Gossip in a Smartphone Peer-to-Peer Network

Calvin Newport
Georgetown University
Washington, DC, USA
cnewport@cs.georgetown.edu

ABSTRACT

In this paper, we study the fundamental problem of gossip in the *mobile telephone model*: a recently introduced variation of the classical *telephone model* modified to better describe the local peer-to-peer communication services implemented in many popular smartphone operating systems. In more detail, the mobile telephone model differs from the classical telephone model in three ways: (1) each device can participate in at most one connection per round; (2) the network topology can undergo a parameterized rate of change; and (3) devices can advertise a parameterized number of bits about their state to their neighbors in each round before connection attempts are initiated. We begin by describing and analyzing new randomized gossip algorithms in this model under the harsh assumption of a network topology that can change completely in every round. We prove a significant time complexity gap between the case where nodes can advertise 0 bits to their neighbors in each round, and the case where nodes can advertise 1 bit. For the latter assumption, we present two solutions: the first depends on a shared randomness source, while the second eliminates this assumption using a pseudorandomness generator we prove to exist with a novel generalization of a classical result from the study of two-party communication complexity. We then turn our attention to the easier case where the topology graph is stable, and describe and analyze a new gossip algorithm that provides a substantial performance improvement for many parameters. We conclude by studying a relaxed version of gossip in which it is only necessary for nodes to each learn a specified fraction of the messages in the system. We prove that our existing algorithms for dynamic network topologies and a single advertising bit solve this relaxed version up to a polynomial factor faster (in network size) for many parameters. These are the first known gossip results for the mobile telephone model, and they significantly expand our understanding of how to communicate and coordinate in this increasingly relevant setting.

KEYWORDS

gossip; mobile telephone model; smartphone; peer-to-peer

PODC'17, , July 25–27, 2017, Washington, DC, USA.

ACM ISBN 978-1-4503-4992-5/17/07…$15.00.
http://dx.doi.org/10.1145/3087801.3087813

1 INTRODUCTION

This paper describes and analyzes new gossip algorithms in the mobile telephone model: an abstraction that captures the local device-to-device communication capabilities available in most smartphone operating systems; e.g., as implemented by services such as Bluetooth LE [16], WiFi Direct [2], and Apple's Multipeer Connectivity framework [20].

Motivation. Smartphones are a ubiquitous communication platform: there are currently over 3.9 billion smartphone subscriptions worldwide [18]. Most smartphone communication leverages one-hop radio links to cell towers or WiFi access points. In recent years, however, the major smartphone operating systems have included increasingly stable and useful support for local peer-to-peer communication that allows a device to talk directly to a nearby device (using local radio broadcast) while avoiding cellular and WiFi infrastructure.

The ability to create these local links, combined with the ubiquity of smartphones, enables scenarios in which large groups of nearby smartphone users run applications that create peer-to-peer meshes supporting infrastructure-free networking. There are many possible motivations for these smartphone peer-to-peer networks. For example, they can support communication in settings where network infrastructure is *censored* (e.g., government protests), *overwhelmed* (e.g., a large festival or march), or *unavailable* (e.g., after a disaster or at a remote event). In addition, in developing countries, cellular data minutes are often bought in blocks and carefully conserved—increasing interest in networking operations that do not require cellular infrastructure.

To further validate the potential usefulness of smartphone peer-to-peer networks, consider the FireChat application, which implements group chat using smartphone peer-to-peer services. In the few years since its initial release, it has been widely adopted in over 120 countries and has been used successfully in multiple government protests, festivals (e.g., at Burning Man, which is held far from cell towers), and disaster scenarios [9].

Developing useful applications for this smartphone peer-to-peer setting requires distributed algorithms that can provide global reliability and efficiency guarantees on top of an unpredictable collection of local links. As detailed below, the models that describe this emerging setting are sufficiently different from existing models that new algorithms and analysis techniques are required. This paper addresses this need by describing and analyzing new gossip algorithms for this important setting.

The Mobile Telephone Model. The mobile telephone model studied in this paper was introduced in recent work [10, 23]. It is a variant of the classical telephone peer-to-peer model (e.g., [3, 6–8, 11–15]) modified to better describe the capabilities and constraints of existing smartphone peer-to-peer services. The details of the mobile

telephone model are inspired, in particular, by the current specifications of Apple's Multipeer Connectivity framework [20]: a peer-to-peer service available in every iOS version since iOS 7 that allows nodes to advertise services, discover nearby advertisers, and attempt to connect to nearby advertisers, using only local radio broadcast. (The definition of the classical telephone model, and differences between the classical telephone and mobile telephone model, are detailed and discussed below in the related work section.)

In more detail, the mobile telephone model abstracts the basic *scan-and-connect* dynamics of the Multipeer framework as follows. Time proceeds in synchronous rounds. In each round, a connected graph describes the underlying network topology for that round. At the beginning of each round, each device (also called a *node* in the following) learns its neighbors in the topology graph (e.g., as the result of a scan). Each device can then attempt to initiate a connection with a neighbor. Each node can support at most one connection—so if multiple nodes attempt to connect with the same target, only one connection will succeed. If two nodes connect, they can perform a bounded amount of reliable communication before the round ends.

We parameterize this model with a *tag length* $b \geq 0$. At the beginning of each round, each node can choose a *tag* consisting of b bits to advertise. When performing a scan, each node learns both the ids and chosen tags of its neighbors (where $b = 0$ means there are no tags). These tags can change from round to round. In our previous study of rumor spreading with parameter $b = 1$ [10], for example, at the beginning of a given round, each node that already knows the rumor advertises a 1 with its tag, while other nodes advertise a 0. This simplified the rumor spreading task by enabling nodes that know the rumor to only attempt to connect to nodes that do not. This capability of nodes to use tags to deliver limited information to their neighbors is motivated by the ability of devices to choose and change their service advertisements in the Multipeer framework.

We also parameterize the model with a *stability factor* $\tau \geq 1$. The underlying network topology must stay stable for at least τ rounds between changes. For $\tau = 1$, for example, the network topology can change completely in every round, while for $\tau = \infty$, the topology never changes. There exist finer-grained approaches for capturing intermediate levels of stability (e.g., T-interval connectivity [19]), but in this paper we study only the two extreme cases of fully dynamic and fully stable topologies, so our simpler stability factor definition is sufficient. The need to model topology changes is motivated by the inherently mobile nature of the smartphone setting.

Results. In this paper, we describe and analyze new algorithms for the *gossip* problem in the mobile telephone model with respect to different model parameter and algorithm assumptions. This problem assumes a subset of nodes start with messages (also called *tokens*). The goal is to spread these messages to the entire network. Gossip is fundamental in distributed computing and is considered particularly important for ad hoc networks such as the smartphone meshes studied in this paper (c.f., the introductory discussion in [24]).

Below (and in Figure 1) we state and discuss our main results. In the following, let $n > 1$ be the network size and $k, 1 \leq k \leq n$, be the number of tokens in the system. For a given topology graph, we use α to describe its vertex expansion (see the model discussion below) and Δ to describe its maximum degree.[1] We assume the topologies are connected. All round complexity results hold with high probability in n (i.e., probability at least $1 - 1/n$).

Assumptions	Algorithm	Gossip Round Complexity
Standard Gossip		
$b = 0, \tau \geq 1$	BlindMatch	$O((1/\alpha)k\Delta^2 \log^2 n)$
$b = 1, \tau \geq 1$	SharedBit*	$O(kn)$
$b = 1, \tau \geq 1$	SimSharedBit**	$O(kn + (1/\alpha)\Delta^{1/\tau} \log^6 n)$
$b = 1, \tau = \infty$	CrowdedBin	$O((k/\alpha) \log^6 n)$
ϵ-Gossip ($0 < \epsilon < 1$)		
$b = 1, \tau \geq 1$	SharedBit*	$O\left(\frac{n\sqrt{\Delta \log \Delta}}{(1-\epsilon)\alpha}\right)$

Figure 1: A summary of gossip and ϵ-gossip round complexity bounds proved in this paper. (In the ϵ-gossip problem, it is assumed that every node starts with a message, but each node need only learn an ϵ-fraction of the n total messages.) In the following: n is the network size, k is the number of gossip messages, α and Δ are the vertex expansion and maximum degree, respectively, of the network topology graph, b is the tag length, and τ is the stability factor. All results hold with high probability in n (i.e., at least $1 - 1/n$). Notice, the result for $b = 0$ and $\tau \geq 1$ is the best known result even for the easier case of $b = 0$ and $\tau = \infty$. (*) The SharedBit algorithm (alone among all algorithms studied) requires shared randomness. (**) The SimSharedBit algorithm is existential in the sense that it depends on a pseudorandomness generator that we prove exists in Section 5.

We start by considering the difficult setting where $b = 0$ and $\tau = 1$; i.e., nodes cannot use tags and the network topology graph can change completely in each round. In Section 4, we describe and analyze a natural strategy for this setting called BlindMatch, which has nodes select neighbors with uniform randomness to send connection attempts.[2] We prove that BlindMatch solves gossip in $O((1/\alpha)k\Delta^2 \log^2 n)$ rounds. This bound might seem pessimistic at first glance, but it is known that disseminating even a single message in the mobile telephone model with this strategy can take $\Omega(\Delta^2/\sqrt{\alpha})$ rounds in some networks [23]. Indeed, this lower bound holds even for the easier assumption that $\tau = \infty$. Accordingly, we do not consider $b = 0$ and $\tau = \infty$ as a distinct case in this paper. (To provide intuition for why $\Omega(\Delta^2)$ rounds are sometimes necessary, consider two stars centered on u and v, respectively, where each star has around Δ points and u and v are connected by an edge. Assume u starts with a gossip message. For v to receive this message two events must happen: (1) u selects v for a connection; and (2) v accepts u's connection from all incoming connections in that round. The first event occurs with probability $\approx 1/\Delta$, and because v can expect a constant fraction of its neighbors to send it connection

[1] If the topology is dynamic, then α is defined as the minimum expansion over all rounds, and Δ is defined as the largest maximum degree over all rounds.

[2] This is essentially the well-known PUSH-PULL strategy from the classical telephone model with the key exception that in our model if a node receives multiple connection attempts, only one succeeds. As discussed in the related work and Section 4, this well-motivated model change requires new analysis techniques to understand information propagation.

attempts in any given round, the second event also occurs with probability $\approx 1/\Delta$.) Our BlindMatch result provides the benchmark against which we attempt to improve with the algorithms that follow.

In Section 5, we consider the case where $b = 1$ and $\tau \geq 1$; i.e., the network can still change completely in each round, but now nodes can advertise a single bit to their neighbors. We begin by describing and analyzing an algorithm called SharedBit. This algorithm assumes a shared randomness source which is used to implement (essentially) a random hash function that allows nodes to hash their current set of known messages to a single bit to be used as their one-bit advertising tag. The key guarantee of this function is that nodes with the same sets advertise the same bit, and nodes with different sets have a constant probability of advertising different bits. This helps nodes seek out productive connections with neighbors (e.g., connections in which at least one node learns something new). We prove that SharedBit solves gossip in $O(kn)$ rounds.

We next seek to eliminate the shared randomness assumption. To do so, we describe SimSharedBit which solves gossip in $O(kn + (1/\alpha)\Delta^{1/\tau} \log^6 n)$ rounds, without assuming a shared randomness source. Notice, because $\alpha \geq 2/n$ and $\Delta \leq n$, this solution is always within log factors of the SharedBit for large k, and for small k it is still comparable for many values of α, Δ, and/or τ.

The SimSharedBit algorithm depends on a novel generalization of *Newman's Theorem* [21]—a well-known result on public randomness simulation from the study of two-party communication complexity. We prove that there exists an appropriate pseudorandom number generator that can provide sufficient randomness for the SharedBit strategy. We then elect a leader in $O((1/\alpha)\Delta^{1/\tau} \log^6 n)$ rounds using an algorithm from [23], and use this leader to disseminate a small generator seed. We note that our generalization of Newman's Theorem is potentially of standalone interest as the techniques we introduced can be used to study pseudorandomness in many different graph algorithm settings.

In Section 6, we consider the impact of topology changes on gossip time. In particular, we consider the case where $b = 1$ and $\tau = \infty$; i.e., the network topology is stable. We describe and analyze CrowdedBin, an algorithm that solves gossip in $O((1/\alpha)k \log^6 n)$ rounds. This algorithm matches or outperforms the $O(kn)$ round complexity of SharedBit for all α values (ignoring log factors). For well-connected networks (e.g., constant α), it performs almost a factor of n faster. These results hint that large increases to stability are more valuable to gossip algorithms than large increases to tag length (for most of our solutions, increasing b beyond 1 only improves performance by at most logarithmic factors).

The benefit of stable network topologies is that nodes can transmit larger amounts of information about their current state to their neighbors by using their single bit advertisement tag over multiple rounds. CrowdedBin leverages this capability to help nodes efficiently converge on an accurate estimate of k—which is not known in advance. This process depends on nodes testing guesses by throwing their tokens into a number of bins corresponding to the current guess, and then seeking/spreading evidence of crowding (as established by a new balls-in-bins algorithm described in Section 6). Once all nodes learn an appropriate guess of k, CrowdedBin deploys an efficient parallel rumor spreading strategy to efficiently disseminate the k tokens.

Finally, we consider the *ϵ-gossip* problem, which is parameterized with a fraction $\epsilon, 0 < \epsilon < 1$, assumes that $k = n$, and relaxes the gossip problem to require only that every node receives at least $n\epsilon$ of the n total tokens. This variation is useful for settings where it is sufficient for nodes to learn *enough* rumors to complete the task at hand; e.g., when an algorithm requires responses from only a majority quorum of nodes.

In Section 7, we re-analyze the SharedBit gossip algorithm from Section 5. Deploying a novel argument based on finding productive "coalitions" of nodes, we show that SharedBit solves ϵ-gossip in $O\left(\frac{n\sqrt{\Delta \log \Delta}}{(1-\epsilon)\alpha}\right)$ rounds. Recall that SharedBit solves regular gossip in $O(n^2)$ rounds under the $k = n$ assumption. Therefore, when ϵ is a constant fraction and the network is well-connected (α is large), SharedBit solves ϵ-gossip up to a (sub-linear) polynomial factor faster than the standard gossip problem.

Related Work. The mobile telephone model used in this paper was first introduced in a study of rumor spreading by Ghaffari and Newport [10]. We also recently studied leader election in this same model [23]. As noted, the mobile telephone model is a variation of the classical telephone model (first introduced by Frieze and Grimmett [7]) adapted to better describe smartphone peer-to-peer networks. The mobile model differs from the classical model in two ways: (1) the classical model implicitly fixes $b = 0$ and (typically) $\tau = \infty$; and (2) the classical model allows nodes to accept an unbounded number of incoming connections.

It is important to emphasize that most of the well-known bounds in the classical model depend on this assumption of unbounded connections, and removing this assumption requires new analysis techniques; c.f., the discussion in [10]. We note that work by Daum et al. [4] (which preceded [10, 23]) also pointed out the dependence of existing telephone model bounds on unbounded concurrent connections.

A fundamental problem in peer-to-peer networks is *rumor spreading*, in which a single message must be disseminated from a designated source to all nodes (this is equivalent to gossip with $k = 1$). This problem is well-understood in the classical telephone model, where spreading times are often expressed with respect to spectral properties of the network topology graph such as graph conductance (e.g., [12]) and vertex expansion (e.g., [3, 6, 13, 15]). This existing work established that efficient rumor spreading is possible with respect to both graph properties in the classical model. In [10], we studied this problem in the mobile telephone model. We proved that efficient rumor spreading with respect to conductance *is not* possible in the mobile telephone model, but efficient spreading with respect to vertex expansion *is* possible. We then proved that for $b = 1$ and $\tau \geq 1$, a simple random spreading strategy solves the problem in $O((1/\alpha)\Delta^{1/\tau}\text{polylog}(n))$ rounds—matching the tight $\Theta((1/\alpha)\log^2 n)$ result from the classical telephone model within log factors for $\tau \geq \log \Delta$. In [23], we built on these results to solve leader election in similar asymptotic time.

Though gossip is well-studied in peer-to-peer models (see [24] for a good overview), little is known about how to tackle the problem in the mobile telephone model, where concurrent connections

are now bounded but nodes can leverage advertising tags.[3] Finally, we note that there are application similarities between gossip in the mobile telephone model and existing reliable multicast solutions for mobile ad hoc (e.g., [17]) and delay-tolerant (e.g., [1]) networks. These existing solutions, however, tend to be empirically evaluated and depend on the ability to predict information about link behavior (e.g., estimated link duration or an advance schedule of when given links will be present).

2 MODEL AND PROBLEM

We describe a smartphone peer-to-peer network using the *mobile telephone model*. As elaborated in the introduction, the basic properties of this model—including its scan-and-connect behavior, dynamic topologies, and the ability to advertise a bounded tag—are inspired in particular by the behavior of the Apple Multipeer Connectivity framework for smartphone peer-to-peer networking.

In more detail, we assume executions proceed in synchronous rounds labeled 1, 2, ... We describe a peer-to-peer network topology in each round r as an undirected connected graph $G_r = (V, E_r)$ that can change from round to round, constrained by the stability factor (see below). We call the sequence of graphs $G_1, G_2, \ldots$ that describe the evolving topology a dynamic graph. We assume the definition of the dynamic graph is fixed at the beginning of the execution.

We assume a computational process (also called a *node* in the following) is assigned to each vertex in V, and use $n = |V|$ to indicate the network size. We assume all nodes start during round 1. At the beginning of each round r, we assume each node u learns its neighbor set $N(u)$ in G_r. Node u can then select at most one node from $N(u)$ and send a connection proposal. A node that sends a proposal cannot also receive a proposal. If a node v does not send a proposal, and at least one neighbor sends a proposal to v, then v can *accept* an incoming proposal. There are different ways to model how v selects a proposal to accept. In this paper, for simplicity, we assume v accepts an incoming proposal selected with uniform randomness from the incoming proposals. If node v accepts a proposal from node u, the two nodes are *connected* and can perform a bounded amount of interactive communication to conclude the round. We leave the specific bound on communication per connection as a problem parameter.

Model Parameters. We parameterize the mobile telephone model with two integers, a *tag length* $b \geq 0$ and a *stability factor* $\tau \geq 1$. We allow each node to select a *tag* containing b bits to advertise at the beginning of each round. That is, if node u chooses tag b_u at the beginning of a round, all neighbors of u learn b_u before making their connection decisions in this round. A node can change its tag from round to round.

We also allow for the possibility of the network topology changing between rounds. We bound the allowable changes with a stability factor $\tau \geq 1$. For a given τ, the dynamic graph describing the changing topology must satisfy the property that at least τ rounds must pass between any changes to the topology. For $\tau = 1$, the graph can change arbitrarily in every round. We use the convention of stating $\tau = \infty$ to indicate the graph never changes.

Vertex Expansion and Maximum Degree. Several of our results express time complexity bounds with respect to the *vertex expansion* α of the dynamic graph describing the network topology. To define α, we first review a standard definition of vertex expansion for a fixed static unconnected graph $G = (V, E)$.

For a given $S \subseteq V$, define the *boundary* of S, indicated ∂S, as follows: $\partial S = \{v \in V \setminus S : N(v) \cap S \neq \emptyset\}$: that is, ∂S is the set of nodes not in S that are directly connected to S by an edge in E. Next define $\alpha(S) = |\partial S|/|S|$. As in [10, 13], we define the *vertex expansion* $\alpha(G)$ of our static graph $G = (V, E)$ as follows:

$$\alpha(G) = \min_{S \subset V, 0 < |S| \leq n/2} \alpha(S).$$

Notice that despite the possibility of $\alpha(S) > 1$ for some S, we always have $\alpha(G) \leq 1$. We define the vertex expansion α of a *dynamic* graph $G_1, G_2 \ldots$, to be the minimum vertex expansion over all of the dynamic graph's constituent static graphs (i.e., $\alpha = \min\{\alpha(G_i) : i \geq 1\}$).

Similarly, we define the maximum degree Δ of a dynamic graph to be the maximum degree over all of the dynamic graph's constituent static graphs.

The Gossip Problem. The gossip problem assumes each node is provided an upper bound[4] $N \geq n$ on the network size and a unique ID (UID) from $[N]$. The problem assumes some subset of nodes begins with a gossip message to spread (which we also call a *token*). We use k to describe the size of this subset and assume that k is not known to the nodes in advance. A given node can start the execution with multiple tokens, but no token starts at more than one node. We treat gossip tokens as comparable black boxes that can only be communicated between nodes through connections (e.g., a node cannot transmit a gossip token to a neighbor by spelling it out bit by bit using its advertising tags). If a node begins an execution with a token or has received the token through a connection, we say that the node *owns*, *knows* or has *learned* that token. We assume that a pair of connected nodes can exchange at most $O(1)$ tokens and $O(\text{polylog}(N))$ additional bits during a one round connection.

Solving the Gossip Problem. The gossip problem requires all nodes to learn all k tokens, Formally, we say a distributed algorithm *solves the gossip problem in* $f(n, k, \alpha, b, \tau)$ *rounds*, if with probability at least $1 - 1/n$, all nodes know all k tokens by round $f(n, k, \alpha, b, \tau)$ when executed in a network of size n, with k tokens, vertex expansion α, tag length b, and stability factor τ. We omit parameters when not relevant to the bound.

Probability Preliminaries. The analyses that follow leverage the following well-known probability results:

Theorem 2.1. *For* $p \in [0, 1]$: $(1 - p) \leq e^{-p}$ *and* $(1 + p) \geq 2^p$.

[3] It might be tempting to simply run k parallel instances of the rumor spreading strategy from [10] to gossip k messages, but this approach fails for three reasons: (1) our model allows only $O(1)$ tokens to be sent per connection per round; (2) each of the k instances requires its own advertising tag bit, whereas all of our new gossip results focus on the case where $b \leq 1$; and (3) nodes do not know k in advance. Accordingly, most results presented in this paper require substantial technical novelty.

[4] For the sake of concision, the results described in the introduction and Figure 1 make the standard assumption that N is a polynomial upper bound on n, allowing us to replace N with n within logarithmic factors inside asymptotic notation. In the formal theorem statements for these results, however, we avoid this simplification and leave N in place where used—enabling a slightly finer-grained understanding of the impact of the looseness of network size estimation on our complexity guarantees.

Theorem 2.2 (Chernoff Bound: Lower Bound Form). *Let $Y = \sum_{i=1}^{t} X_i$ be the sum of $t > 0$ i.i.d. random indicator variables X_1, X_2,..., X_t, and let $\mu = E(Y)$. Fix some fraction δ, $0 < \delta < 1$. It follows:*

$$\Pr(X \leq (1-\delta)\mu) \leq e^{-\frac{\delta^2\mu}{2}}.$$

Theorem 2.3 (Chernoff Bound: Upper Bound Form). *Let $Y = \sum_{i=1}^{t} X_i$ be the sum of $t > 0$ i.i.d. random indicator variables X_1, X_2,..., X_t, and let $\mu = E(Y)$. Fix some value $\delta > 1$. It follows:*

$$\Pr(X \geq (1+\delta)\mu) \leq e^{-\frac{\delta\mu}{3}}.$$

Theorem 2.4 (Chernoff-Hoeffding Bound). *Let $X_1, X_2, ..., X_t$, be $t \geq 1$ i.i.d. random indicator variables. Let $\mu = E(X_i)$ and fix some $\delta > 0$. It follows:*

$$\Pr\left(\frac{1}{t}\sum_{i=1}^{t} X_i \geq \mu + \delta\right) \leq e^{-2\delta^2 t}.$$

Theorem 2.5 (Markov's Inequailty). *Let X be a nonnegative random variable and $a > 0$ be a real number. It follows:*

$$\Pr(X \geq a) \leq \frac{E(X)}{a}.$$

3 TOKEN TRANSFER SUBROUTINE

An obstacle to solving gossip in the mobile telephone model is deciding which tokens to exchange between two connected nodes. In more detail, once two nodes u and v with respective token sets T_u and T_v connect, even if they *know* $T_u \neq T_v$, they must still identify at least one token $t \notin T_u \cap T_v$ to transfer for this round of gossip to be useful. Complicating this task is the model restriction that u and v can only exchange $O(\text{polylog}(N))$ bits before deciding which tokens (if any) to transfer. This is not (nearly) enough bits to encode a full token set (a simple counting argument establishes that every coding scheme will require $\Omega(N)$ bits for some sets). Therefore, a more efficient routine is needed to implement this useful token transfer.

Here we describe a *transfer subroutine* that solves this problem and is used by multiple gossip algorithms described in this paper. This routine, which we call $Transfer(\epsilon)$, for an error bound ϵ, $0 < \epsilon < 1$, is a straightforward application of an existing algorithmic tool from the literature on two-party communication complexity. It guarantees the following: if $Transfer(\epsilon)$ is called by two connected nodes u and v, with respective token sets T_u and T_v, and $T_u \neq T_v$, then with probability at least $1-\epsilon$ the smallest token t (by a predetermined token ordering) that is *not* in $T_u \cap T_v$, will be transferred by the node that knows t to the node that does not. This routine requires u and v to exchange only $O(\log^2 N \cdot \log(\frac{\log N}{\epsilon}))$ controls bits in addition to token t. It also assumes some fixed ordering on tokens.

Equality Testing. We use one of the many known existing solutions to the *set equality* (EQ) problem from the study of two-party communication complexity. In our setting with u and v (described) above, these existing solutions provide u and v a way to test the equality of T_u and T_v, and they offer the following guarantee: if $T_u = T_v$, then u and v will correctly determine their sets are equal with probability 1, else if $T_u \neq T_v$ then u and v will *erroneously* determine their sets are equal with probability no more than 1/2. These existing solutions assume only private randomness and require u and v to exchange no more than $O(\log N)$ bits. A nice property of most such solutions is that each trial is independent. Therefore, if u and v repeat this test c times, for some integer $c \geq 1$, then the error probability drops exponentially fast with c to 2^{-c}. Let us fix one such equality testing routine and call it $EQTest(c)$, where parameter $c \geq 1$ determines how many trials to execute in testing the equality.

The Transfer Subroutine. We now deploy $EQTest(\epsilon')$, for $\epsilon' = \lceil \log(\frac{\log N}{\epsilon}) \rceil$, as a subroutine to implement the $Transfer(\epsilon)$ routine. In particular, recall that for a given u and v, we can understand T_u and T_v to both be subsets of the values in $[N]$ (as each node in the network can label each token with its UID from $[N]$ at the beginning of the execution). Our goal is to identify the smallest location value in $[N]$ that is in $T_u \cup T_v$ but not in $T_u \cap T_v$. To do so, we can implement a binary search over the interval $[N]$, using $EQTest(\epsilon')$ to test the equality of the interval in question between u and v. In more detail:

Transfer(ϵ):
  $a \leftarrow 1$; $b \leftarrow N$
  while $a \neq b$
    $result \leftarrow$ **EQTest**(ϵ') executed on $T_u \cap [a, \lfloor b/2 \rfloor]$ and $T_v \cap [a, \lfloor b/2 \rfloor]$
    if $result = notequal$ **then** $b \leftarrow \lfloor b/2 \rfloor$ **else** $a \leftarrow \lfloor b/2 \rfloor + 1$
  transfer token a to the other node if you know token a

The above logic implements a basic binary search over the interval $[N]$ to identify the smallest value in this interval that is in exactly one of the two sets T_u and T_v. If every call to $EQTest$ succeeds then the search succeeds and $Transfer$ behaves correctly. There are at most $\log N$ calls to $EQTest$, each of which fails with probability $2^{-\epsilon'} \leq \epsilon / \log N$. Therefore, by a union bound, the probability that at least one of the $\log N$ calls to $EQTest$ fails is less than ϵ, as claimed. From a communication complexity perspective, each call to $EQTest(\epsilon')$ requires $O(\log N \cdot \epsilon') = O(\log N \cdot \log(\log N/\epsilon))$ bits, and we make $\log N$ such calls. Therefore, the total communication complexity is in $O(\log^2 N \cdot \log(\frac{\log N}{\epsilon}))$, as claimed.

4 GOSSIP WITH 0-BIT TAGS AND DYNAMIC TOPOLOGY

Here we consider the most difficult case for gossip in our model: nodes cannot advertise any information to their neighbors ($b = 0$), and the network topology graph can change arbitrarily in every round ($\tau = 1$). We study the performance of a straightforward random token propagation strategy.

A detailed treatment of this algorithm, including the full proof details, are deferred to the full version of this paper [22].

The BlindMatch Gossip Algorithm. At the beginning of each round $r \geq 1$, each node $u \in V$ flips a fair coin to decide whether to be a *sender* or a *receiver* in r. If u decides to be a sender, it selects a neighbor uniformly from among its neighbors in this round and sends it a connection proposal. If u decides to be a receiver it waits

to receive proposals. If two nodes u and v connect, they execute the token transfer subroutine.

Analysis. Our goal is to prove the following theorem regarding the performance of BlindMatch:

Theorem 4.1. *The BlindMatch gossip algorithm solves the gossip problem in $O((1/\alpha)k\Delta^2 \log^2 N)$ rounds when executed with tag length $b = 0$ in a network with stability $\tau \geq 1$.*

This time bound might seem pessimistically large at first glance, but as shown in [23], there are networks in which simple random connection strategies require $\Omega(\Delta^2/\sqrt{\alpha})$ rounds to spread even a single message. The proof details adapt our recent analysis of leader election strategies in the mobile telephone model under the assumption that $b = 0$ [23] to account for multiple gossip messages.

5 GOSSIP WITH 1-BIT TAGS AND DYNAMIC TOPOLOGY

Here we describe and analyze two gossip algorithm that now assume $b = 1$ (i.e., nodes can advertise a single bit to their neighbors in each round). The graph, however, can still change arbitrarily in every round (i.e., the algorithms must work for any $\tau \geq 1$). Our goal is exploit the shift from $b = 0$ to $b = 1$ to solve gossip more efficiently in most cases. Our first algorithm, called SharedBit, assumes shared randomness. Our second algorithm, called SimSharedBit, does not, but it relies on an existential pseudorandomness generator.

5.1 Shared Randomness

Below we provide a condensed description of the SharedBit gossip algorithm and discuss its performance. A detailed treatment of this algorithm, including the full proof details, are deferred to the full version of this paper [22].

The SharedBit Gossip Algorithm (Overview). The algorithm assumes each node has access to a shared random bit string $\hat{r}$ of length $O(N^3 \log N)$, where each bit in $\hat{r}$ is generated with independent and uniform randomness. At the beginning of each round r, each node u uses fresh bits from $\hat{r}$ to hash the set of tokens it knows at the beginning of this round to a single bit $b_u(r)$. This bit $b_u(r)$ is what u advertises to its neighbors in round r. If $b_u(r) = 0$, then u will receive connection proposals in this round. If $b_u(r) = 1$ and u has at least one neighbor advertising 0, then u will choose one these neighbors with uniform randomness and send it a connection proposal. If two nodes u and v connect in round r, they use the token transfer subroutine to attempt to exchange tokens. If a node runs out of fresh bits $\hat{r}$ to use it can cycle back to the beginning of this shared string.

Analysis. Our goal is to prove the following theorem regarding the SharedBit gossip algorithm:

Theorem 5.1. *The SharedBit gossip algorithm solves the gossip problem in $O(kn)$ rounds when executed with shared randomness and tag length $b = 1$ in a network with stability $\tau \geq 1$.*

The proof of this theorem begins by establishing the following key property of the token hash strategy used by SharedBit (where $T_u(r)$ is set of tokens known by node u at the beginning of round r):

Lemma 5.2. *Fix two nodes $u, v \in V$, $u \neq v$, and a round r, $1 \leq r \leq cN^2$. Fix a $r-1$ round execution of SharedBit, and let $p = \Pr(b_u(r) \neq b_v(r))$ be the probability (defined over the random selection of the relevant bits in $\hat{r}$) that u and v generate different advertising bits in round r. If $T_u(r) = T_v(r)$ then $p = 0$, else if $T_u(r) \neq T_v(r)$, then $p = 1/2$.*

This property ensures that any successful connection created by SharedBit is *productive* in the sense that at least one endpoint will learn a new token. The remainder of the proof first notes that with constant probability there will be *some* productive connections in any given round. It then leverages a stochastic dominance argument (to sidestep inter-round dependencies) to show that $\Theta(kn)$ rounds are sufficient to guarantee enough productive rounds to solve the gossip problem for k tokens.

5.2 Eliminating the Shared Randomness Assumption

Shared randomness is not always a reasonable assumption. With this in mind, we describe and analyze SimSharedBit, a variation of SharedBit that does not require shared randomness. A detailed description and analysis of SimSharedBit is deferred to the full version of this paper [22]. Here we summarize the algorithm's strategy and state its main performance guarantee.

SimSharedBit Strategy. The high-level strategy for SimSharedBit is to first elect a leader that disseminates a *seed* string that can be used to generate sufficient randomness to run SharedBit. This leader election uses a strategy from [23], and requires an additional $O((1/\alpha)\Delta^{1/\tau}\text{polylog}(N))$ rounds. Notice, however, this time is asymptotically dominated by kn (ignoring log factors) for many parameter values.

We emphasize that the number of shared bits required by SharedBit is much too large to be efficiently disseminated directly: our model restricts connections to deliver polylog(N) bits per round, while SharedBit requires $\Omega(N^3)$ shared bits. The seed selected and disseminated by the leader, by contrast, will be small enough to be fully transmitted over a single round connection.

To prove that there exists a randomness generator that can extract sufficient randomness for our purpose from seeds of this small size, we adapt the technical details of Newman's Theorem [21] from the simpler world of two-party communication to the more complicated world of n parties on a distributed and changing network topology. In more detail, we prove the existence of a multiset $\mathcal{R}'$, containing only poly(N) bit strings of the length required for SharedBit, that is *sufficiently* random to guarantee that if a leader chooses $\hat{r}$ uniformly from $\mathcal{R}'$, the SharedBit algorithm using shared randomness $\hat{r}$ is still likely to solve gossip efficiently. Because $\mathcal{R}'$ contains only poly(N) strings, the leader can identify the string it selected using only polylog(N) bits (this selection is the *seed* it disseminates)—enabling efficient propagation of this information. The existential nature of SimSharedBit is entirely encapsulated in the existence of this set $\mathcal{R}'$.

Analysis. Our goal is to prove the following theorem regarding SimSharedBit:

Theorem 5.3. *There exists a bit string multiset $\mathcal{R}'$ of size $N^{\Theta(1)}$, such that the SimSharedBit gossip algorithm using this $\mathcal{R}'$ as its*

source of simulated shared bit strings solves the gossip problem in $O(kn + (1/\alpha)\Delta^{1/\tau} \log^6 N)$ *rounds when executed with tag length* $b = 1$ *in a network with stability* $\tau \geq 1$.

The key technical novelty in proving this theorem is the proof of the existence of a sufficiently random $\mathcal{R}'$. Our argument generalizes the strategy deployed to prove Newman's Theorem [21]. Whereas the classical Newman's Theorem result applies the probabilistic method to prove the existence of a sufficiently random bit string multiset for all possible inputs for two players, we have to account for $n \geq 2$ players, all possible different dynamic graphs, assignments of tokens, and rounds required for the leader election to complete. These counts, however, are of sizes no more than exponential in N, allowing the probabilistic method argument to proceed with proper adjustments to constants. Our generalization is potentially of standalone interest as it can be used to provide similar public randomness simulation results for other dynamic network algorithms.

6 GOSSIP WITH 1-BIT TAGS AND STABLE TOPOLOGY

The preceding gossip algorithms make no assumptions about the stability of the underlying network topology. Their analysis holds for every $\tau \geq 1$. In some settings, this assumption might be pessimistic. Here we seek better performance when the network topology graph does not change (i.e., $\tau = \infty$).

In more detail, we describe and analyze the CrowdedBin algorithm, which solves gossip in $\tilde{O}(k/\alpha)$ rounds with $b = 1$ and $\tau = \infty$ (where $\tilde{O}$ hides polylog(N) factors). This algorithm is comparable to the trivial $\Omega(k)$ lower bound for gossiping k tokens in a model where a node can only send one token per round. It also outperforms our best result for $\tau = 1$—the $O(kn)$ round complexity of SharedBit—for every $\alpha \in \omega(1/n)$ (ignoring log factors).

Discussion: Crowded Bins. The name CrowdedBin comes from a core behavior in the algorithm in which nodes toss their tokens into a fixed number of bins corresponding to their current estimate $\hat{k}$ of k (the number of tokens in the network). Nodes do not know k in advance. Determining this value is crucial to enabling efficient parallel dissemination of their tokens. Leveraging a new balls-in-bins analysis, we upper bound the number of tokens in any given bin *if* the estimate $\hat{k}$ is sufficiently large. The nodes therefore search for crowded bins as evidence that they need a larger estimate of k. This mechanism provides a way to check that a current guess $\hat{k}$ is too small while only paying a time complexity price relative to $\hat{k}$ (as there are only $\hat{k}$ bins required to check for crowding). Because the sequence of guesses we try are geometrically increasing, the cost of checking estimates smaller than k will sum up to $\tilde{O}(k)$.

Discussion: Spreading Bits versus Spreading Tokens. We also emphasize that the CrowdedBin algorithm makes a clear distinction between propagating information using the advertising bits and propagating the tokens themselves (which are treated as black boxes, potentially large in size, that require a pairwise connection for transfer). Combining the stability of the network with each node's ability to advertise a bit to *all* its neighbors in each round, nodes first attempt to stabilize to a consistent and accurate estimate of k, and a consistent set of *tags* describing the network's tokens. Once stabilized, this information can then support the efficient spreading of the tokens, link by link, to the whole network. Accordingly, CrowdedBin can be understood to occur in two phases (which, in practice, might substantially overlap). During the first phase, nodes use their advertising bits to efficiently learn about the network. During the second phase, nodes use this knowledge to efficiently spread gossip tokens. The first phase depends on network stability as this is what allows a node to communicate complicated information to its neighborhood using its small advertising tag over many rounds.

The PPUSH Rumor Spreading Strategy. The CrowdedBin algorithm uses a simple rumor spreading strategy called PPUSH as a subroutine to help spread tokens once the network has stabilized. This algorithm was introduced in our earlier study of rumor spreading in the mobile telephone model [10]. PPUSH assumes a subset of nodes start with a common rumor m, and the goal is to spread m to all nodes. It requires $b \geq 1$.

In more detail, the strategy PPUSH works as follows: (1) at the beginning of each round, if a nodes knows m (i.e., it is *informed*), it advertises bit 1, otherwise if it does not know m (i.e., it is *uninformed*), it advertises bit 0; (2) each informed node that has at least one uninformed neighbor in this round, chooses an uninformed neighbor with uniform randomness and attempts to form a connection to spread the rumor. In [10], we proved the following key result about the performance of PPUSH:

Theorem 6.1 (Adapted from [10]). *With high probability in* N*: PPUSH succeeds in spreading the rumor to all nodes in* $O(\log^4 N/\alpha)$ *rounds when executed in the mobile telephone model with* $b \geq 1$, $\tau = \infty$*, and a topology graph with expansion* α.

We will leverage this theorem in our analysis of our gossip algorithm. We also use the following useful property proved in [10] which relates network diameter to expansion:[5]

Theorem 6.2 (Adapted from [10]). *Fix a connected graph with* n *nodes, expansion* α*, and diameter* D*. It follows that* $D = O(\log n/\alpha)$.

6.1 The CrowdedBin Gossip Algorithm

We are now ready to describe our CrowdedBin algorithm. We present it here in its own section with its description divided into parts to clarify its presentation.

In the following, we assume each node $u \in V$ identifies itself with a *tag* t_u chosen uniformly from the space $\{1, 2, ..., N^\beta\}$, where $\beta \geq 2$ is constant we fix in our analysis. Let $\ell = \beta \log N$ be the number of bits needed to describe a tag. To simplify notation, we assume in the following that N is a power of 2.

Parallelizing Instances. Nodes do not know in advance the value of k (the number of tokens in the system). They consider $\log N$ *estimates* of k: $k_1, k_2, ..., k_{\log N}$, where each $k_i = 2^i$. The nodes run in parallel a separate gossip *instance* for each estimate. We use the notation *instance* i to refer to the instance corresponding to estimate

[5] The actual result we proved in [10] is that it is always possible to spread a rumor in $O(\log n/\alpha)$ rounds in the mobile telephone model in a graph with expansion α. The rumor spreading time in a given network can never be smaller than the network diameter, which provides a trivial lower bound on the problem.

k_i. In order to run $\log N$ instances in parallel, each node uses $\log N$ rounds to simulate one round each of the $\log N$ instances. That is, nodes divide rounds into *simulation groups* consisting of $\log N$ rounds. Round j of simulation group i is used to simulate round i of instance j.

Instance Schedules. Each instance i groups its rounds into *blocks* containing $\ell + \log N$ rounds each. It then groups these blocks into *bins* containing $\gamma \log N$ blocks each, where $\gamma > 1$ is a constant we fix in our analysis below. Finally, it groups the bins into *phases* consisting of k_i bins each. In other words, the schedule for instance i is made up of phases, where each phase has k_i bins, which are each made up of $\gamma \log N$ blocks, which each contain $\ell + \log N$ rounds: adding to a total of $\gamma(\beta + 1)k_i \log^2 N$ total rounds per phase.

Initialization. Each node $u \in V$ that begins an execution of the CrowdedBin algorithm with a gossip token, independently selects a bin for its token for each of the $\log N$ instances. That is, for each instance i, u selects a bin $b_u(i)$ with uniform independent randomness from $\{1, 2, ..., k_i\}$. Each node u also maintains, for each instance i, and each bin j for this instance, a set $T_u(i, j)$ containing the tags it has seen so far for tokens in bin j in instance i. For each instance i, if node u has a token it initializes $T_u(i, b_u(i)) = \{t_u\}$ (i.e., it places its own tag in the bin it selected for that instance). Node u also maintains a set Q_u containing the tokens it has received so far, where each token in Q_u is also labeled with its tag. Finally, each node u maintains a variable est_u, initialized to 1, which describes the current instance node u is participating in.

Participation. Each node will only participate in a single instance at a time, and it will only participate in complete phases of an instance. In more detail, if some instance i starts a new phase in round r, and some node u has $est_u = i$ at the start of round r, node u is now committed to participate in this full phase of instance i. As we will detail, its estimate cannot change again until this phase completes.

To participate in a phase of instance i, node u does the following. First, for each bin j, $1 \leq j \leq k_i$, u orders the tags in $T_u(i, j)$ (if any) in increasing order. It will use the first ℓ rounds of the first block to spell out the smallest such tag, bit by bit, using its advertising bits (here the assumption that $b \geq 1$ is needed). It will then use the first ℓ rounds of the second block to spell out the second smallest tag, and so on. There are $\gamma \log N$ total blocks in this bin. If u knows more than this many tags for this bin, it transmits only the first $\gamma \log N$. Node u transmits all 0's during the blocks in this bin for which it has no tags to advertise (here is where we use the assumption that the smallest possible tag is 1—preventing a block of all 0's from being mistaken for a tag.)

During the rounds dedicated to bin j, node u also collects the bits advertised by its neighbors in each block. If it learns of a tag t_v that is not currently in $T_u(i, j)$, it will put it aside and then add it to this set once the rounds dedicated to bin j in this phase conclude.

We have only so far described what node u does during the first ℓ rounds for each block in our fixed instance j. During the remaining $\log N$ rounds in these blocks, u will attempt to disseminate the actual tokens corresponding to the tags advertised (here we emphasize the difference between spelling out the bits of a tag using advertising bits and actually transmitting a token, which requires two nodes to form a connection). In more detail, u executes the PPUSH rumor spreading strategy from [10] (see above) during the last $\log N$ rounds of each block in the current bin. In more detail, for a given block h in this bin, if u advertised tag t in the first ℓ rounds of this block, *and* u actually has the token corresponding to tag t in Q_u, it executes PPUSH in the remaining rounds of this block using this token as the rumor and advertising 1 (i.e., it runs PPUSH with the status of an already informed node). Otherwise, node u runs PPUSH advertising 0 (i.e., it runs the PPUSH as an uniformed node).

Increasing Size Estimates. A core behavior in this algorithm is how nodes upgrade their current estimate of the value k (stored in est_u for each node u). As described above, each node initializes their estimate to 1. As described below, these estimates can only grow during an execution. We call an increase in this estimate at a given node an *upgrade*. There are two events that trigger an upgrade at a given node u.

The first event is that node u sees "activity" on an instance $i' > est_u$, where est_u is its current estimate. The term "activity" in this context means seeing a 1-bit advertised in an instance i' round. If this event occurs, then u knows that some other node has already increased its estimate beyond est_u, so u should upgrade its estimate as well. The second event is that node u fills a bin in its current estimate. That is, there is some bin j such that $|T(est_u, j)| \geq \gamma \log N$. We call this event a *crowded bin*, and u can use this as evidence that est_u does not have enough bins for the number of tags in the system and therefore est_u is too small of an estimate for k. If this event occurs, u will increase est_u by 1 (unless est_u is already at its maximum value in which case it will remain unchanged.).

Recall, as specified above, that if a node u increases its estimate est_u to a new value, it will complete the phase of whatever instance it was participating in before switching to the new estimate moving forward. This restriction simplifies the algorithmic analysis.

6.2 Analysis

Our goal in this section is to prove the below theorem regarding CrowdedBin. Some proof details have been deferred to the full version of this paper [22].

Theorem 6.3. *The CrowdedBin gossip algorithm solves the gossip problem in $O((1/\alpha)k \log^6 N)$ rounds when executed with tag length $b = 1$ in a network with stability $\tau = \infty$.*

At the beginning of an execution each node randomly assigns a tag from $\{1, 2, ..., N^\beta\}$ to its token, and then randomly assigns the token to a bin in each of the $\log N$ instances. We call the global collection of these assignments for a given execution a *configuration*. Fix a configuration. We call a given instance i of this configuration, $1 \leq i \leq \log N$, *crowded*, if the configuration has an instance i bin with at least $\gamma \log N$ unique tags assigned to it. The *target* instance for our fixed configuration is the smallest instance i that is *not* crowded. If every instance is crowded, then we say the *target* instance is *undefined*. We begin our analysis by defining what it means for a configuration to be *good* with respect to these terms:

Definition 6.4. A configuration is *good* if and only if it satisfies the following two properties: (1) every token is assigned a unique tag; and (2) the target instance i is defined, and $k_i \leq 2k$.

A direct corollary of the above definition is that if a configuration is good, and i is the target, then $k_i > k/(\gamma \log N)$. We now bound the probability that the nodes generate a good configuration. We will show that increasing the constant β, used to define the space $\{1, 2, ..., N^\beta\}$ from which tags are drawn, and the constant γ, used to define the number of blocks per bin, increases the high probability that a configuration is good. To make this argument we begin by establishing a non-standard balls-in-bins argument applicable to our specific algorithm's behavior.

Lemma 6.5. *Fix some constant $\gamma \geq 9$. Assume k balls, $1 \leq k \leq N$, are thrown into $k' \geq k$ bins with independent and uniform randomness. The probability that at least one bin has at least $\gamma \log N$ balls, is less than $1/N^{(\gamma/3)-2}$.*

We apply the above balls-in-bin argument to prove the following lemma which argues that good configurations are likely.

Lemma 6.6. *Fix some constant $c \geq 1$. For a tag space constant $\beta \geq c + 3$, and a bin size constant $\gamma \geq 3c + 9$, the nodes generate a good configuration with probability at least $1 - 1/N^c$.*

The below lemma follows directly from the definition of good and the mechanism by which our algorithm updates estimates:

Lemma 6.7. *In an execution with a good configuration with target instance i, no node ever sets its local estimate to a value larger than i. That is, for all u and all rounds, $est_u \leq i$.*

We now continue our analysis by bounding the time required for all nodes to reach the target instance. We do so with two arguments: the first concerning the rounds required for nodes to learn of a larger estimate existing in the system, and the second concerning the rounds required for the largest estimate to increase if it is still less than the target. For the following results, let D describe the diameter of the static network topology:

Lemma 6.8. *Fix an execution with a good configuration with target instance i. Assume that at the beginning of round r of this execution the largest estimate in the system is $i_{max} \leq i$. By round $r' = r + O(Dk_{i_{max}} \log^3 N)$ either: the largest estimate in the system is larger than i_{max}, or all nodes have estimate i_{max}.*

Lemma 6.9. *Fix an execution with a good configuration with target instance i. Assume that at the beginning of round r of this execution the largest estimate in the system is $i_{max} < i$. By round $r' = r + O(Dk_{i_{max}} \log^3 N)$ the largest estimate in the system is larger than i_{max}.*

We now leverage Lemmas 6.8 and 6.9 to bound the rounds required for all nodes to permanently stabilize to the target instance.

Lemma 6.10. *Fix an execution with a good configuration with target instance i. By round $r = O(Dk_i \log^3 N)$, every node has estimate i. That is, for every node u, $est_u = i$ by round r.*

Proof. By the definition of our algorithm, estimates never decrease. By Lemma 6.7, no node will ever adopt an estimate greater than i. Combined, it follows that we can keep applying Lemma 6.9 to increase the largest estimate until the largest estimate reaches i. We can then apply a single instance of Lemma 6.8 to ensure all nodes have this estimate, permanently satisfying the lemma.

To bound the time required for these applications of the above lemmas, we leverage our observation that the largest estimate can only increase. It follows that in the worst case we apply Lemma 6.9 exactly once for each of the estimates leading up to the target i. Because these estimates form a geometric sequence (e.g., 2, 4, 8, ...), the total rounds needed for these applications of Lemma 6.9 is upper bounded by:

$$\begin{aligned} O(Dk_1 \log^3 N) + ... + O(Dk_i \log^3 N) &= \\ O\left((D \log^3 N)(k_1 + k_2 + ... + k_i)\right) &= O(Dk_i \log^3 N) \end{aligned}$$

The final application of Lemma 6.8 to spread estimate i to all remaining nodes once it exists in the system adds only a single an additional $O(Dk_i \log^3 N)$ rounds. The lemma statement follows. □

The preceding arguments bound the rounds required for useful information to propagate through the network via the nodes' advertising bits. We now conclude our proof by turning our attention to the rounds required for the actual tokens (which must be passed one at a time through pairwise connections) to spread.

Proof of Theorem 6.3. Assume for now that the configuration is good and i its target instance. Let round $r = O(Dk_i \log^3 N)$ be the round specified by Lemma 6.10 for the network to converge its estimate. That is, every node has the same estimate i by round r. By definition, no bin is crowded for instance i in a good configuration. It follows that *every* tag for *every* bin in this instance will be spread in *every* round by the nodes that know that tag in that round. Following the same propagation arguments used in Lemmas 6.8 and 6.9, after at most D more phases of instance i, all nodes will know all tags. This requires at most $O(Dk_i \log^3 N)$ rounds. Therefore by some round $r' = O(Dk_i \log^3 N)$, the system will have reached a *stable* state in which every node has the same estimate i and knows the tag for every token in the system. This information will never again change so we can turn our attention for the rounds required to finish propagating the actual tokens after this point of stabilization.

To bound this token propagation time, fix an arbitrary token t with tag q in instance i. Because we assume the system has stabilized, every node has q assigned to the same block of the same bin in their instance i phase. It follows that if we append together the last $\log N$ rounds from these blocks (i.e., the rounds in which nodes run PPUSH for the tag described in the first ℓ rounds of the block), we obtain a proper execution of PPUSH rumor spreading for token t during these rounds. That is, every time we come to the last $\log N$ rounds of q's block, all nodes are running PPUSH for rumor t, picking up where they left off in the previous instance.

Applying Theorem 6.1 from above, it follows that with high probability in N, $O(\log^4 N/\alpha)$ rounds are sufficient for t to spread to all nodes after stabilization. Each phase provides $\log N$ rounds of PPUSH, so $O(\log^3 N/\alpha)$ phases are sufficient after stabilization.

The key observation is that each execution of instance i services *all* k rumors after stabilization, as each rumor has its own fixed bin in the instance i phase. Therefore, $O(\log^3 N/\alpha)$ phases are sufficient to spread *all* k rumors in parallel. A union bound establishes that all $k \leq N$ instances succeed with a slightly reduced high probability.

From a probability perspective, we know from Lemma 6.6 that the configuration is good with high probability. We just argued above that if the configuration is good, then with an additional high probability the tokens will all spread in the stated time, once the system stabilizes. We can increase both high probabilities to the desired exponent by increasing the constant β and γ used in the definition of crowded bins, and the constant factor in the time bound for PPUSH. A union bound then shows that both good events occur with high probability.

From round cost perspective, we established that the time to stabilization is at most $O(Dk_i \log^3 N)$ rounds, while the time to complete propagation after stabilization is at most $O(\log^3 N/\alpha)$ instance i phases, which each require $O(k_i \log^3 N)$ rounds. The final time complexity is then in: $O(Dk_i \log^3 N + (k_i \log^6 N)/\alpha)$.

By the definition of a good configuration, we know $k_i \leq 2k$, and by Theorem 6.2, we know $D = O(\log N/\alpha)$. We can therefore simplify this complexity to $O((k \log^6 N)/\alpha)$ rounds, as required. □

7 ϵ-GOSSIP WITH 1-BIT TAGS AND DYNAMIC TOPOLOGY

Here we consider ϵ-Gossip: a relaxed version of the gossip problem that is parameterized with some ϵ, $0 < \epsilon < 1$ (e.g., as also studied in [5]). In more detail, the problem assumes all n nodes start with a token. To solve ϵ-gossip there must be a subset S of the n nodes in the system, where $|S| \geq \epsilon n$ *and* for every $u, v \in S$, u knows v's token and v knows u's token. Our goal here is to prove that for reasonably well-connected graphs and constant ϵ, *almost* solving gossip can be significantly faster than *fully* solving gossip.

Crucially, we do not present a new algorithm to tackle this new problem. We instead reanalyze SharedBit gossip to study how quickly it achieves ϵ-Gossip for a given ϵ bound. Our goal is to prove the following result (in the full version of this paper [22] we also prove a corollary about SimSharedBit achieving a similar result without requiring shared randomness):

Theorem 7.1. *Fix some ϵ, $0 < \epsilon < 1$. The SharedBit gossip algorithm solves the ϵ-gossip problem in $O\left((n\sqrt{\Delta \log \Delta})/((1-\epsilon)\alpha)\right)$ rounds when executed with shared randomness and tag length $b = 1$ in a network with stability $\tau \geq 1$.*

Given that $\Delta \leq n$, this new time complexity is faster than the $O(n^2)$ rounds required by SharedBit (for $k = n$) when ϵ is a constant fraction and $\alpha = \omega(\log \Delta/(\sqrt{\Delta \log \Delta}))$.

The full details of this result are provided in the full version of this paper [22]. The core idea behind this analysis is to first prove that in any round such that the problem is not solved, there exists a *coalition* of nodes such that: (1) the size of the coalition is $\approx \epsilon n$; and (2) no node *not in* the coalition has the same token set as a node *in* the coalition. Because of the large size of this coalition, we can then prove that there exist many links from this coalition to nodes outside its boundaries. By definition, each of these links, if transformed into a connection, would productively transfer a token. We can then study the expected number of these links to transform in this manner. Put another way, for constant ϵ, this proof leverages the intuition that until a constant fraction of the messages have spread, there is potential for lots of spreading.

REFERENCES

[1] Scott Burleigh, Adrian Hooke, Leigh Torgerson, Kevin Fall, Vint Cerf, Bob Durst, Keith Scott, and Howard Weiss. 2003. Delay-tolerant networking: an approach to interplanetary internet. *IEEE Communications Magazine* 41, 6 (2003), 128–136.

[2] Daniel Camps-Mur, Andres Garcia-Saavedra, and Pablo Serrano. 2013. Device-to-device communications with Wi-Fi Direct: overview and experimentation. *IEEE wireless communications* 20, 3 (2013), 96–104.

[3] Flavio Chierichetti, Silvio Lattanzi, and Alessandro Panconesi. 2010. Rumour Spreading and Graph Conductance.. In *Proceedings of the ACM-SIAM symposium on Discrete Algorithms (SODA)*.

[4] Sebastian Daum, Fabian Kuhn, and Yannic Maus. 2016. Rumor Spreading with Bounded In-Degree. In *International Colloquium on Structural Information and Communication Complexity (SIRROCO)*.

[5] Shlomi Dolev, Seth Gilbert, Rachid Guerraoui, and Calvin Newport. 2007. Gossiping in a multi-channel radio network. In *Proceedings of the Symposium on Distributed Computing (DISC)*.

[6] Nikolaos Fountoulakis and Konstantinos Panagiotou. 2010. Rumor spreading on random regular graphs and expanders. In *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques*. Springer, 560–573.

[7] Alan M Frieze and Geoffrey R Grimmett. 1985. The shortest-path problem for graphs with random arc-lengths. *Discrete Applied Mathematics* 10, 1 (1985), 57–77.

[8] Alan M Frieze and Geoffrey R Grimmett. 1985. The shortest-path problem for graphs with random arc-lengths. *Discrete Applied Mathematics* 10, 1 (1985), 57–77.

[9] Open Garden. Accessed: Feb., 2017. FireChat Phone-to-Phone App. (Accessed: Feb., 2017). http://www.opengarden.com/FireChat.

[10] Mohsen Ghaffari and Calvin Newport. 2016. How to Discreetly Spread a Rumor in a Crowd. In *Proceedings of the International Symposium on Distributed Computing (DISC)*.

[11] George Giakkoupis. 2011. Tight bounds for rumor spreading in graphs of a given conductance. In *Proceedings of the Symposium on Theoretical Aspects of Computer Science (STACS)*.

[12] George Giakkoupis. 2011. Tight bounds for rumor spreading in graphs of a given conductance. In *Proceedings of the Symposium on Theoretical Aspects of Computer Science (STACS)*.

[13] George Giakkoupis. 2014. Tight bounds for rumor spreading with vertex expansion. In *Proceedings of the ACM-SIAM Symposium on Discrete Algorithms (SODA)*.

[14] George Giakkoupis and Thomas Sauerwald. 2012. Rumor spreading and vertex expansion. In *Proceedings of the ACM-SIAM symposium on Discrete Algorithms (SODA)*. 1623–1641.

[15] George Giakkoupis and Thomas Sauerwald. 2012. Rumor spreading and vertex expansion. In *Proceedings of the ACM-SIAM symposium on Discrete Algorithms (SODA)*. SIAM, 1623–1641.

[16] Carles Gomez, Joaquim Oller, and Josep Paradells. 2012. Overview and evaluation of bluetooth low energy: An emerging low-power wireless technology. *Sensors* 12, 9 (2012), 11734–11753.

[17] Thiagaraja Gopalsamy, Mukesh Singhal, D Panda, and P Sadayappan. 2002. A reliable multicast algorithm for mobile ad hoc networks. In *Proceedings of the IEEE International Conference on Distributed Computing Systems (ICDCS)*. IEEE, 563–570.

[18] Ericsson Inc. Accessed: Feb., 2017. Latest mobile statistics: key figures (Ericsson Mobility Report). (Accessed: Feb., 2017). https://www.ericsson.com/mobility-report/latest-mobile-statistics.

[19] Fabian Kuhn, Nancy Lynch, and Rotem Oshman. 2010. Distributed computation in dynamic networks. In *Proceedings of the Symposium on Principles of Distributed Computing (PODC)*. ACM, 513–522.

[20] David Mark, Jayant Varma, Jeff LaMarche, Alex Horovitz, and Kevin Kim. 2015. Peer-to-Peer Using Multipeer Connectivity. In *More iPhone Development with Swift*. Springer, 239–280.

[21] Ilan Newman. 1991. Private vs. common random bits in communication complexity. *Information processing letters* 39, 2 (1991), 67–71.

[22] Calvin Newport. 2017. Gossip in a Smartphone Peer-to-Peer Network. In *Proceedings of the Symposium on Principles of Distributed Computing (PODC)*. Full version available on arXiv and at: http://people.cs.georgetown.edu/~cnewport/pubs/gossipmobile-full.pdf.

[23] Calvin Newport. 2017. Leader Election in a Smartphone Peer-to-Peer Network. In *Proceedings of the IEEE International Parallel and Distributed Processing Symposium (IPDPS)*. Full version available online at: http://people.cs.georgetown.edu/~cnewport/pubs/le-IPDPS2017.pdf.

[24] Devavrat Shah et al. 2009. Gossip algorithms. *Foundations and Trends in Networking* 3, 1 (2009), 1–125.