same messages and transition to the same states in an execution of $\mathcal{A}$ on $\mathcal{G}$ as they would in an execution of $\mathcal{A}$ on $K_{i \cdot f(k)}$ by itself. By definition, these executions must stabilize on $K_{i \cdot f(k)}$ within $f(k)$ rounds. At this point, let S_i' and T_i' be the multisets of states of the nodes in the path and complete graph, respectively.

If there is no $j \in \{0, \ldots, i-1\}$ such that $S_j = S_i'$ and $T_j \subseteq T_i'$, then we define configuration C_i as $S_i = S_i'$ and $T_i = T_i'$. Otherwise, if such a j does exist, we extend the execution of $\mathcal{A}$ on $\mathcal{G}$ as follows (Figure 3, bottom). Arrange the nodes of $C_j = S_j \cup T_j$ as the complete graph $K_{(j+1) \cdot f(k)}$ and any $f(k)$ nodes from $T_i' \setminus T_j$ as a path $p_0 p_1 \cdots p_{f(k)-1}$; attach the remaining nodes to the far end $p_{f(k)-1}$ of the path. Then repeat the same process as before, executing $\mathcal{A}$ for $f(k)$ rounds when these components are connected in round t by a single edge $\{x, p_t\}$, where $x \in K_{(j+1) \cdot f(k)}$ is any fixed node. Configuration C_j is reachable by induction, and again all nodes of $K_{(j+1) \cdot f(k)}$ must send the same messages and transition to the same states in the execution of $\mathcal{A}$ on $\mathcal{G}$ as they would in an execution of $\mathcal{A}$ on $K_{(j+1) \cdot f(k)}$ alone. So, by definition of $f(k)$, all nodes in $K_{(j+1) \cdot f(k)}$ must be idle after these $f(k)$ rounds. Also, $p_{f(k)-1}$

remains idle throughout this extended execution, so the initially idle non-path nodes attached to it are also idle at this time. Redefine S'_i as the $f(k)$ path nodes and T'_i as all other nodes, which as we've argued will all be idle. Again, if there is no $j \in \{0, \ldots, i-1\}$ such that $S_j = S'_i$ and $T_j \subseteq T'_i$, then define configuration C_i as $S_i = S'_i$ and $T_i = T'_i$; otherwise, extend the execution of $\mathcal{A}$ on $\mathcal{G}$ by another $f(k)$ rounds as above.

Suppose for contradiction that the condition for defining C_i is never met and the execution of $\mathcal{A}$ on $\mathcal{G}$ is extended forever. In every round of this execution, there is at least one idle node. So consider the execution of $\mathcal{A}$ on a modified $\mathcal{G}$ containing an extra node that is attached to some idle node in each round. This execution is identical to the one on $\mathcal{G}$, but the extra node would never leave its initial state α_0. Thus, there must also be non-idle nodes in every round, or else this execution stabilizes with uninformed nodes, contradicting the correctness of $\mathcal{A}$. But then this infinite execution contains non-idle nodes in every round, contradicting the supposition that $\mathcal{A}$ eventually stabilizes. Thus, multisets S'_i and T'_i with the desired condition will be found in finite time. Clearly, $C_i = S_i \cup T_i = S'_i \cup T'_i$ is reachable; we conclude by showing it satisfies the required conditions.

1. Every intermediate S'_i is defined as the states of nodes in the path components which always comprise $f(k)$ nodes. Thus, $|S_i| = f(k)$.
2. The time-varying graph $\mathcal{G}$ is defined on $|K_{i \cdot f(k)}| + |p_0 p_1 \cdots p_{f(k)-1}| = (i+1) \cdot f(k)$ nodes. Since $|S_i| = f(k)$, we have $|T_i| = i \cdot f(k)$.
3. Consider any $j \neq i$. If $j < i$ then this condition must be met since the execution defining C_i only stopped once it was true. Otherwise, if $j > i$, then by Condition 2, $|T_j| > |T_i|$ and thus $T_j \not\subseteq T_i$ is trivially true.

Thus, the infinite sequence $(C_i)_{i=0}^{\infty}$ can be defined fulfilling all of the conditions above. But this contradicts Lemma 6 stating that all such sequences are finite. ◀

5 A Logspace Algorithm for Stabilizing Termination

In this section, we shift our attention from impossibilities and lower bounds to an idle-start algorithm called Countdown that solves broadcast with stabilizing termination in our anonymous, dynamic setting. This algorithm uses $\mathcal{O}(\log n)$ memory – which is not far from the $\omega(1)$ lower bound of Theorem 7 – and stabilizes in $\mathcal{O}(n)$ rounds which is worst-case asymptotically optimal.

At a high level, the Countdown algorithm (Algorithm 1) coordinates a sequence of broadcast attempts, each lasting twice as many rounds as its predecessor until one succeeds. To facilitate these attempts, nodes store two values: `Current`, the number of rounds remaining in the current attempt; and `Maximum`, the total duration of the current attempt. In each round, non-idle nodes involved in an ongoing attempt broadcast their `Current` and `Maximum` values to their neighbors and then decrement `Current`. Idle nodes that were previously not involved in the attempt but receive these messages will join in by setting their own `Current` and `Maximum` values accordingly. This continues until messages are sent with `Current` $= 0$, indicating the end of the current attempt. If any idle node receives such a message, it detects that the broadcast should have gone on for longer. It responds by initiating a new broadcast attempt whose duration is double the previous one. These attempts continue until some attempt makes all nodes non-idle, at which point no node will initiate another attempt and the algorithm will stabilize.

Before analyzing this algorithm's correctness and complexity, we define some notation. Let $v.\texttt{var}_t$ denote the value of variable `var` in the state of node v at time t (i.e., the start of round t). In this notation, the Countdown algorithm initializes the broadcaster b with $b.\texttt{Current}_0 = 0$

Algorithm 1 COUNTDOWN for Node v.

Initialization. Set `Current` to 0 and `Maximum` to 1 if v is the broadcaster and both variables to -1 otherwise.

Sending Messages.
1: **if** `Current` $\neq -1$ **then**
2: SEND: `msg`(`Current`, `Maximum`)

State Transitions.
3: **if** `Current` $\neq -1$ **then**
4: `Current` $\leftarrow$ `Current` $- 1$
5: **else if** a message `msg`(c, m) was received **then**
6: **if** $c = 0$ **then** ▷ Initiate a new attempt.
7: `Current` $\leftarrow 2m$
8: `Maximum` $\leftarrow 2m$
9: **else if** $c > 0$ **then** ▷ Join the ongoing attempt.
10: `Current` $\leftarrow c - 1$
11: `Maximum` $\leftarrow m$

and $b.\texttt{Maximum}_0 = 1$ and all other nodes $v \neq b$ with $v.\texttt{Current}_0 = v.\texttt{Maximum}_0 = -1$. Denote the set of non-idle nodes in round t as $S_t = \{v \in V : v.\texttt{Current}_t \neq -1\}$. We begin our analysis by proving that all non-idle nodes share the same `Current` and `Maximum` values.

▶ **Lemma 8.** *For all times t and any non-idle node $v \in S_t$, we have $v.\texttt{Current}_t = c_t$ and $v.\texttt{Maximum}_t = m_t$, where $c_0 = 0$, $m_0 = 1$, and*

$$(c_{t+1}, m_{t+1}) = \begin{cases} (2m_t, 2m_t) & \text{if } c_t = 0; \\ (c_t - 1, m_t) & \text{otherwise.} \end{cases}$$

Proof. Argue by induction on t. Only the broadcaster b is initially non-idle, so $c_0 = b.\texttt{Current}_0 = 0$ and $m_0 = b.\texttt{Maximum}_0 = 1$ by initialization. Now suppose the lemma holds up to and including some time $t \geq 0$ and let c_t and m_t be the unique values of $v.\texttt{Current}_t$ and $v.\texttt{Maximum}_t$ for all $v \in S_t$, respectively. Consider any node $v \in S_{t+1}$; if none exist, the lemma holds trivially. We have two cases:

1. $c_t = 0$. Suppose to the contrary that $v \in S_t$; i.e., v was also non-idle at time t. Then $v.\texttt{Current}_t = c_t = 0$. Thus, v must execute Line 4 in round t, yielding $v.\texttt{Current}_{t+1} = -1$ and becoming idle by time $t+1$, a contradiction. So v was idle at time t but became non-idle by time $t+1$, meaning it must have received one or more messages from non-idle nodes in round t. By Line 2 and the induction hypothesis, all of those messages are `msg`$(c_t = 0, m_t)$. So v must execute Lines 7–8 in round t, yielding $v.\texttt{Current}_{t+1} = v.\texttt{Maximum}_{t+1} = 2m_t$. Our choice of $v \in S_{t+1}$ was arbitrary, so $c_{t+1} = m_{t+1} = 2m_t$.
2. $c_t > 0$. First suppose $v \in S_t$. Then $v.\texttt{Current}_t = c_t > 0$, so v executes Line 4 in round t, yielding $v.\texttt{Current}_{t+1} = c_t - 1$ and $v.\texttt{Maximum}_{t+1} = m_t$ as claimed. Now suppose $v \notin S_t$. To transition from idle to non-idle in round t, v must receive one or more messages from non-idle nodes in round t. By Line 2, all of those messages are `msg`$(c_t > 0, m_t)$. So v must execute Lines 10–11, yielding $v.\texttt{Current}_{t+1} = c_t - 1$ and $v.\texttt{Maximum}_{t+1} = m_t$. Our choice of $v \in S_{t+1}$ was arbitrary, so $c_{t+1} = c_t - 1$ and $m_{t+1} = m_t$. ◀

Using the c_t and m_t values defined in Lemma 8, we next show that a broadcast attempt lasting k rounds either informs all nodes and stabilizes or involves at least $k + 1$ non-idle nodes before initiating a new attempt.

▶ **Lemma 9.** *If $k := c_t = m_t > 0$, then either (1) $S_{t+k} = V$ and $S_{t+k+1} = \varnothing$, or (2) $|S_{t+k}| \geq k+1$ and $c_{t+k+1} = m_{t+k+1} = 2k$.*

Proof. Consider any time t at which $k := c_t = m_t > 0$, the start of a new broadcast attempt by the set of nodes S_t. First suppose that for some $0 \leq i \leq k$, we have $S_{t+i} = V$; i.e., all nodes are non-idle. By Lemma 8, we have $c_{t+i} = c_t - i$. Since all nodes v are non-idle at time $t+i$ and thus have $v.\texttt{Current}_{t+i} = c_{t+i} = c_t - i$, they all execute Line 4 in round $t+i$ by decrementing $v.\texttt{Current}$. If $i < k$, then $v.\texttt{Current}_{t+i+1} = c_t - i - 1 \neq -1$, so all nodes remain non-idle and the process repeats; otherwise, if $i = k$, then $v.\texttt{Current}_{t+i+1} = c_t - k - 1 = -1$. This renders all nodes idle at time $t+k+1$, so Case 1 has occurred.

Now suppose that $S_{t+i} \neq V$ for all $0 \leq i \leq k$; i.e., there is at least one idle node throughout the attempt. We argue by induction on $0 \leq i \leq k$ that $|S_{t+i}| \geq |S_t| + i$; i.e., at least one idle node becomes non-idle in each round. The $i = 0$ case holds trivially, so suppose the claim holds up to and including some $i < k$. Since $c_{t+i} = c_t - i = k - i$ by Lemma 8, any node in S_{t+i} must remain non-idle until time $t + i + c_{t+i} = t + k$. So $|S_{t+i+1}| \geq |S_{t+i}|$. If the induction hypothesis is in fact a strict inequality, we are done:

$$|S_{t+i}| > |S_t| + i \quad \Rightarrow \quad |S_{t+i+1}| \geq |S_{t+i}| \geq |S_t| + i + 1.$$

So suppose instead that $|S_{t+i}| = |S_t| + i$. There must exist non-idle nodes at time $t+1$ since $c_{t+1} = c_t - 1 > -1$ by Lemma 8, and there must exist idle nodes at time $t+1$ by supposition. Thus, since the dynamic network is 1-interval connected, there must be some idle node $v \in V \setminus S_{t+i}$ that receives a message from a non-idle node in round $t+1$, causing v to become non-idle (Lines 7–8 or 10–11). By the induction hypothesis, $|S_{t+i+1}| \geq |S_{t+i}| + 1 = |S_t| + i + 1$.

By this induction argument, we have $|S_{t+k}| \geq |S_t| + k \geq k+1$. By Lemma 8, we have that $c_{t+k} = c_t - k = 0$, and with another application of the same lemma, we conclude that $c_{t+k+1} = m_{t+k+1} = 2m_{t+k} = 2m_t = 2k$. So Case 2 has occurred. ◀

Our algorithm's correctness follows from the previous lemma and its time and space complexities are obtained with straightforward counting arguments.

▶ **Theorem 10.** *Countdown (Algorithm 1) correctly solves broadcast with stabilizing termination from an idle start in $\mathcal{O}(n)$ rounds and $\mathcal{O}(\log n)$ space for anonymous, synchronous, 1-interval connected dynamic networks.*

Proof. By Lemma 8, we have $c_1 = m_1 = 2$, allowing us to apply Lemma 9. But suppose to the contrary that this and all subsequent applications of the lemma result in Case 2 – where a new attempt is initiated with double the duration – and not Case 1, where all nodes are informed ($S_{t+k} = V$) and the algorithm stabilizes ($S_{t+k+1} = \varnothing$). Then there exists an attempt of duration $k \geq n$, which, by Lemma 9, ends with $|S_{t+k}| \geq k+1 \geq n+1$ non-idle nodes. But there are only n nodes in the network, a contradiction. So Countdown must inform all nodes and stabilize in finite time.

It remains to bound runtime and memory. Each time a new broadcast attempt is initiated, the `Maximum` variable is doubled and the algorithm runs for another `Maximum` rounds. As we already showed, once `Maximum` reaches or exceeds n, the subsequent attempt will inform all nodes and stabilize. Thus, `Maximum` doubles at most $\lceil \log_2 n \rceil$ times, meaning Countdown stabilizes in at most $\sum_{i=0}^{\lceil \log_2 n \rceil} 2^i = \mathcal{O}(2^{1+\lceil \log_2 n \rceil} - 1) = \mathcal{O}(n)$ rounds. This analysis also shows that the largest attainable `Maximum` value before its final doubling is $n-1$, so $\texttt{Maximum} \leq 2(n-1) = \mathcal{O}(n)$. Since $-1 \leq \texttt{Current} \leq \texttt{Maximum}$, we also have $\texttt{Current} = \mathcal{O}(n)$, implying that Countdown has $\mathcal{O}(\log n)$ space complexity. ◀

6 Conclusion

This paper investigated what memory is necessary for anonymous, synchronous, 1-interval connected dynamic networks to deterministically solve broadcast with some termination conditions. We considered both *termination detection* where the broadcaster must eventually declare that every node has been informed and *stabilizing termination* where nodes must eventually stop sending messages. Combining our results with the established literature, we now know the following about this problem:

- *Termination Detection*. Regardless of memory, broadcast with termination detection is impossible for idle-start algorithms (Theorem 1) and for non-idle-start algorithms when the number of broadcasters is unknown (Theorem 2). Any (non-idle-start) algorithm solving broadcast with termination detection must use $\Omega(\log n)$ memory per node (Theorem 3). The best known space complexity for this problem follows from Di Luna and Viglietta's history trees algorithm which uses $\mathcal{O}(n^3 \log n)$ memory in the worst case [16].
- *Stabilizing Termination*. Any idle-start algorithm solving broadcast with stabilizing termination must use $\omega(1)$ memory per node (Theorem 7). As a positive result, this problem is solvable with logarithmic memory under standard synchrony: Countdown is a $\mathcal{O}(\log n)$ memory, linear time algorithm achieving stabilizing termination without identifiers or knowledge of n (Theorem 10).

For stabilizing termination, our $\omega(1)$ memory bound holds only for idle-start algorithms and our $\mathcal{O}(\log n)$ memory Countdown algorithm happens to be idle-start. In the non-idle-start regime where non-broadcaster nodes can send messages from their initial states, can we obtain a sublogarithmic space algorithm? Can any lower bound be shown? The contradiction at the heart of our lower bound technique for idle-start algorithms identified configurations C_i that are reachable with or without an extra uninformed node. In the non-idle-start case, however, this extra uninformed node may send new and unaccounted for messages and we can no longer guarantee C_i will still be reached, requiring a different approach.

The $\Omega(\log n)$ and $\mathcal{O}(n^3 \log n)$ memory bounds for termination detection leave open a significant gap for further improvement. Our logarithmic lower bound shows that termination detection requires enough memory to count to n, and indeed there is a straightforward solution for termination detection if (an upper bound on) n can be obtained: simply wait for n rounds after broadcasting information for the first time and then declare all nodes have been informed. Can broadcast with termination detection be achieved without solving exact counting? If not, what approaches could yield algorithms for exact counting that are more space-efficient than history trees? Viglietta recently proposed the existence of logspace counting algorithms as an open problem unlikely to be solved by history trees [37]; we are unsure such algorithms exist at all, as we suspect our $\Omega(\log n)$ bound can be improved.

Finally, we note that all impossibility results and lower bounds in this paper apply also to any problem broadcast reduces to, such as exact counting. Thus, these results and any future improvements shed important light on the requirements of nontrivial terminating computation in anonymous dynamic networks.

References

1 Karine Altisen, Stéphane Devismes, Anaïs Durand, Colette Johnen, and Franck Petit. Self-Stabilizing Systems in Spite of High Dynamics. *Theoretical Computer Science*, 964:113966, 2023. doi:10.1016/j.tcs.2023.113966.

2 David Andersen, Hari Balakrishnan, Frans Kaashoek, and Robert Morris. Resilient Overlay Networks. In *Proceedings of the 18th ACM Symposium on Operating Systems Principles*, pages 131–145, Banff, AB, Canada, 2001. ACM. doi:10.1145/502034.502048.

3 John Augustine, Gopal Pandurangan, and Peter Robinson. Distributed Algorithmic Foundations of Dynamic Networks. *ACM SIGACT News*, 47(1):30, 2016.

4 Alysson Bessani, Eduardo Alchieri, João Sousa, André Oliveira, and Fernando Pedone. From Byzantine Replication to Blockchain: Consensus is Only the Beginning. In *2020 50th Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN)*, pages 424–436, Valencia, Spain, 2020. IEEE. doi:10.1109/DSN48063.2020.00057.

5 Arnaud Casteigts. *A Journey through Dynamic Networks (with Excursions)*. Habilitation à diriger des recherches, University of Bordeaux, Bordeaux, France, 2018.

6 Arnaud Casteigts, Paola Flocchini, Bernard Mans, and Nicola Santoro. Deterministic Computations in Time-Varying Graphs: Broadcasting under Unstructured Mobility. In *Theoretical Computer Science*, volume 323 of *IFIP Advances in Information and Communication Technology*, pages 111–124, Berlin, Heidelberg, 2010. Springer. doi:10.1007/978-3-642-15240-5_9.

7 Arnaud Casteigts, Paola Flocchini, Bernard Mans, and Nicola Santoro. Shortest, Fastest, and Foremost Broadcast in Dynamic Networks. *International Journal of Foundations of Computer Science*, 26(04):499–522, 2015. doi:10.1142/S0129054115500288.

8 Arnaud Casteigts, Paola Flocchini, Walter Quattrociocchi, and Nicola Santoro. Time-Varying Graphs and Dynamic Networks. *International Journal of Parallel, Emergent and Distributed Systems*, 27(5):387–408, 2012. doi:10.1080/17445760.2012.668546.

9 Maitri Chakraborty, Alessia Milani, and Miguel A. Mosteiro. A Faster Exact-Counting Protocol for Anonymous Dynamic Networks. *Algorithmica*, 80(11):3023–3049, 2018. doi:10.1007/s00453-017-0367-4.

10 Arjun Chandrasekhar, Deborah M. Gordon, and Saket Navlakha. A Distributed Algorithm to Maintain and Repair the Trail Networks of Arboreal Ants. *Scientific Reports*, 8(1):9297, 2018. doi:10.1038/s41598-018-27160-3.

11 Giuseppe Di Luna and Roberto Baldoni. Non Trivial Computations in Anonymous Dynamic Networks. In *19th International Conference on Principles of Distributed Systems (OPODIS 2015)*, volume 46 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 33:1–33:16, Rennes, France, 2016. Schloss Dagstuhl – Leibniz-Zentrum für Informatik. doi:10.4230/LIPICS.OPODIS.2015.33.

12 Giuseppe A. Di Luna and Giovanni Viglietta. Computing in Anonymous Dynamic Networks Is Linear. In *2022 IEEE 63rd Annual Symposium on Foundations of Computer Science (FOCS)*, pages 1122–1133, Denver, CO, USA, 2022. IEEE. doi:10.1109/FOCS54457.2022.00108.

13 Giuseppe A. Di Luna and Giovanni Viglietta. Optimal Computation in Leaderless and Multi-Leader Disconnected Anonymous Dynamic Networks. In *37th International Symposium on Distributed Computing (DISC 2023)*, volume 281 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 18:1–18:20, L'Aquila, Italy, 2023. Schloss Dagstuhl – Leibniz-Zentrum für Informatik. doi:10.4230/LIPIcs.DISC.2023.18.

14 Giuseppe Antonio Di Luna, Roberto Baldoni, Silvia Bonomi, and Ioannis Chatzigiannakis. Conscious and Unconscious Counting on Anonymous Dynamic Networks. In *Distributed Computing and Networking*, volume 8314 of *Lecture Notes in Computer Science*, pages 257–271, Berlin, Heidelberg, 2014. Springer. doi:10.1007/978-3-642-45249-9_17.

15 Giuseppe Antonio Di Luna, Roberto Baldoni, Silvia Bonomi, and Ioannis Chatzigiannakis. Counting in Anonymous Dynamic Networks under Worst-Case Adversary. In *2014 IEEE 34th International Conference on Distributed Computing Systems*, pages 338–347, Madrid, Spain, 2014. IEEE. doi:10.1109/ICDCS.2014.42.

16 Giuseppe Antonio Di Luna and Giovanni Viglietta. Brief Announcement: Efficient Computation in Congested Anonymous Dynamic Networks. In *Proceedings of the 2023 ACM Symposium on Principles of Distributed Computing*, pages 176–179, Orlando, FL, USA, 2023. ACM. doi:10.1145/3583668.3594590.

17 Michael Dinitz, Jeremy Fineman, Seth Gilbert, and Calvin Newport. Smoothed Analysis of Information Spreading in Dynamic Networks. In *36th International Symposium on Distributed Computing (DISC 2022)*, volume 246 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 18:1–18:22, Augusta, GA, USA, 2022. Schloss Dagstuhl – Leibniz-Zentrum für Informatik. doi:10.4230/LIPIcs.DISC.2022.18.

18 Michael Dinitz, Jeremy T. Fineman, Seth Gilbert, and Calvin Newport. Smoothed Analysis of Dynamic Networks. *Distributed Computing*, 31(4):273–287, 2018. doi:10.1007/s00446-017-0300-8.

19 Michael Feldmann, Christian Scheideler, and Stefan Schmid. Survey on Algorithms for Self-Stabilizing Overlay Networks. *ACM Computing Surveys*, 53(4):74:1–74:24, 2020. doi:10.1145/3397190.

20 Paola Flocchini, Giuseppe Prencipe, and Nicola Santoro, editors. *Distributed Computing by Mobile Entities: Current Research in Moving and Computing*, volume 11340 of *Lecture Notes in Computer Science*. Springer, Cham, 2019. doi:10.1007/978-3-030-11072-7.

21 Vincent Gramoli. From Blockchain Consensus Back to Byzantine Consensus. *Future Generation Computer Systems*, 107:760–769, 2020. doi:10.1016/j.future.2017.09.023.

22 Heiko Hamann. *Swarm Robotics: A Formal Approach*. Springer, Cham, 1 edition, 2018. doi:10.1007/978-3-319-74528-2.

23 Walter Hussak and Amitabh Trehan. On Termination of a Flooding Process. In *Proceedings of the 2019 ACM Symposium on Principles of Distributed Computing*, pages 153–155, Toronto, ON, Canada, 2019. ACM. doi:10.1145/3293611.3331586.

24 Walter Hussak and Amitabh Trehan. On the Termination of Flooding. In *37th International Symposium on Theoretical Aspects of Computer Science (STACS 2020)*, volume 154 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 17:1–17:3, Montpellier, France, 2020. Schloss Dagstuhl – Leibniz-Zentrum für Informatik. doi:10.4230/LIPIcs.STACS.2020.17.

25 Walter Hussak and Amitabh Trehan. Termination of Amnesiac Flooding. *Distributed Computing*, 36(2):193–207, 2023. doi:10.1007/s00446-023-00448-y.

26 Dariusz R. Kowalski and Miguel A. Mosteiro. Polynomial Counting in Anonymous Dynamic Networks with Applications to Anonymous Dynamic Algebraic Computations. *Journal of the ACM*, 67(2):1–17, 2020. doi:10.1145/3385075.

27 Fabian Kuhn, Nancy Lynch, and Rotem Oshman. Distributed Computation in Dynamic Networks. In *Proceedings of the 42nd ACM Symposium on Theory of Computing*, pages 513–522, Cambridge, MA, 2010. ACM. doi:10.1145/1806689.1806760.

28 Jintao Liu, Arthur Prindle, Jacqueline Humphries, Marçal Gabalda-Sagarra, Munehiro Asally, Dong-yeon D. Lee, San Ly, Jordi Garcia-Ojalvo, and Gürol M. Süel. Metabolic Co-Dependence Gives Rise to Collective Oscillations within Biofilms. *Nature*, 523(7562):550–554, 2015. doi:10.1038/nature14660.

29 Nancy A. Lynch. *Distributed Algorithms*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1996.

30 Othon Michail, Ioannis Chatzigiannakis, and Paul G. Spirakis. Naming and Counting in Anonymous Unknown Dynamic Networks. In *Stabilization, Safety, and Security of Distributed Systems*, volume 8255 of *Lecture Notes in Computer Science*, pages 281–295, Cham, 2013. Springer. doi:10.1007/978-3-319-03089-0_20.

31 Regina O'Dell and Roger Wattenhofer. Information Dissemination in Highly Dynamic Graphs. In *Proceedings of the 2005 Joint Workshop on Foundations of Mobile Computing*, pages 104–110, Cologne, Germany, 2005. ACM. doi:10.1145/1080810.1080828.

32 Shunhao Oh, Dana Randall, and Andréa W. Richa. Adaptive Collective Responses to Local Stimuli in Anonymous Dynamic Networks. In *2nd Symposium on Algorithmic Foundations of Dynamic Networks (SAND 2023)*, volume 257 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 6:1–6:23, Pisa, Italy, 2023. Schloss Dagstuhl – Leibniz-Zentrum für Informatik. doi:10.4230/LIPIcs.SAND.2023.6.

33 Arthur Prindle, Jintao Liu, Munehiro Asally, San Ly, Jordi Garcia-Ojalvo, and Gürol M. Süel. Ion Channels Enable Electrical Communication in Bacterial Communities. *Nature*, 527(7576):59–63, 2015. doi:10.1038/nature15709.

34 Michel Raynal, Julien Stainer, Jiannong Cao, and Weigang Wu. A Simple Broadcast Algorithm for Recurrent Dynamic Systems. In *2014 IEEE 28th International Conference on Advanced Information Networking and Applications*, pages 933–939, Victoria, BC, Canada, 2014. IEEE. doi:10.1109/AINA.2014.115.

35 Leonie Reichert, Samuel Brack, and Björn Scheuermann. A Survey of Automatic Contact Tracing Approaches Using Bluetooth Low Energy. *ACM Transactions on Computing for Healthcare*, 2(2):1–33, 2021. doi:10.1145/3444847.

36 Corina E. Tarnita, Alex Washburne, Ricardo Martinez-Garcia, Allyson E. Sgro, and Simon A. Levin. Fitness Tradeoffs between Spores and Nonaggregating Cells Can Explain the Coexistence of Diverse Genotypes in Cellular Slime Molds. *Proceedings of the National Academy of Sciences*, 112(9):2776–2781, 2015. doi:10.1073/pnas.1424242112.

37 Giovanni Viglietta. History Trees and Their Applications. In *Structural Information and Communication Complexity*, volume 14662 of *Lecture Notes in Computer Science*, pages 3–23, Cham, 2024. Springer. doi:10.1007/978-3-031-60603-8_1.