Broadcasts in Anonymous, Dynamic Networks: A New Algorithm and Impossibility Results

Volker Turau
Hamburg University of Technology, Germany

Abstract

The broadcast problem is the task of disseminating a message from a single source node to all other nodes in a distributed system, ensuring that every node eventually receives the message despite possible network constraints such as delays, failures, or limited topology knowledge. In this work, we consider the broadcast problem in anonymous, synchronous, dynamic networks. A dynamic network is a network, whose topology changes over time, meaning that communication links can unpredictably appear and disappear. We present a randomized algorithm requiring $O(\log \log n)$ bits of storage per node and terminating in $O(m \log n)$ rounds with high probability. It solves broadcast with stabilizing termination for anonymous, synchronous, 1-interval-connected networks using messages of size $O(1)$. The algorithm is a non-idle-start algorithm. The best known idle-start algorithm for this problem requires $O(\log n)$ space, also a lower memory bound of $\omega(1)$ space is known. Our contribution affirmatively answers a question of Parzych and Daymude (DISC 2024). We also extend this result to dynamic networks with bounded connectivity time. Furthermore, we prove that for two variants of the broadcast problem in this setting no randomized algorithms exist.

2012 ACM Subject Classification Theory of computation → Distributed algorithms; Theory of computation → Graph algorithms analysis

Keywords and phrases Distributed algorithms, dynamic networks, randomized algorithms, impossibility results

Digital Object Identifier 10.4230/LIPIcs.SAND.2026.6

1 Introduction

In recent years biological scenarios have sparked the interest in memory-less or extremely low memory networks. This has led to sub-logarithmic space algorithms. For example, the bit-dissemination problem was studied under the assumption that agents only have a moderate amount of memory available [2]. An efficient protocol was proposed in [11]. It achieves consensus in $O(\textit{polylog}\ n)$ rounds with high probability assuming that each node of the network can only memorize $O(\log \log n)$ bits of information from one round to the next. Propagating information from one or more network nodes to the rest of the network has been the in focus of research in distributed computing for many years. It has been studied under various assumptions using different names such as rumor spreading, gossip, and broadcast.

The task of a broadcast algorithm is to distribute a message from one network node to all other nodes. There are numberless applications of broadcast. The simplest realization of broadcasting in connected networks is flooding. The originator of a message $\mathfrak{m}$ forwards $\mathfrak{m}$ to all neighbors and when a node receives $\mathfrak{m}$ for the first time, it sends it to all its neighbors in the communication graph. In recent years, many broadcast algorithms have been proposed addressing a wide range of requirements, including dynamic networks and networks where nodes have extremely low memory resources. The focus of this work is on synchronous, anonymous, dynamic networks with low memory nodes. In anonymous networks nodes neither have unique identifiers nor use port numbers. A dynamic network is one, in which the communication topology changes over time, that is, edges can appear and disappear as time progresses [5, 13, 8]. These networks are also known as adversarial dynamic networks, because network nodes do not know in advance during which rounds links become available

or disappear, i.e., links are selected by an adversary. Dynamic networks are used to model mobility, failures, and recoveries. Different theoretical models for dynamic networks assume different levels of stability, e.g., always connected or recurrently connected, etc.

Broadcast algorithms for dynamic systems were investigated under different boundary conditions [18, 4, 3]. Recently Parzych and Daymude considered the broadcast problem in anonymous, dynamic 1-interval-connected networks [17]. They consider two categories of algorithms: *idle-start* and *non-idle-start*. In the first category a node different from the initial broadcasting node may only send a message after it received a message. In the second, all nodes can send messages in every round including the initial round. Furthermore, they distinguish between algorithms that achieve *stabilizing termination* (a.k.a. silent algorithms) and algorithms with *termination detection*. In the first case all nodes eventually stop sending messages. In the second case, the broadcasting node itself becomes aware, when all nodes are informed, i.e. the broadcast is complete.

Parzych et al. proved that broadcast with termination detection is impossible for idle-start algorithms and also for non-idle-start algorithms when the number of broadcasters is unknown. For the case that the termination condition is relaxed to stabilizing termination, they proved that any idle-start algorithm must use $\omega(1)$ memory per node, separating the static and dynamic settings for anonymous broadcast. In addition they present an algorithm solving broadcast in the idle start realm with stabilizing termination in $O(n)$ rounds, $O(\log n)$ space, and messages of size $O(\log n)$. They asked the question whether in the non-idle-start realm a sub-logarithmic space algorithm exists for this problem.

The main contribution of this paper is an affirmatively answer to the question of Parzych et al. within the probabilistic framework. We propose a randomized algorithm $\mathcal{A}$ for stabilizing termination in the non-idle-start realm that with high probability terminates and requires only $O(\log \log n)$ storage per node using messages of size $O(1)$. In particular we prove the following theorem for anonymous, dynamic 1-interval-connected networks.

▶ **Theorem 1.** *There exists a non-idle-start algorithm $\mathcal{A}$ solving the broadcast problem with stabilizing termination with high probability for anonymous, synchronous, dynamic 1-interval-connected networks provided each node knows n. With high probability $\mathcal{A}$ terminates in time $O(m \log n)$, requires $O(\log \log n)$ memory per node, and uses messages of size $O(1)$.*

We also relax the assumption of 1-interval-connectivity to bounded connectivity time θ (see Sect. 3). In this case, the algorithm terminates with high probability in time $O(ct\, m \log ct\, m)$. The statement about memory consumption remains unchanged. Furthermore, we extend some of the impossibility results of [17] to the realm of randomized algorithms. In particular we prove the following two theorems.

▶ **Theorem 2.** *No idle-start Monte Carlo or Las Vegas algorithm can solve broadcast with termination detection for anonymous, synchronous, 1-interval connected dynamic graphs.*

▶ **Theorem 3.** *No non-idle-start Monte Carlo or Las Vegas algorithm can solve broadcast with termination detection for anonymous, synchronous, 1-interval connected dynamic graphs if nodes are unaware of the number of broadcasters.*

The paper is organized as follows. In the next section we discuss related work. This is followed by a description of our model and notation. In Sec. 4 we present the novel broadcast algorithm requiring $O(\log \log n)$ space and prove Theorem 1. The impossibility results – Theorem 2 and 3 – are proved in the final section.

2 State of the Art

The most primitive realization of broadcast in static networks is flooding, a.k.a. uncontrolled broadcast. It can be applied in anonymous, asynchronous, static networks. To control flooding requires each node to locally keep a record of already forwarded messages. For synchronous, static, distributed systems, broadcasting algorithms not requiring any local storage are known. Hussak and Trehan proposed *amnesiac flooding* $\mathcal{A}_{\mathsf{AF}}$ [10]. Algorithm $\mathcal{A}_{\mathsf{AF}}$ does not require nodes to have unique identifiers, it relies on port numbers to identify the neighbors that did not send a message. Both forms of flooding require the network to be connected at any time, i.e., they are unsuitable for dynamic networks [19]. Surprisingly it requires only a simple extension to make $\mathcal{A}_{\mathsf{AF}}$ to work correctly despite link failures, i.e., a mild form of dynamic network. The corresponding algorithm $\mathcal{A}_{\mathsf{AFI}}$ of [19] requires port numbers. It can be used in networks with temporarily intermittent channels, but the number of times links are unavailable must be bounded. All these broadcast algorithms are idle-start algorithms, they don't provide termination detection.

Recently Parzych et al. considered broadcasting in dynamic networks. They assume networks, where nodes are anonymous, i.e., they neither have unique identifiers nor port numbers, and have no knowledge of global parameters, including the number of nodes [17]. The communication is based on a local reliable broadcast mechanism, each node can reliably send a message to all current neighbors, i.e., those that are in the current round connected to the sender. The authors consider synchronous, 1-interval connected dynamic networks and prove that broadcast with termination detection is impossible for idle-start algorithms and otherwise requires $\Omega(\log n)$ memory per node. Also, they prove that even if the termination condition is relaxed to stabilizing termination any idle-start algorithm must use $\omega(1)$ memory per node. Finally, they present an idle-start algorithm solving broadcast with stabilizing termination using $O(\log n)$ memory per node. Table 1 summaries the state of the art for broadcast algorithms in anonymous, synchronous networks.

Table 1 Broadcast Algorithms for Anonymous Synchronous Networks.

	Memory	Requirements	Size Mess.	Runtime
Amnesiac flooding [10]	none	connected graph, port numbers	$O(1)$	$O(n)$
$\mathcal{A}_{\mathsf{AFI}}$ [19]	$O(1)$	bounded number f of unavailable edges, port numbers	$O(1)$	$O(Diam(G)+f)$
Countdown [17]	$O(\log n)$	1-interval connected, idle-start	$O(\log n)$	$O(n)$
This work	$O(\log\log n)$	1-interval connected, non-idle-start, knowledge of n	$O(1)$	$O(m\log n)$
This work	$O(\log\log(\theta m))$	bounded connectivity time θ, non-idle-start, knowledge of n	$O(1)$	$O(\theta m\ \log(\theta m))$

3 Model and Notation

A dynamic network is modeled by a dynamic graph $G=(V,E)$, where V is a set of n nodes and $E:\mathbb{N}\longrightarrow\mathcal{P}(E')$ is a function mapping a round number $r\in\mathbb{N}$ to a set $E(r)$ of edges drawn from $E'=\{(u,v):u,v\in V\}$, here $\mathcal{P}(S)$ denotes the power set of S. Intuitively, a dynamic graph G is an infinite sequence $G_1,G_2,\ldots$ of instantaneous graphs – a.k.a. snapshots

of G – whose edge sets are subsets of E'. Note that nodes do not know in advance during which rounds edges become available. Denote by m the total number of different edges appearing in some $E(i)$, clearly $m \leq n(n-1)/2$. A static network is a special case of a dynamic network in which $E(i+1) = E(i)$ for all $i \in \mathbb{N}$. The set V is assumed throughout this work to be static, that is, it remains the same throughout G's life time.

▶ **Definition 4** ([14]). *The* connectivity time θ *of a dynamic graph* $G = (V, E)$ *is the minimum* $k \in \mathbb{N}$ *such that for all* r *the static graph* $(V, \cup_{i=r}^{r+k-1} E(i))$ *is connected. Dynamic graphs with* $\theta = 1$ *are called* 1-connected interval graphs.

That is to say, during every interval of length θ at least one edge crosses any cut of G.

In this paper we consider anonymous, synchronous, distributed systems and assume that nodes have no port numbers; i.e., they cannot distinguish among their neighbors. Algorithms are executed in rounds of fixed length. Each node synchronously executes the same distributed algorithm. Every round is divided in three phases: (i) nodes receive messages sent in the previous round, (ii) based on the received messages nodes change their state and determine the messages to be sent, and (iii) send messages. Due to the anonymity of the network, we assume that nodes communicate with their neighbors using a local reliable broadcast mechanism. This implies that nodes cannot send different messages to different neighbors in the same round, as usually is possible in networks with port numbers. To distinguish this local broadcast from the global broadcast (i.e., the network-wide task), we call it a *local broadcast*. Only nodes that are connected to the sender in the current round do receive the message. This model captures the situation, for instance, typically found in wireless radio communications, where messages are sent to all nodes within communication range.

The goal of a global broadcasting algorithm is to disseminate a message created by a node (called the *broadcaster*) to all nodes of the network. A node is called *informed* if it is the broadcaster or has received the message from an informed node. A broadcast is complete, when all nodes in the network are informed. We distinguish two types of algorithms: *idle-start* and *non-idle-start* algorithms. In the first case, a node other then the initial broadcaster can only send a message after it received a message from an informed neighbor. In the second case, all nodes can send messages in every round including the initial round.

Furthermore, we distinguish between algorithms that achieve stabilizing termination and algorithms with termination detection. In the first case, all nodes eventually stop sending messages. In the second case, the broadcasting node itself becomes within finite time aware, when all nodes are informed, i.e. the broadcaster can correctly and irrevocably decide that the broadcast is complete.

4 Non-Idle Start Algorithm for Stabilizing Termination

For broadcasting algorithms achieving stabilizing termination a $\omega(1)$ memory bound was proven for idle-start algorithms and 1-interval connected graphs [17]. The authors present an idle-start algorithm called *Countdown* for this task. It coordinates a sequence of local broadcast attempts, each lasting twice as many rounds as its predecessor until one succeeds. Each local broadcast contains a hop-count, which is decremented at informed nodes. If a message with hop-count 0 reaches an informed node, the forwarding ends. When an uninformed node is reached, a new wave of flooding is initiated, the initial hop-count is doubled. To facilitate these attempts, nodes store the number of rounds remaining in the current attempt and the total duration of the current attempt, i.e., the algorithm requires $O(\log n)$ memory. It stabilizes in $O(n)$ rounds and uses messages of size $O(\log n)$.

The authors of [17] asked the question, whether in the the non-idle-start regime, where non-broadcaster nodes can send messages in every round – including the round in which the broadcast was initiated – a sub-logarithmic space algorithm for 1-interval connected networks can be obtained. It appears that all deterministic algorithms require $O(\log n)$ memory since they all – in one way or another – need to count up to n. This even holds for the most simple algorithm in which all nodes broadcast for n rounds the first message they receive, then in every round at least one more node receives the message.

In the following we present a non-idle-start randomized algorithm for 1-interval connected graphs achieving stabilizing termination for the global broadcast problem. With high probability it terminates in $O(m \log n)$ rounds and requires $O(\log \log n)$ bits of storage per node. It uses messages of size $O(1)$. Thus, it affirmatively answers the question of [17] for randomized algorithms. We tackle the challenge in designing a counter that counts long enough to achieve stabilizing termination and with high probability uses only $O(\log \log n)$ bits of storage. This is achieved by using an approximate counting algorithm, this is a probabilistic technique to increment a counter invented by R. Morris [15].

4.1 The Morris$^+$ Counter

In Morris' algorithm, the counter represents an *order of magnitude estimate* of the actual count k. It allows to retrieve a multiplicative approximation to k using only $O(\log \log k)$ bits. The Morris counter was analyzed by Flajolet [9], who showed that $O(\log \log k + \log(1/\epsilon) + \log(1/\delta))$ bits of memory are sufficient to return an ϵ-approximation with probability $1 - \delta$. Unfortunately a success probability of at least $1 - 1/k$ is impossible with $O(\log \log k)$ bits using the Morris counter in its original form. A recent result of Nelson et al. improved this [16]. They proposed a parameterized variant of the Morris counter, the *Morris(a) counter*, with a parameter $a \in (0, 1)$. It maintains a counter *count*. For each event to be counted, *count* is incremented with probability $1/(1 + a)^{count}$. Upon being queried the Morris(a) counter returns the value $((1 + a)^{count} - 1)/a$. The expected return value after k increments is k, it uses only $\log \log k + \log(1/a) + O(1)$ bits. The advantage of this counter is that $O(\log \log k)$ memory suffices to have a failure probability of at most $1/poly(k)$. By setting $a = \epsilon^2/(8 \ln(1/\delta))$, the space usage of Morris(a) is $\log \log k + 2 \log(1/\epsilon) + \log \log(1/\delta) + O(1)$ bits with high probability, and it outputs a $(1 \pm 2\epsilon)$ approximation with probability $1 - 2/\delta$ (Theorem 1.2, [16]).

Unfortunately, there is a small catch to using the Morris(a) counter. The success probability is only guaranteed when the number of counted events is at least $8/a$. Nelson et al. show that this is not a serious limitation. The remedy is to maintain a separate counter $count_{sep}$ exactly, deterministically up until this value. This costs at most $\log(1/a) + O(1)$ bits of space. With $a = \epsilon^2/(8 \ln(1/\delta))$, $\epsilon = 1/4$, and $\delta = 1/n^3$ our given space limit is not violated. Thus, if we count $\hat{N} \geq N_\delta$ events with $N_\delta = \Omega(\log(1/\delta))$ the success probability is guaranteed (see [16] for details). As long as $count_{sep} < N_\delta$ each invocation of the Morris(a) counter increments $count_{sep}$. Whenever $count_{sep} \geq ((1+a)^{count+1} - 1)/a$ occurs, the counter *count* is incremented (see Section 6 for details). In the following we implicitly assume such an implementation of the Morris(a) counter and call it the Morris$^+$ counter.

4.2 The Algorithm $\mathcal{A}$

The usage of a probabilistic counter for the hop-count has the disadvantage that it may significantly under- or overrate the number of events to be counted. In case the counter is significantly overrated, the algorithm may prematurely terminate. Even if a single overrating

happens with low probability, it has to be shown that with high probability the effect of the total number of overratings is low. Furthermore, in case of an overrating the forwarding of the message must be guaranteed. In the non-idle-start regime we can realize this by enforcing uninformed nodes to regularly broadcast their state. This is impossible in the idle-start regime, where nodes (except the broadcaster) are only allowed to broadcast a message after they themselves received a message. If the number of events to be counted is underrated, the termination of the algorithm can be delayed indefinitely, this must be avoided.

The proposed randomized algorithm $\mathcal{A}$ uses two types of messages of size $O(1)$. Message *INF* contains the information to be forwarded to all nodes. Initially the broadcaster locally broadcasts this message. The second message *IDLE* is locally broadcasted in every round by each uninformed node. An informed node that receives an *IDLE* message locally broadcasts for a specified number of rounds – called the *active interval* – the *INF* message. Due to the dynamic character of the network an uninformed node may not receive any of these *INF* messages. Therefore, when an informed node – indicated by variable *informed* – receives an *IDLE* message outside its active interval it increases the length of its active interval and restarts locally broadcasting the *INF* message. We will prove that for networks with bounded connectivity time, with high probability all nodes eventually receive the *INF* message and become silent.

Each node independently maintains a Morris$^+$ counter that is initiated upon the start of algorithm $\mathcal{A}$. Each counter approximately counts the number of locally broadcasted *INF* messages, note that the expected value of the counter equals the actual number of broadcasts. This number is used to control the lengths of the active intervals. The goal is that these lengths form with high probability a geometric sequence, i.e., each term after the first is found by multiplying the previous one by a fixed number, called the common ratio. The length of the active interval is not explicitly stored. Instead only the value of the internal Morris$^+$ counter – indicated by a variable *count* – is stored. When this counter reaches the value of variable max the current active interval ends. Upon receiving a message *IDLE* a new active interval begins, this is achieved by incrementing variable max. Note that the Morris$^+$ counter is never reset. The length of the active interval depends on the current value of variable max. The expected length is $((1+a)^{max}-1)/a$. Thus, the common ratio is roughly $1+a$. The crucial point is that with high probability, the actual length of the active interval is very close to the expected value. We will prove that with high probability, the Morris$^+$ counter uses only $O(\log\log n)$ bits of memory per node before algorithm $\mathcal{A}$ terminates.

Algorithm $\mathcal{A}$ uses one Morris$^+$ counter per node, with $a = \epsilon^2/(8\ln(1/\delta))$, where $\epsilon = 1/4$ and $\delta = 1/n^3$, this requires each node to know n. Algorithm 1 shows the proposed algorithm $\mathcal{A}$. Function $morrisA$ implements the Morris$^+$ as described above with one minor change (see Section 6). Whenever a node locally broadcasts the message *INF* function $morrisA$ is called afterwards. In this call, variable *count* is probabilistically incremented and returned, the original Morris$^+$ counter returns $((1+a)^{count}-1)/a$.

▶ **Definition 5.** *For $t \geq 0$ let $I_t = \{v \in V \mid v.informed_t = true \vee v$ receives a message INF in round $t\}$ and $U_t = V \setminus I_t$.*[1] *An edge $e = (u, v)$ is called* active *in round t if $e \in E(t)$ – i.e., edge e exists in G during round t – $u \in U_t$, and $v \in I_t$, otherwise it is called* passive.

Note that $I_0 = \{v_b\}$, where v_b is the initial broadcasting node. Furthermore, $I_t \subseteq I_{t'}$ and $U_{t'} \subseteq U_t$ for all $t' \geq t$. Once an edge becomes passive, it remains passive forever. For each round t the following holds: If G is 1-interval connected, then as long as $U_t \neq \emptyset$ there exists

[1] If x is an variable of node v, then $v.x_t$ denotes the value of x at the start of round t.

Algorithm 1 Las Vegas algorithm $\mathcal{A}$: Stabilization termination with non-idle start.

```
Initialization do
    count ← 0, max ← 1
    if broadcaster then
        informed ← true
    else
        informed ← false

In each round do
    if informed then
        if received msg(IDLE) and count = max then
            max ← max + 1
        if count < max then
            count ← morrisA(count)
            localBroadcast(INF)
    else
        if received msg(INF) then
            informed ← true
            count ← morrisA(count)
            localBroadcast(INF)
        else
            localBroadcast(IDLE)
```

an active edge $e = (u, v)$ in round t. Thus, if t falls into an active interval of v, then v locally broadcasts a message *INF* that is received by u and hence, u becomes informed in round $t + 1$. If round t does not belong to an active interval of v then we have $v.count_t = v.max_t$. Remember that node u locally broadcasts the message *IDLE* in round t. In round $t + 1$ node v will receive this message and increase $v.max$. Thus, in round $t + 1$ node v enters a new active interval and the expected length of this active interval increases approximately by a factor of $1 + a$. Hence, in every round at least one node becomes informed or the length of an active interval is increased. The idea of the proof of Theorem 1 is that while the lengths of the active intervals of the informed nodes increase, when an edge (u, v) becomes active, the probability that node v is in its active interval also increases. Then also the probability that u gets informed in this round increases. These ideas enable us to prove that with high probability all nodes are informed after $O(m \log m)$ rounds.

We consider a fixed execution $\mathcal{E}$ of algorithm $\mathcal{A}$. For each edge $e = (u, v)$ of G let $t_1^e < t_2^e < \ldots < t_l^e$ be the rounds of $\mathcal{E}$ in which e is active. If e is infinitely often active, then $l = \infty$. If $l < \infty$ then edge e is never active after round t_l^e. In the next lemma, we prove that with high probability, the round numbers in which a fixed edge $e = (u, v)$ is active, grow exponential with the value of $v.max$.

▶ **Lemma 6.** *Let $e = (u, v)$ be an edge and i such that $i + 1 < l$. Then*

$$t_{i+1}^e > \frac{3((1+a)^{v.max_{t_i^e+1}} - 1)}{4a}$$

with probability at least $1 - 2/n^3$.

Proof. Note that $i+1 < l$ implies that t^e_{i+1} is not the last round in which edge e is active. If v receives a message *IDLE* in round t^e_i or if $v.count_{t^e_i} < v.max_{t^e_i}$ then v locally broadcasts in round t^e_i a message *INF*. Node u receives this message in round $t^e_i + 1$, thus $u \in I_{t^e_i+1}$. If u receives in round t^e_i a message *INF*, then also $u \in I_{t^e_i+1}$. Thus, in both cases round t^e_i is the last round in which e is active, i.e., $l = i$. This contradicts the choice of i.

Thus $v.max_{t^e_i} = v.count_{t^e_i}$ and v does neither receive a message *IDLE* nor does u receive a message *INF* in round t^e_i. Let $\hat{N}$ be the number of invocations of the Morris$^+$ counter of v until round t^e_{i+1}. Then $t^e_{i+1} \geq \hat{N}$. If $\hat{N} \leq 1/a = 2^7 \ln(1/\delta) = 2^9 \log n$ then Morris$^+$ counter as described above counts exactly, i.e., $\hat{N} \geq (((1+a)^{v.max_{t^e_i}+1} - 1))/a$ and the statement follows. So we can assume that $\hat{N} > 1/a$.

Clearly, u locally broadcasts a message *IDLE* in round t^e_i. Node v receives this message in round $t^e_i + 1$ and increments variable *max*, i.e., $v.max_{t^e_i+2} = v.max_{t^e_i} + 1$. Then, as long as $v.count < v.max_{t^e_i} + 1$ node v locally broadcasts message *INF*. If $v.count_{t^e_{i+1}} < v.max_{t^e_i} + 1$ in round t^e_{i+1} holds, then $u \in I_{t^e_{i+1}+1}$. Thus, in this case round t^e_{i+1} is the last round in which e is active. This again contradicts the choice of i. Hence, v has stopped locally broadcasting message *INF* before round t^e_{i+1}. Then Lemma 8 implies that

$$t^e_{i+1} \geq \hat{N} \geq \frac{3((1+a)^{v.max_{t^e_i}+1} - 1)}{4a}$$

with probability at least $1 - 2/n^3$. ◀

In the following lemma, we prove that between each pair of rounds t^e_i, t^e_{i+1} the value of the counter $v.max$ is incremented at least once.

▶ **Lemma 7.** *For $0 \leq i < l$ we have $v.max_{t^e_i} \geq i$.*

Proof. The proof is by induction on i. The case $i = 1$ trivially holds. Let $i > 1$. Assume that $t^e_i = t^e_{i-1} + 1$. This yields $u \in I_{t^e_{i-1}+2}$, hence $t^e_i = t^e_l$, a contradiction. Thus, $t^e_i \geq t^e_{i-1} + 2$. Since t^e_i exists we have $u \in U_{t^e_i}$. By induction assumption we have $v.max_{t^e_{i-1}} \geq i - 1$. In round $t^e_{i-1} + 1$ node u does not receive a message *INF*, because this would imply $u \in I_{t^e_{i-1}+1} \cap U_{t^e_i} = \emptyset$. Hence, v does not locally broadcast a message *INF* in round t^e_{i-1}. This yields $v.count_{t^e_{i-1}} = v.max_{t^e_{i-1}}$. Node u locally broadcasts a message *IDLE* in round t^e_{i-1}. Thus, v receives at the latest in round $t^e_{i-1} + 1$ a message *IDLE* and increments $v.max$. Hence, $v.max_{t^e_i} \geq v.max_{t^e_{i-1}+2} \geq v.max_{t^e_{i-1}} + 1 \geq i - 1 + 1 = i$. ◀

With these preparatory works, we can prove the main result about algorithm $\mathcal{A}$ as outlined above.

▶ **Theorem 1.** *There exists a non-idle-start algorithm $\mathcal{A}$ solving the broadcast problem with stabilizing termination with high probability for anonymous, synchronous, dynamic 1-interval-connected networks provided each node knows n. With high probability $\mathcal{A}$ terminates in time $O(m \log n)$, requires $O(\log \log n)$ memory per node, and uses messages of size $O(1)$.*

Proof. To prove that $\mathcal{A}$ solves the broadcast problem with stabilizing termination with high probability, it suffices to prove that with high probability, the sequence $t^e_1, t^e_2, \ldots, t^e_l$ is finite for each edge e of G. There are two reasons for the sequence of an edge $e = (u, v)$ to be finite: Either u becomes informed, because it received a message *INF* from v in round t^e_l or from a neighbor $w \neq v$ in or after round t^e_l, or edge e never belongs to G after round t^e_l. Thus, if there exists T, such that $t^e_l < T$ for all edges e, then all nodes are informed in round T.

Let e be a fixed edge and i such that $i+1 < l$. Then Lemma 6 implies that with probability at least $1-2/n^3$ we have $t^e_{i+1} > 3((1+a)^{v.max_{t^e_i}+1}-1)/(4a)$ and therefore by Lemma 7

$$i \le v.max_{t^e_i} < \frac{\log_2((4/3)at^e_{i+1}+1)}{\log_2(1+a)} - 1.$$

For $e \in E$ and $T > 0$ define $f(e,T) = max\{i \mid t^e_i \le T\}$. Thus, if $f(e,T)+1 < l$ then with probability at least $1-2/n^3$

$$f(e,T) \le \frac{\log_2((4/3)aT+1)}{\log_2(1+a)} - 1.$$

Similarly, if $f(e,T)+1 = l$

$$f(e,T) \le \frac{\log_2((4/3)aT+1)}{\log_2(1+a)} \tag{1}$$

with probability at least $1-2/n^3$. Note that Eq. (1) also holds if $l = 1$, i.e., if edge e is only active once.

For $e \in E$ let A_e be the event that Eq. (1) is false. Hence. $\mathbf{Pr}[A_e] \le 2/n^3$. The union bound implies that $\mathbf{Pr}\left[\bigvee_{e\in E} A_e\right] \le 1/n$. Hence, with probability at least $1-1/n$ Eq. (1) is is true for all edges $e \in E$. Up to round T each edge can be at most $\log_2(2aT+1)/\log_2(1+a)$ times be active. Since there are at most m edges, there are at most $m\log_2(2aT+1)/\log_2(1+a)$ active edges up to time T. Since, in each round at least one edge must be active, Eq. (1) leads to

$$T \le \sum_{e\in E} f(e,T) \le m\left(\frac{\log((4/3)aT+1)}{\log(1+a)}\right) \tag{2}$$

with probability at least $1-1/n$. Since the left side grows linearly in T, while the right side grows only linearly in $\ln T$, the value of T cannot be too large. Thus, there exists a round $\hat{T}$, during which all nodes are informed. What is the value of $\hat{T}$?

We consider the equality $\hat{T} = m\log\Big((4/3)a\hat{T}+1\Big)/\log(1+a)$ and assume $\hat{T} = 2^L$. Thus, for large values of $\hat{T}$ we have $2^L = m(\log(4a/3)+L)/\log(1+a)$. We apply Lemma 9 with $k = m/\log(1+a)$, $c_1 = 1$ and $c_2 = \log 4a/3$, this yields $O(L) \in O(\log m)+O(\log\log m)+o(1)$. Therefore, we have $\hat{T} \in O(m\log m)$. Hence, all nodes are within $O(m\log n)$ rounds with probability at least $1-1/n$ informed.

It remains to prove, that with high probability after $O(m\log n)$ no more messages are sent. Let v be an informed node and $t_f = \max\{t^e_l \mid e = (v,u), u \in N(v)\}$. Then $v.count_{t_f} \le v.max_{t_f}$. Let N_{t_f} be the number of invocations of the Morris$^+$ counter of node v until the round, in which $v.count$ reached the value $v.max_{t_f}-1$. Clearly, in that round, not all neighbors of v were already informed, because in this case $v.max$ would not be incremented again. This is true, because an increment of $v.max$ only happens after v received a message *IDLE*. As shown above, with probability at least $1-2/n^3$ there exists C with $Cm\log m \ge N_{t_f}$. Now Lemma 8 yields

$$Cm\log m \ge N_{t_f} \ge (1-\epsilon)\frac{(1+a)^{v.max_{t_f}}-1}{a}$$

with probability at least $1-2/n^3$. This yields

$$(1+a)^{v.max_{t_f}} \le 1 + \frac{aCm\log m}{1-\epsilon}.$$

In round t_f a neighbor of v locally broadcasts a message *IDLE* that is received by v in round $t_f + 1$. Furthermore, after round $t_f + 1$, node v will never receive a message *IDLE* again. Thus, either $v.max_s = v.max_{t_f}$ for all $s \geq t_f + 1$, or if v receives a message *IDLE* in round $t_f + 1$ and $v.count_{t_f+1} = v.max_{t_f}$. In the later case we have $v.max_s = v.max_{t_f} + 1$ for all $s \geq t_f + 1$. Anyway, in the worst case, node v continues to locally broadcast a message *INF* until the first round t' with $v.count_{t'} = v.max_{t_f} + 1$.

Let N_v be the number of invocations of the Morris$^+$ counter of node v until $v.count_{t'} = v.max_{t_f} + 1$. By Lemma 8 we have

$$(1+\epsilon)\,\frac{(1+a)^{v.max_{t_f}+2}-1}{a} \geq N_v$$

with probability at least $1 - 2/n^3$. Thus

$$(1+\epsilon)(1+a)^2\left(\frac{1}{a} + \frac{Cm\log m}{1-\epsilon}\right) \geq N_v.$$

Since $(1+a)^2 \leq 4$ and $1/a = (24/\epsilon^2)\log n$, node v does no send any messages after $O(m \log m)$ rounds with probability $1 - 2/n^3$. The union bound implies that with probability at least $1 - 1/n$ algorithm $\mathcal{A}$ is silent after $O(m \log n)$ rounds. Thus, with high probability, the number of invocations of the Morris$^+$ counter of each node is in $O(m \log n)$. By Lemma 8 the required memory is in $O(\log \log n)$. ◀

Theorem 1 can be extended to bounded connectivity time. Assume that the connectivity time θ of G is larger than 1. Then during every interval of length θ an edge appears in any cut of G. Thus, if during θ consecutive rounds there is no active edge, then all nodes are informed. Hence, the left side of Eq. (2) becomes T/θ. The additional factor $1/\theta$ does not result in any significant changes, only the time until termination is affected. The argument following Eq. (2) can be adopted with a minor change: Instead of m, we must use θm. This yields that with high probability algorithm $\mathcal{A}$ terminates in $O(\theta m \log(\theta m))$ rounds requiring $O(\log \log(\theta m))$ bits of memory per node.

It remains to prove Lemma 8 to 9. Lemma 8 is implicitly contained in [16]. We include it to make the paper self-contained.

▶ **Lemma 8** (Theorem 1.2, [16]). *Let $\epsilon \in (0, 1/2)$, $\delta \leq 1/n^3$, and $a = \epsilon^2/(8\ln(1/\delta))$. If $\hat{N} > 1/a$ is the number of invocations of the Morris(a) counter until the internal variable count reaches the value max, then with probability at least $1 - 2e^{-\epsilon^2/8a}$ the Morris(a) counter uses $\log\log\hat{N} + 2\log(1/\epsilon) + \log\log(1/\delta) + O(1)$ bits and $\hat{N}$ satisfies*

$$(1+\epsilon)\,\frac{(1+a)^{max+1}-1}{a} \geq \hat{N} \geq (1-\epsilon)\,\frac{(1+a)^{max+1}-1}{a}.$$

Proof. Let Z_i be the random variable denoting the number of increments it takes for *count* to increase from i to $i+1$. Then

$$\sum_{i=0}^{max} Z_i = \hat{N}.$$

Since when $count = i$, each increment causes *count* to increase with probability $p_i = (1+a)^{-i}$, Z_i follows the geometric distribution

$$\mathbf{Pr}\left[Z_i = s\right] = (1-p_i)^{s-1}p_i.$$

As shown in Section 2.2. of [16] we have

$$\mathbf{Pr}\left[\sum_{i=0}^{max} Z_i \leq (1-\epsilon)\sum_{i=0}^{max}\frac{1}{p_i}\right] \leq e^{-\epsilon^2/8a}.$$

Therefore, with probability at least $1-e^{-\epsilon^2/8a}$, we have

$$\hat{N} \geq (1-\epsilon)\frac{(1+a)^{max+1}-1}{a}.$$

Similarly, we have with probability at least $1-e^{-\epsilon^2/8a}$

$$(1+\epsilon)\frac{(1+a)^{max+1}-1}{a} \geq \hat{N}.$$

The result follows from the union bound. ◀

▶ **Lemma 9.** *The solution of the equality* $2^L = k(c_1 L + c_2)$ *with* $c_1, c_2 > 0$ *constants and* $k > 1$ *satisfies* $L \in O(\log_2 k) + O(\log_2 \log_2 k) + o(1)$.

Proof. Set $a = \ln 2$, then $a > 0$. We will reshape the equation $2^L = k(c_1 L + c_2)$. Let $z = c_1 L + c_2$, then $e^{aL} = kz$ and $L = z/c_1 - c_2/c_1$. Thus, $kz = e^{az/c_1}2^{-c_2/c_1}$ and $kze^{-az/c_1} = 2^{-c_2/c_1}$ and finally

$$\frac{-a}{c_1}ze^{-az/c_1} = \frac{-a}{kc_1}2^{-c_2/c_1}.$$

Let $w = -az/c_1$, then the last equation becomes $we^w = -a\frac{2^{-c_2/c_1}}{kc_1}$. Note that $-a\frac{2^{-c_2/c_1}}{kc_1} < 0$. We make use of the Lambert-W function [6], the inverse of we^w: $w = W\left(-a\frac{2^{-c_2}}{k}\right)$ resp.,

$$L = z/c_1 - c_2/c_1 = -w/a - c_2/c_1 = -\frac{1}{\ln 2}W\left(-a\frac{2^{-c_2}}{k}\right) - c_2/c_1.$$

The Lambert function has two branches W_0 and W_{-1}. Only the branch W_{-1} provides solutions that are relevant for larger values of k. For small values θ (i.e., $\theta \longrightarrow 0^+$) the solution satisfies $W_{-1}(-\theta) = \ln(\theta) - \ln(-\ln(\theta)) + o(1)$ [6]). Thus, this holds for $\theta = a\frac{2^{-c_2}}{k} \in O(1/k)$. Hence,

$$L = -\frac{1}{\ln 2}\left(\ln\left(a\frac{2^{-c_2}}{k}\right) - \ln\left(-\ln\left(a\frac{2^{-c_2}}{k}\right)\right) + o(1)\right) - c_2/c_1.$$

This implies $L = O(\log_2 k) + O(\log_2 \log_2 k)$. Hence, $L \in O(\log_2 k)$. ◀

4.3 Open Problems

We conjecture that algorithm $\mathcal{A}$ terminates in $O(n \log n)$ rounds. In its current implementation variable max is incremented at the beginning of every active interval. An alternative to be explored is to increment max by $\ln(1-1/(1+a)^{max})/\ln(1+a)$, this would double the length of the active interval. Also, we believe that the space complexity of $O(\log n)$ for algorithm *Countdown* of [17] can be reduced with high probability by using probabilistic counting. It is questionable whether this can be done with messages of size $O(1)$.

Kowalski and Mosteiro study the all-to-all communication problem, where each node has an input to be delivered to all other nodes [12] for anonymous, dynamic networks with limited message size, and logarithmic internal memory per node. They prove that this task

can be done in time polynomial in the number of nodes and a lower bound on a metric measuring the dynamically evolving graphs that represent the network topology. An open question is whether the usage of an approximate counting algorithm – as done in this work – can reduce the demands for internal memory to $O(\log \log n)$.

The recent advent of programmable switches opens a new facet of distributed algorithms, here storage limitations become a major issue. Basat et al. introduced the μ-CONGEST model, where on top of bandwidth restrictions, the storage space on nodes is also limited to μ words, in line with many real-world systems [1]. We believe that approximate counting algorithms, e.g. the Morris$^+$ counter, will lead to new efficient algorithms for this new model.

5 Impossibility Results

Parzych et al. considered broadcast with termination detection and proved that this is impossible for deterministic idle-start algorithms, where only the broadcaster can initially send messages [17]. In the same work they also proved that even without an idle start this task cannot be achieved by a deterministic algorithm, provided that nodes have no knowledge of the number of broadcasters. In the following we extend both results to the realm of randomized algorithms.

A Monte Carlo algorithm is a randomized algorithm that may produce incorrect results, but with bounded error probability. To prove the non-existence of a Monte Carlo algorithm for the broadcast with termination detection problem, we do a proof by contradiction. We assume there exists an idle-start Monte Carlo algorithm $\mathcal{A}$ that solves this problem. For this algorithm we construct a time-varying graph G_α and show that independently of the random decisions $\mathcal{A}$ makes, the broadcaster will announce termination before all nodes are informed. Hence, any execution of $\mathcal{A}$ on G_α is incorrect, a contradiction. A Las Vegas algorithm is a randomized algorithm that always produces correct results, the running time of different executes varies. To prove the non-existence of such an algorithm for the given problem, we will use the same graph G_α and show that there exists at least one incorrect execution.

▶ **Theorem 2.** *No idle-start Monte Carlo or Las Vegas algorithm can solve broadcast with termination detection for anonymous, synchronous, 1-interval connected dynamic graphs.*

Proof. a) Assume there exists an idle-start Monte Carlo algorithm $\mathcal{A}$ that solves broadcast with termination detection for anonymous, synchronous, 1-interval connected dynamic graphs within $T_{\mathcal{A}}(n)$ rounds with probability $p_{\mathcal{A}}(n) > 0$. Let G be a fixed static, connected graph with n_0 nodes, and α a constant such that each execution of $\mathcal{A}$ terminates for G with b as the broadcaster after at most α rounds. Hence, after at most α rounds b irrevocably declares the broadcast to be complete. The probability that after this round all nodes are informed is at least $p_{\mathcal{A}}(n_0)$. We execute $\mathcal{A}$ on a time-varying graph G_α as described in [17]. G_α consists of two static components, i.e., all edges of these two components remain present throughout the lifetime of G_α. The first static component is the graph G and the second is a path $P = v_0, v_1, \ldots, v_{\alpha+1}$. In each round t there is a single edge (b, v_t) connecting the broadcaster b to a node in P, thus, G_α is 1-interval connected (see Fig. 1). Since $\mathcal{A}$ is an idle-start algorithm, the construction of G_α ensures that no node of P ever sends a message to b or to any other node of G during the first α rounds.

Let $\mathcal{E}_\alpha$ be a fixed execution of $\mathcal{A}$ on G_α. The actions of $\mathcal{A}$ that amount to the state of each node of G_α at the end of round t are determined by the state of the nodes of G_α in $\mathcal{E}_\alpha$ at the beginning of round t and the sequence $\mathcal{S}_v$ of random bits that each node v uses as an auxiliary input to determine its state changes and the messages to be sent. Denote by $\mathcal{E}_G$

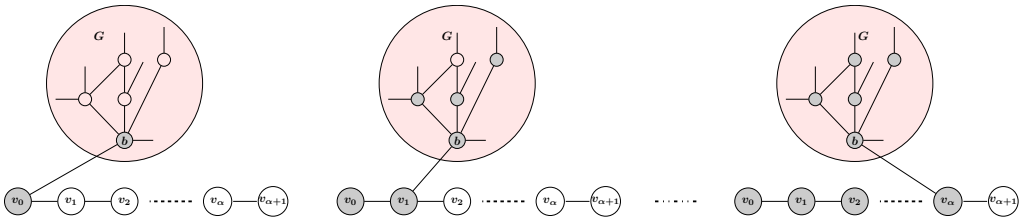

Figure 1 Snapshots of G_α after rounds $0, 1$, and α; gray nodes are informed, figure based on [17].

the execution of $\mathcal{A}$ on G, where each node v uses S_v as auxiliary input. We will prove that for every round t the restriction of $\mathcal{E}_\alpha$ to G coincides with $\mathcal{E}_G$. This requires to prove that at the beginning of each round t the nodes of G are in the same state in both executions. This is done by induction on t.

Let $t = 0$. Initially all nodes of G have the same state in both executions $\mathcal{E}_\alpha$ and $\mathcal{E}_G$. Since $\mathcal{A}$ is an idle-start algorithm only broadcaster b broadcasts a message and changes its state in round 0. According to our assumption, b has no knowledge of the graph nor its neighborhood. Thus, since b has identical states in both executions and uses the same random bits as auxiliary input, b will broadcast an identical message in $\mathcal{E}_\alpha$ and $\mathcal{E}_G$ in round 0. Therefore, all nodes of G have at the beginning of round 1 identical states in $\mathcal{E}_\alpha$ and $\mathcal{E}_G$.

Let $t > 0$. By induction hypothesis, all nodes in G have identical states at the start of round t in $\mathcal{E}_\alpha$ and $\mathcal{E}_G$. Thus, since the nodes use the same random bits in both executions, they also send the same messages in round t. Since by construction no node of P sends a message to b or any other node of G in round t, we conclude that at the beginning round $t+1$ all nodes of G have the same state in $\mathcal{E}_\alpha$ and $\mathcal{E}_G$. Hence, $\mathcal{E}_\alpha$ and $\mathcal{E}_G$ are identical on G and thus, $\mathcal{E}_G$ is a legal execution of $\mathcal{A}$ on G. This implies that after at most α rounds broadcaster b will announces termination in execution $\mathcal{E}_G$ and thus, also in $\mathcal{E}_\alpha$. The construction of G_α entails that node $v_{\alpha+1}$ cannot be reached in α rounds from b. Thus, in round α node $v_{\alpha+1}$ is still uninformed. Hence, any execution of $\mathcal{A}$ on G_α is incorrect, a contradiction.

b) Assume there exists an idle-start Las Vegas algorithm $\mathcal{A}$ that correctly solves broadcast with termination detection for anonymous, synchronous, 1-interval connected dynamic graphs within expected $T_{\mathcal{A}}(n)$ rounds. Let G be a static, connected graph with n_0 nodes and b the broadcaster of G. Let $X_{\mathcal{E}}$ be a random variable that denotes the runtime of execution $\mathcal{E}$ of algorithm $\mathcal{A}$ on G. Then $\mathbf{Ex}[X_{\mathcal{E}}] = T_{\mathcal{A}}(n_0)$. Thus, there is an execution $\mathcal{E}^*$ of $\mathcal{A}$ on G with runtime $X_{\mathcal{E}^*} \leq T_{\mathcal{A}}(n_0)$. As in a) we construct a graph $G_{X_{\mathcal{E}^*}}$ and show that execution $\mathcal{E}^*$ implies an execution of $\mathcal{A}$ on $G_{\mathcal{E}^*}$ that is incorrect. This contradicts the fact, that $\mathcal{A}$ is a Las Vegas algorithm, i.e., it is always correct. ◀

Inspired by the impossibility results of [7] about systems with multiple leaders the authors of [17] consider systems with several broadcasters. Termination detection in this case means that every broadcaster correctly and irrevocably decides that the broadcast is complete. They proved that even without an idle start, no deterministic algorithm can solve broadcast with termination detection if nodes have no knowledge of the number of broadcasters. In the following we prove that there is also no randomized algorithm solving this problem.

The overall idea of the proof is similar to the above proof, but the construction of the graph is more involved. To describe the execution of a randomized algorithm each node uses a sequence of random bits as auxiliary input. Thus, the execution of the randomized algorithm is determined by the auxiliary inputs for each node. First, we prove that no non-idle-start Las Vegas algorithm can solve the broadcast problem with termination detection. We construct

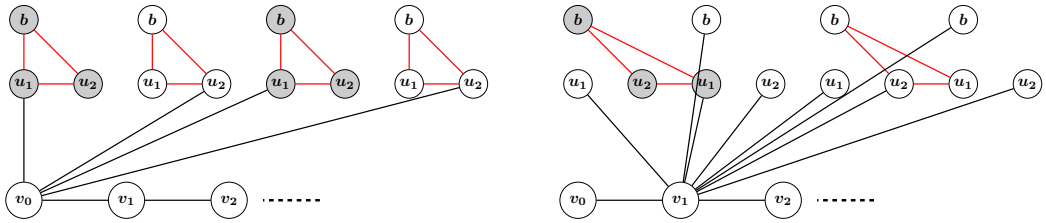

Figure 2 The first two snapshots of G_α. The edges of the complete graphs are depicted in red and nodes belong to X_1 (resp. X_2) are depicted in gray.

a 1-interval connected dynamic graph G_δ for which the assumed Las Vegas algorithm $\mathcal{A}$ fails. The construction is done in a graduate style, edges in round t are based on the execution of $\mathcal{A}$ during the first $t-1$ rounds. The construction is similar to that of [17] but requires considerable changes. The proof for the non-existence of a Monte Carlo algorithm is similar.

▶ **Theorem 3.** *No non-idle-start Monte Carlo or Las Vegas algorithm can solve broadcast with termination detection for anonymous, synchronous, 1-interval connected dynamic graphs if nodes are unaware of the number of broadcasters.*

Proof. Assume there exists a Las Vegas algorithm $\mathcal{A}$ that is unaware of the number of broadcasters and correctly solves broadcast with termination detection for anonymous, synchronous, 1-interval connected dynamic graphs within expected $T_{\mathcal{A}}(n)$ rounds. Each node v uses the sequence S_v of random bits as auxiliary input. Thus, the execution of $\mathcal{A}$ is determined by the auxiliary inputs S_v for each $v \in V$. Let $X_{\mathcal{E}}$ be a random variable that denotes the runtime of an execution $\mathcal{E}$ of $\mathcal{A}$ on a graph with n nodes. Then $\mathbf{Ex}[X_{\mathcal{E}}] = T_{\mathcal{A}}(n)$.

Let G be the complete graph G with three nodes b, u_1, and u_2. Node b is the single broadcaster of G. Let $\alpha = \lceil T_{\mathcal{A}}(3) \rceil$. We consider a fixed execution of $\mathcal{A}$ for G. The auxiliary inputs are denoted by S_b for node b and S_1 (resp. S_2) for node u_1 (resp. u_2). We will prove that this execution will not announce termination at or before round α. This holds for every execution of $\mathcal{A}$ for G, i.e., for all auxiliary inputs for the three nodes of G. This yields $\mathbf{Ex}[X_{\mathcal{E}}] > \alpha$, a contradiction.

Consider a fixed execution of $\mathcal{A}$ for G with auxiliary inputs S_b, S_1, and S_2. Denote the configurations of G for this execution by $C_0, C_1, \ldots$. We will prove that this execution requires more than α rounds. To do so, we will construct a 1-interval connected dynamic graph G_δ. Let $\delta = 2^\alpha$. The graph G_δ has $3\delta + \alpha + 1$ nodes. Denote the first $\alpha + 1$ nodes by $v_0, \ldots, v_\alpha$ and the remaining 3δ nodes by W. The nodes $v_0, \ldots, v_\alpha$ form a path P and the α edges of P remain present throughout the lifetime of G_δ. The remaining edges of G_δ are defined gradually for each time t such that G_δ is 1-interval connected for all t.

First consider the case $t = 0$. The 3δ nodes of W are partitioned into 3 groups W_b, W_1, and W_2 each containing δ nodes. We assign the same auxiliary input to the nodes of each group: nodes of W_b use S_b and those of W_1 (resp. W_2) use S_1 (resp. S_2). Also each node of W_b is a broadcaster. The nodes of W form δ complete graphs with three nodes each, one from each set W_b, W_1, and W_2. Each of these complete graphs is in configuration C_0. We split these δ complete graphs into two groups X_1 and Y_1 of equal size. Finally, every node of W_1 that belongs to a graph of X_1 and every node of W_2 that belongs to a graph of Y_1 is connected to v_0 (see Fig. 2). Thus, for $t = 0$ graph G_δ has $4\delta + \alpha$ edges and δ broadcasters.

To define G_δ for $t = 1$ we consider a first round of execution of $\mathcal{A}$ on G_δ based on the assigned auxiliary inputs. Note that during this round each node from W_b can receive a message only from the two nodes in W_1 and W_2 that belong to the same complete graph.

Furthermore, each node of W_2 (resp. W_1) that belongs to a complete graph of X_1 (resp. Y_1) can receive a message only from the two nodes in W_b and W_1 (resp. W_b and W_2) that belong to the same complete graph. Thus, the nodes from W_b and W_2 that belong to a graph from X_1 are in the same state as b and u_2 in C_1. Also, the nodes from W_b and W_1 that belong to a graph from Y_1 are in the same state as b and u_1 in C_1.

Now we can define the edges of G_δ for $t = 1$. We again form complete graphs of order three. Each contains a node of W_b, a node of W_2 that belongs to a graph from X_1 and a node of W_1 that belongs to a graph from Y_1. Thus, there are $\delta/2$ complete graphs of order three, each is in configuration C_1. They are split into into two groups X_2 and Y_2. Finally, every node of W_1 that belongs to a graph of X_2 and every node of W_2 that belongs to a graph of Y_2 is connected to v_1. All nodes that do not belong to one of the $\delta/2$ complete graphs are also connected by an edge to v_1 (see Fig. 2).

We continue to define the edges of G_δ for $t > 1$ in this way. The number of broadcasters is decreased in every round. The assumption that the number of broadcasters is unknown to the nodes implies that changing that number has no influence on the behavior of $\mathcal{A}$ on nodes (except for the initial broadcast). We execute $\mathcal{A}$ for one more round on G_δ using the fixed auxiliary input. Continuing in this way, we can create $2^{\alpha-i}$ copies of configuration C_i for each $i = 0, \ldots, \alpha$ while only informing a single node in the path P at a time. Thus, after α rounds we still have $2^{\alpha-\alpha} = 1$ copies of G in G_δ, but node v_α of G_α is not yet informed. Thus, if the broadcaster of this remaining complete graph would announce termination, algorithm $\mathcal{A}$ would be incorrect. Thus, for the given auxiliary input algorithm $\mathcal{A}$ does not announce termination within the first α rounds. This completes the proof for a Las Vegas algorithm.

The proof for the non-existence of a Monte Carlo algorithm is similar. Assume there exists an idle-start Monte Carlo algorithm $\mathcal{A}$ that solves the given broadcast with termination detection within $T_\mathcal{A}(n)$ rounds with probability $p_\mathcal{A}(n) > 0$. We set $\alpha = T_\mathcal{A}(3)$ and repeat the above construction of G_α. This shows that for every auxiliary input algorithm $\mathcal{A}$ is incorrect for a complete graph with three nodes. This is a contradiction since $p_\mathcal{A}(n) > 0$. ◀

5.1 Open Problems

We believe that with the techniques used above, some of the impossibility results for broadcast with termination detection presented in [4] can be extended to the realm of randomized algorithms. Parzych et al. proved in [17] lower bounds for the space complexity for deterministic broadcast with termination detection and stabilizing termination for anonymous dynamic networks. It is an open question, whether randomized algorithms can come close to these bounds. The best known space complexity for broadcast with termination detection follows from Di Luna and Viglietta's history trees algorithm which uses $O(n^3 \log n)$ memory in the worst case [8]. We conjecture that randomized algorithms can come closer to the currently best known lower bound of $\Omega(\log n)$ [17].

6 Implementation of the Morris$^+$ Counter

To implement the statement *With probability* $(1+a)^{-count}$ we do *count* Bernoulli experiments each succeeding with probability $(1 + a)^{-1}$. If all experiments have a positive outcome then we increment *count*.

Algorithm 2 Implementation of function *morrisA*.

```
int count_sep ← 0
function morrisA(int : count): count
    if count_sep < N_δ then
        count_sep ← count_sep + 1
        if count_sep ≥ ((1 + a)^(count+1) − 1)/a then
            count ← count + 1
    else
        With probability (1 + a)^(−count) do
            count ← count + 1
    return count
```

References

1 Ran Ben Basat, Keren Censor-Hillel, Yi-Jun Chang, Wenchen Han, Dean Leitersdorf, and Gregory Schwartzman. Bounded memory in distributed networks. In *Proc. of the 37th ACM Symposium on Parallelism in Algorithms and Architectures*, pages 566–581, 2025. doi:10.1145/3694906.3743302.

2 Lucas Boczkowski, Amos Korman, and Emanuele Natale. Minimizing message size in stochastic communication patterns: Fast self-stabilizing protocols with 3 bits. In *Proc. 2017 Annual ACM-SIAM Symp. on Discrete Algo. (SODA)*, pages 2540–2559, 2017. doi:10.1137/1.9781611974782.168.

3 Arnaud Casteigts, Paola Flocchini, Bernard Mans, and Nicola Santoro. Deterministic computations in time-varying graphs: Broadcasting under unstructured mobility. In *IFIP Int. Conference on Theoretical Computer Science*, pages 111–124. Springer, 2010. doi:10.1007/978-3-642-15240-5_9.

4 Arnaud Casteigts, Paola Flocchini, Bernard Mans, and Nicola Santoro. Shortest, fastest, and foremost broadcast in dynamic networks. *Int. Journal of Foundations of Computer Science*, 26(4):499–522, 2015. doi:10.1142/S0129054115500288.

5 Arnaud Casteigts, Paola Flocchini, Walter Quattrociocchi, and Nicola Santoro. Time-varying graphs and dynamic networks. *Int. Journal of Parallel, Emergent and Distributed Systems*, 27(5):387–408, 2012. doi:10.1080/17445760.2012.668546.

6 Robert M. Corless, Gaston H. Gonnet, David E. G. Hare, David J. Jeffrey, and Donald E. Knuth. On the Lambert W Function. *Advances in Computational Mathematics*, 5(1):329–359, 1996. doi:10.1007/BF02124750.

7 Giuseppe A Di Luna and Giovanni Viglietta. Optimal computation in leaderless and multi-leader disconnected anonymous dynamic networks. In *37th Int. Symp. on Distr. Comp. (DISC)*, pages 18–1. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2023. doi:10.4230/LIPIcs.DISC.2023.18.

8 Giuseppe A Di Luna and Giovanni Viglietta. Efficient computation in congested anonymous dynamic networks. *Distributed Computing*, pages 1–18, 2025. doi:10.1007/s00446-025-00481-z.

9 Philippe Flajolet. Approximate counting: a detailed analysis. *BIT Numerical Mathematics*, 25(1):113–134, 1985. doi:10.1007/BF01934993.

10 Walter Hussak and Amitabh Trehan. Termination of amnesiac flooding. *Distributed Computing*, 36(2):193–207, 2023. doi:10.1007/S00446-023-00448-Y.

11 Amos Korman and Robin Vacus. Early adapting to trends: self-stabilizing information spread using passive communication. *Distributed Computing*, 37(4):335–362, 2024. doi:10.1007/S00446-024-00462-8.

12 Dariusz R. Kowalski and Miguel A. Mosteiro. Anonymous adversarial dynamic networks with logarithmic memory and communication. *Theoretical Computer Science*, 1066:115740, 2026. doi:10.1016/j.tcs.2025.115740.

13 Fabian Kuhn, Nancy A. Lynch, and Rotem Oshman. Distributed computation in dynamic networks. In Leonard J. Schulman, editor, *Proc. 42nd ACM Symp. on Theory of Computing, STOC*, pages 513–522. ACM, 2010. doi:10.1145/1806689.1806760.

14 Othon Michail, Ioannis Chatzigiannakis, and Paul G Spirakis. Causality, influence, and computation in possibly disconnected synchronous dynamic networks. *Journal of Parallel and Distributed Computing*, 74(1):2016–2026, 2014. doi:10.1016/J.JPDC.2013.07.007.

15 Robert Morris. Counting large numbers of events in small registers. *Communications of the ACM*, 21(10):840–842, 1978. doi:10.1145/359619.359627.

16 Jelani Nelson and Huacheng Yu. Optimal bounds for approximate counting. In *Proc. of the 41st ACM SIGMOD-SIGACT-SIGAI Symp. on Principles of Database Systems*, pages 119–127, 2022. doi:10.1145/3517804.3526225.

17 Garrett Parzych and Joshua J. Daymude. Memory lower bounds and impossibility results for anonymous dynamic broadcast. In Dan Alistarh, editor, *38th Int. Symp. on Distr. Comp. DISC*, volume 319 of *LIPIcs*, pages 35:1–35:18, 2024. doi:10.4230/LIPIcs.DISC.2024.35.

18 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. IEEE, 2014. doi:10.1109/AINA.2014.115.

19 Volker Turau. Synchronous concurrent broadcasts for intermittent channels with bounded capacities. In Tomasz Jurdzinski and Stefan Schmid, editors, *Struc. Inf. and Com. Complexity – 28th SIROCCO*, volume 12810 of *LNCS*, pages 296–312. Springer, 2021. doi:10.1007/978-3-030-79527-6_17.