SIAM J. COMPUT.
Vol. 46, No. 1, pp. 58–79



# RUMOR SPREADING WITH NO DEPENDENCE ON CONDUCTANCE*

KEREN CENSOR-HILLEL†, BERNHARD HAEUPLER‡, JONATHAN KELNER§, AND PETAR MAYMOUNKOV¶

**Abstract.** In this paper, we study how a collection of interconnected nodes can efficiently perform a global computation in the $\mathcal{GOSSIP}$ model of communication. In this model nodes do not know the global topology of the network and may only initiate contact with a single neighbor in each round. This contrasts with the much less restrictive $\mathcal{LOCAL}$ model, where a node may simultaneously communicate with all of its neighbors in a single round. A basic question in this setting is how many rounds of communication are required for the information dissemination problem, in which each node has some piece of information and is required to collect all others. In the $\mathcal{LOCAL}$ model this is quite simple: each node broadcasts all of its information in each round, and the number of rounds required will be equal to the diameter of the underlying communication graph. In the $\mathcal{GOSSIP}$ model, each node must independently choose a single neighbor to contact, and the lack of global information makes it difficult to make any sort of principled choice. As such, researchers have focused on the *uniform gossip algorithm*, in which each node independently selects a neighbor uniformly at random. When the graph is well-connected, this works quite well. In a string of beautiful papers, researchers proved a sequence of successively stronger bounds on the number of rounds required in terms of the conductance $\phi$ and graph size $n$, culminating in a bound of $\Theta(\phi^{-1}\log n)$. In this paper, we give the first protocol that works efficiently on any topology. In particular we give an algorithm that solves the information dissemination problem in at most $O(D+\text{polylog}(n))$ rounds in a network of diameter $D$, with *no dependence on the conductance*. This is at most an additive polylogarithmic factor from the trivial lower bound of $D$. In fact, we prove that something stronger is true: *any* algorithm that requires $T$ rounds in the $\mathcal{LOCAL}$ model can be simulated in $O(T+\text{polylog}(n))$ rounds in the $\mathcal{GOSSIP}$ model. We thus prove that these two models of distributed computation are equivalent up to an additive polylogarithmic term.

**Key words.** gossip model, local model, conductance decomposition, sparse spanners, information spreading

**AMS subject classification.** 68W15

**DOI.** 10.1137/14099992X

## 1. Introduction.

Many distributed applications require nodes of a network to perform a global task using only local knowledge. Typically a node initially only knows the identity of its neighbors and gets to know a wider local neighborhood in the underlying communication graph by repeatedly communicating with its neighbors. Among the most important questions in distributed computing is how certain global computation problems, e.g., computing a maximal independent set or a graph coloring, can be performed with such local constraints.

*Received by the editors December 15, 2014; accepted for publication (in revised form) November 2, 2016; published electronically January 31, 2017. A preliminary version of this paper appeared in *Proceedings of the 44th Symposium on Theory of Computing*, 2012.
http://www.siam.org/journals/sicomp/46-1/99992.html
**Funding:** The first author was partially supported by ISF grant 1696/14 and BSF grant 2015803. Part of this work done while the first author was at MIT and supported by the Simons Postdoctoral Fellows Program and NSF award CCF-0939370. The second author was partially supported by NSF award CCF-1527110 and CCF-1618280. The third author was partially supported by NSF grant CCF-0843915.

†Department of Computer Science, Technion, Haifa 32000, Israel (ckeren@cs.technion.ac.il).
‡Department of Computer Science, CMU, Pittsburgh, PA 15213 (haeupler@cs.cmu.edu).
§School of Engineering, MIT, Cambridge, MA 02138 (kelner@mit.edu).
¶The Go Circuit Project, 49 Kademlia st, Sofia 1505, Bulgaria (p@gocircuit.org).

Many upper and lower bounds for distributed tasks are given for the well-known $\mathcal{LOCAL}$ model [35, Chapter 2], which operates in synchronized rounds and allows each node in each round to exchange messages of unbounded size with all of its neighbors. It is fair to say that the $\mathcal{LOCAL}$ model is essentially the established minimal requirement for a distributed algorithm. Indeed, whenever a distributed algorithm is said to have running time $T$ it is implied that, at the least, there exists a $T$-round algorithm in the $\mathcal{LOCAL}$ model.

In many settings, practical system design or physical constraints do not allow a node to contact all of its (potentially very large number of) neighbors at once. In this paper we focus on this case and consider the $\mathcal{GOSSIP}$ model, which restricts each node to initiate at most one *bidirectional* communication with one of its neighbors per round. Notice that although each node can initiate communication over at most one connection in a round, it may communicate with several nodes, via communications initiated by those nodes. As in the $\mathcal{LOCAL}$ model and additional standard models of distributed computing, the complexity measure is the amount of communication an algorithm requires, while computation within each node is negligible. However, in contrast to computations in the $\mathcal{LOCAL}$ model, algorithms for the $\mathcal{GOSSIP}$ model have to decide which neighbor to contact in each round. This is particularly challenging when the network topology is unknown.

Algorithms with such *gossip constraints* have been intensively studied for the so-called RUMOR problem (also known as the *rumor spreading* or *information dissemination* problem), in which each node has some initial input and is required to collect the information of all other nodes. Most previous papers analyzed the simple `UniformGossip` algorithm, which chooses a random neighbor to contact in each round. One important property of this algorithm is its inherent robustness against failures. Moreover, the uniform gossip mixes well on well-connected graphs, and good bounds for its convergence in terms of the graph conductance and vertex expansion have been given [7, 23, 25, 34, 40]. In general, however, uniform gossip has a tendency to repeatedly communicate between well-connected neighbors while not transmitting information across bottlenecks. In fact, the expansion assumptions of all prior works are needed to guarantee that the network topology contains no bottlenecks on which the distributed gossip protocol would get stuck. Only recently have algorithms been designed that try to efficiently deal with bottlenecks in the topology and avoid this behavior [6]. Still, up to the conference version of this work all running time bounds of algorithms depended crucially on notions of expansion of the underlying graph. In fact, for all known gossip algorithms there exist many topologies with small, e.g., say, logarithmic, diameter on which a trivial $\Omega(n)$ number of rounds are needed for the gossip algorithm to complete the information dissemination.

**1.1. Our results.** This paper significantly improves the state of the art by providing the first information spreading algorithm for the $\mathcal{GOSSIP}$ model that efficiently deals with *any* bottleneck and is fast for *all* graphs, with no dependence on their conductance. Our algorithm requires at most $O(D + \text{polylog}(n))$ rounds in a network of size $n$ and diameter $D$. This is at most an additive polylogarithmic factor from the trivial lower bound of $\Omega(D)$ rounds even for the $\mathcal{LOCAL}$ model.

In addition, our results apply more generally to any algorithm in the $\mathcal{LOCAL}$ model. We show how any algorithm that takes $T$ time in the $\mathcal{LOCAL}$ model can be simulated in the $\mathcal{GOSSIP}$ model in $O(T + \text{polylog}(n))$ time, thus incurring only an additional cost which is polylogarithmic in the size of the network $n$. Our main result that leads to this simulation is an algorithm for the $\mathcal{GOSSIP}$ model in which each

node exchanges information (perhaps indirectly) with each of its neighbors within a polylogarithmic number of rounds. This holds for every graph, despite the possibility of large degrees. A key ingredient in this algorithm is a recursive decomposition of graphs into clusters of sufficiently large conductance, allowing fast (possibly indirect) exchange of information between nodes inside clusters. The decomposition guarantees that the number of edges between pairs of nodes that did not exchange information decreases by a constant fraction. To convert the multiplicative polylogarithmic overhead for each simulated round into the additive overhead in our final simulation result we show connections between sparse graph spanners and algorithms in the $\mathcal{GOSSIP}$ model. This allows us to simulate known constructions of nearly additive sparse spanners [38], which then in turn can be used in our simulations for even more efficient communication. Instead of making expansion assumptions which define away hard to deal with bottlenecks this work provides the first algorithm which implicitly identifies and successfully overcomes any bottleneck in the communication graph topology.

**1.2. Our techniques.** The key step in our approach is to devise a distributed subroutine in the $\mathcal{GOSSIP}$ model to efficiently simulate one round of the $\mathcal{LOCAL}$ model by a small number of $\mathcal{GOSSIP}$ rounds. In particular, the goal is to deliver each node's current messages to all of its neighbors, which we refer to as the NeighborExchange problem. Indeed, we exhibit such an algorithm, called `Superstep`, which requires at most $O(\log^3 n)$ rounds in the $\mathcal{GOSSIP}$ model for *all* graphs:

Theorem 1.1. *The* `Superstep` *algorithm solves* NeighborExchange *in the* $\mathcal{GOSSIP}$ *model in* $O(\log^3 n)$ *rounds, with high probability.*

Our design for the `Superstep` algorithm was inspired by ideas from [6] and started with an attempt to analyze the following very natural algorithm for the NeighborExchange problem: In each round each node contacts a random neighbor whose message is not yet known to it. While this algorithm works well on some graphs, there exist graphs on which it requires a long time to complete due to asymmetric propagation of messages. We give an explicit example and discuss this issue in section 6.

The `Superstep` algorithm is simple and operates by repeatedly performing $O(\log^3 n)$ rounds of the `UniformGossip` algorithm while eliminating some edges after each round. It is divided in to $O(\log n)$ iterations, during which `UniformGossip` has each node choose a random neighbor to contact and exchange messages for $O(\log^2 n)$ rounds, followed by a reversal of the message exchanges to maintain symmetry. From [7, 23] it is known that all pairs of vertices (and in particular all pairs of neighbors) that lie inside a high-conductance subset of the underlying graph exchange each other's messages within a single iteration. An existential graph decomposition result, given in Corollary 3.4, shows that for any graph there is a decomposition into high-conductance clusters with at least a constant fraction of intracluster edges. This implies that the number of remaining message exchanges required decreases by a constant factor in each iteration, which results in a logarithmic number of iterations until NeighborExchange is solved.

This gives a simple algorithm for solving the Rumor problem, which requires all nodes to receive the messages of all other nodes: By iterating `Superstep` $D$ times, where $D$ is the diameter of the network, one obtains an $O(D \cdot \log^3 n)$ round algorithm. This is at most an $O(\log^3 n)$-factor slower than the trivial diameter lower bound and is a drastic improvement compared to prior upper bounds [6, 7, 23, 34], which can be of order $O(n)$ even for networks with constant or logarithmic diameter $D$.

Beyond the Rumor problem, it is immediate that the NeighborExchange

problem bridges the gap between the $\mathcal{LOCAL}$ and $\mathcal{GOSSIP}$ models in general. Indeed, we can simply translate a single round of a $\mathcal{LOCAL}$ algorithm into the $\mathcal{GOSSIP}$ model by first using any algorithm for NeighborExchange to achieve the local broadcast and then performing the same local computations. We call this a simulation and more generally define an $(\alpha(G), \beta(G))$-*simulator* as a transformation that takes any algorithm in the $\mathcal{LOCAL}$ model that runs in $T(G)$ rounds if the underlying topology is $G$, and outputs an equivalent algorithm in the $\mathcal{GOSSIP}$ model that runs in $O(\alpha(G)) \cdot T(G) + O(\beta(G))$ rounds. Thus, the simulation based on the `Superstep` algorithm gives a $(\log^3 n, 0)$-simulator.

However, we show that we can even obtain a $(1, \text{polylog}(n))$-simulator. This implies an $O(D + \text{polylog}(n))$-round rumor spreading algorithm (i.e., with only a linear dependence on $D$). First, we argue that in many natural graph classes, like graphs with bounded genus or excluded minors, one can do better. Indeed we give a simple argument that on any (sparse) graph with *hereditary density*[1] $\delta$ there is a schedule of direct message exchanges such that NeighborExchange is achieved in $2\delta$ rounds. Furthermore, a schedule with $O(\delta)$ direct exchanges can be computed in $\delta \log n$ rounds of the $\mathcal{GOSSIP}$ model even if $\delta$ is not known. This leads to a $(\delta, \delta \log n)$-simulator. We call this algorithm `DirectExchange`.

Theorem 1.2. *For any $\epsilon > 0$, the deterministic algorithm* `Direct Exchange` *solves the* NeighborExchange *problem in the $\mathcal{GOSSIP}$ model using $O(\frac{\delta \log n}{\epsilon^2})$ rounds, where $\delta$ is the hereditary density of the underlying topology. During the algorithm, each node initiates at most $2(1+\epsilon)^2\delta$ exchanges.*

Another way to look at this is that communicating over any hereditary sparse graph remains fast in the $\mathcal{GOSSIP}$ model. Thus, for a general graph, if one knows a sparse subgraph that has short paths from any node to its neighbors, one can solve the NeighborExchange problem by communicating via these paths. Such graphs have been intensely studied and are known as spanners [36, 37].

Indeed, we show interesting connections between simulators and spanners. For one, any fast algorithm for the NeighborExchange problem induces a sparse low-stretch spanner. The `Superstep` algorithm can thus be seen as a new spanner construction in the $\mathcal{GOSSIP}$ model with the interesting property that the total number of messages used is at most $O(n \log^3 n)$. To our knowledge this is the first such construction. The fact that any fast algorithm for the NeighborExchange problem induces a sparse low-stretch spanner, together with a known lower bound on the stretch of sparse spanners [36], also implies that, in general, NeighborExchange requires a logarithmic number of rounds (up to $\log \log n$ factors perhaps) in the $\mathcal{GOSSIP}$ model.

Considering in the other direction, we show that any fast spanner construction in the $\mathcal{LOCAL}$ model can be used to further decrease the multiplicative overhead of our $(\log^3 n, 0)$-simulator. Applying this insight to several known spanner constructions [9, 16, 38, 39] leads to our third main theorem.

Theorem 1.3. *Every algorithm in the $\mathcal{LOCAL}$ model which completes with high probability in $T = T(G)$ rounds when run on the topology $G$ can be simulated in the $\mathcal{GOSSIP}$ model in*

[1] The hereditary density of a graph G is the minimal integer $\delta$ such that for every subset of nodes $S$ the subgraph induced by $S$ has at most $\delta|S|$ edges.

$$O(1) \cdot \min \left\{ \begin{array}{lll} T \cdot \log^3 n, & & \\ T \cdot 2^{\log^* n} \log n & + \log^4 n, & \\ T \cdot \log n & + 2^{\log^* n} \log^4 n, & \\ T & + \log^{O(1)} n, & \\ T \cdot \delta & + \delta \log n, & \\ T \cdot \Delta & & \end{array} \right\}$$

*rounds with high probability, where $n$ is the number of nodes, $\Delta$ the maximum degree and $\delta$ the hereditary density of $G$.*

When we apply this result to the greedy algorithm for the RUMOR problem, where $T = D$, we obtain an algorithm whose $O(D + \text{polylog}(n))$ rounds are optimal up to the additive polylogarithmic term, essentially closing the gap to the known trivial lower bound of $\Omega(D)$.

COROLLARY 1.4. *There is a $\mathcal{GOSSIP}$ algorithm which for any network solves the information dissemination problem in $O(D + \text{polylog}(n))$ rounds with high probability, where $D$ and $n$ are the diameter and number of nodes of the underlying topology, respectively.*

We now give a rough description of how we obtain the $O(D+\text{polylog}(n))$ complexity of rumor spreading. First, we use our $O(\log^3 n)$ NEIGHBOREXCHANGE algorithm to simulate a spanner construction from the $\mathcal{LOCAL}$ model. This takes $O(\text{polylog}(n))$ rounds. The particular spanner we use stretches distances by a constant multiplicative factor plus a polylogarithmic additive factor. In addition, it has a constant hereditary density. Once we have this spanner, we show how to orient its edges in a logarithmic number of rounds, such that the out-degree of every node is constant. This means that over this spanner we can solve NEIGHBOREXCHANGE in a constant number of rounds, using the `DirectExchange` algorithm. Because this is a spanner with a small stretch, it holds that the real neighbors in the original graph are not too far away, and hence this in turn implies that we can quickly communicate with them by repeatedly communicating over the oriented spanner. This allows us to simulate any $T$-round algorithm in the $\mathcal{LOCAL}$ model in $O(T + \text{polylog}(n))$ rounds in the $\mathcal{GOSSIP}$ model, giving the claimed result for rumor spreading.

We give preliminaries in section 2, and in section 3 we show how to solve the NEIGHBOREXCHANGE problem in $O(\log^3 n)$ rounds. Addressing graphs with small hereditary density is done in section 4, and in section 5 we show how to simulate a spanner construction and use it for simulating arbitrary algorithms even faster. We conclude in sections 6 and 7 with a discussion and a summary of this work.

**1.3. Related work.** The problem of spreading information in a distributed system was introduced by Demers et al. [8] for the purpose of replicated database maintenance, and it has been extensively studied thereafter.

One fundamental property of the distributed system that affects the number of rounds required for information spreading is the communication model. The *random phone call* model was introduced by Karp et al. [29], allowing every node to contact one other node in each round. In our setting, this corresponds to the complete graph. This model alone received much attention, such as in bounding the number of calls [11], bounding the number of random bits used [26, 27], bounding the total number of bits [20], and more.

The number of rounds it takes to spread information for the randomized algorithm `UniformGossip`, in which every node chooses its communication partner for the next

round uniformly at random from its set of neighbors, was analyzed in terms of the conductance of the underlying graph by Mosk-Aoyama and Shah [34], by Chierichetti, Lattanzi, and panconesi [7], and later by Giakkoupis [23], whose work showed the optimal bound in terms of conductance, of $O(\frac{\log n}{\Phi(G)})$ rounds, with high probability.

We mention that the uniform randomized algorithm, also referred to as PUSH-PULL in the literature, enjoys two properties that our algorithm does not. The first is that it is naturally robust against failures and the second is that it does not require any memory other than the list of neighbors. Obtaining these properties with the almost optimal complexity of our algorithm is an open problem.

The relationship between the time required for randomized information spreading and the vertex expansion of the underlying graph was studied in [24, 25, 40]. Feige et al. [17] showed in their important paper a bound of $O(\Delta(D + \log n))$ rounds for `UniformGossip` to spread a rumor in a graph of diameter $D$ and maximum degree $\Delta$. Compared with our results, this is efficient for relatively small values of $\Delta$.

Apart from the uniform randomized algorithm, additional algorithms were suggested for spreading information. We give an overview of some of these approaches. Doerr, Friedrich, and Saverwald [14] introduce *quasi-random* rumor spreading, in which a node chooses its next communication partner by deterministically going over its list of neighbors, but the starting point of the list is chosen at random. Results are $O(\log n)$ rounds for a complete graph and the hypercube, as well as improved complexities for other families of graphs compared to the randomized rumor spreading algorithm with uniform distribution over neighbors. This was followed by further analysis of the quasi-random algorithm [15, 18]. A hybrid algorithm, alternating between deterministic and randomized choices [6], was shown to achieve information spreading in the $O(c(\frac{\log n}{\Phi_c(G)} + c))$ round, with high probability, where $\Phi_c(G)$ is the *weak conductance* of the graph, a measure of connectivity of subsets in the graph. Distance-based bounds were given for nodes placed with uniform density in $R^d$ [30, 31], which also address gossip-based solutions to specific problems such as resource location and minimum spanning tree. Doerr, Fouz, and Friedrich [12, 13] have recently presented algorithms for fast information spreading in preferential attachment graphs, which model social networks. Spreading in social networks was also analyzed in [19].

Interesting algorithms for spreading information in *asynchronous* systems were also studied, e.g., by Georgiou et al. [22], and in [3, 43]. Such systems either are semi synchronous, have bounds on the delay of different components, or assume some stochastic behavior of relative processor speeds.

The $\mathcal{LOCAL}$ model of communication, where each node communicates with each of its neighbors in every round, was formalized by Peleg [35]. Information spreading in this model requires a number of rounds which is equal to the diameter of the communication graph. Many other distributed tasks have been studied in this model.

Graph spanners were first introduced in [36, 37] and have been extensively studied since. Distributed constructions were also studied, e.g., in [1, 2, 10, 39]. We are unaware of previous constructions that try to bound the number of messages required for computing sparse spanners, as is implied by our results.

Since publication in [5] the abstractions and ideas developed in this paper have already been used by [28] to give a more robust, deterministic, $O(\log^2 n)$-round NeighborExchange algorithm. This primitive alone does not achieve the *optimal linear dependence* on $D$ provided by the main result of this paper. Instead, information dissemination based on [28] alone requires $O(D \log n + \log^2 n)$ rounds to complete. Combining the faster NeighborExchange of [28] with the the spanner and simulator

machinery from section 5 does not lead to a faster $O(D + \text{polylog}(n))$ information dissemination protocol as the dominating additive cost comes from the unspecified polylogarithmic additive stretch in a spanner construction of [38]. However, it does improve robustness. On the other hand, the derandomization of [28] does not carry over to a $O(D + \text{polylog}(n))$ running time when combined with the techniques of this paper as the spanner construction of [38] is randomized and no deterministic alternative is known.

## 2. Preliminaries and definitions.

**2.1. The `UniformGossip` algorithm.** The `UniformGossip` algorithm is a common algorithm for RUMOR. (It is also known as the PUSH-PULL algorithm in some papers, such as [23].) Initially, each vertex $u$ has some message $M_u$. At each step, vertex $u$ chooses a random incident edge $(u, v)$ at which point $u$ and $v$ exchange all messages currently known to them. The process stops when all vertices know everyone's initial messages. In order to treat this process formally, for any fixed vertex $v$ and its message $M_v$, we treat the set of vertices that know $M_v$ as a set that evolves probabilistically over time, as we explain next.

We begin by fixing an ambient graph $G = (V, E)$, which is connected, unweighted, and undirected. The `UniformGossip` process is a Markov chain over $2^V$, the set of vertex subsets of $G$. Given a current state $S \subseteq V$, one transition is defined as follows. Every vertex $u$ picks an incident outgoing edge $a_u = (u, w) \in E$ uniformly at random from all such candidates. Let us call the set of all chosen edges $A = \{a_u : u \in V\}$ an *activated set*. Further let $A^\circ = \{(u, w) : (u, w) \in A \text{ or } (w, u) \in A\}$ be the symmetric closure of $A$. The new state of the chain is given by $S \cup B$, where by definition a vertex $v$ is in the *boundary* set $B$ if and only if there exists $u \in S$ such that $(u, v) \in A^\circ$. Note that $V$ is the unique absorbing state, assuming a nonempty start set.

We say that an edge $(u, w)$ is *activated* if $(u, w) \in A^\circ$. If we let $S$ model the set of nodes in possession of the message $M_v$ of some fixed vertex $v$ and we assume bidirectional message exchange along activated edges, the new state $S \cup B$ (of the Markov process) actually describes the set of nodes in possession of the message $M_v$ after one distributed step of the `UniformGossip` algorithm.

Consider a $\tau$-step Markov process $K$, whose activated sets at each step are respectively $A_1, \ldots, A_\tau$. Let the *reverse* of $K$, written $K^{\text{rev}}$, be the $\tau$-step process defined by the activated sets $A_\tau, \ldots, A_1$, in this order. For a process $K$, let $K(S)$ denote the end state when started from $S$.

Without loss of generality, for our analysis we will assume that only a single "starting" vertex $s$ has an initial message $M_s$. We will be interested in analyzing the number of rounds of `UniformGossip` that ensure that all other vertices learn $M_s$, which we call the *broadcast time*. Clearly, when more than one vertex has an initial message, the broadcast time is the same since all messages are exchanged in parallel.

LEMMA 2.1 (reversal lemma). *If $u \in K(\{w\})$, then $w \in K^{\text{rev}}(\{u\})$.*

*Proof.* The condition $u \in K(\{w\})$ holds if and only if there exists a sequence of edges $(e_{i_1}, \ldots, e_{i_r})$ such that $e_{i_j} \in A^\circ_{i_j}$ for all $j$, the indices are increasing in that $i_1 < \cdots < i_r$, and the sequence forms a path from $w$ to $u$. The presence of the reversed sequence in $K^{\text{rev}}$ implies $w \in K^{\text{rev}}(\{u\})$. ∎

In communication terms, the lemma says that if $u$ receives a message originating at $w$ after $\tau$ rounds determined by $K$, then $w$ will receive a message originating at $u$ after $\tau$ rounds determined by $K^{\text{rev}}$. Notice that the reversal lemma is deterministic,

in the sense that if $u$ receives the message of $w$ after $\tau$ rounds determined by $K$ then we are *guaranteed* that $w$ receives the message of $u$ after $\tau$ rounds determined by $K^{\text{rev}}$. This is a stronger guarantee than the property given in Lemma 3 in [7], which states that the probability of a node $u$ receiving the message of $w$ in $t$ rounds of `UniformGossip` is equal to the probability of $w$ receiving the message of $u$ in $t$ rounds. The fact that the reversal lemma *promises* symmetry will be crucial for the iterative behavior of our algorithm, in which we make sure that reversed paths are used.

**2.2. Conductance.** The notion of *graph conductance* was introduced by Sinclair [41]. We require a more general version, which we introduce here. We begin with the requisite notation on edge-weighted graphs. We assume that each edge $(u, v)$ has a weight $w_{uv} \in [0, 1]$. For an unweighted graph $G = (V, E)$ and any $u, v \in V$, we define $w_{uv} = 1$ if $(u, v) \in E$ and $w_{uv} = 0$ otherwise. Now, for $S, T \subseteq V$ we set $w(S, T) = \sum_{u \in S, v \in T} w_{uv}$. Note that in this definition it need not be the case that $S \cap T = \emptyset$, so, e.g., $w(S, S)$, when applied to an unweighted graph, counts every edge in $S$ twice. The volume of a set $S \subseteq V$ with respect to $V$ is written as $\text{vol}(S) = w(S, V)$. Sometimes we will have different graphs defined over the same vertex set. In such cases, we will write the identity of the graph as a subscript, as in $\text{vol}_G(S)$, in order to clarify which is the ambient graph (and hence the ambient edge set). Further, we allow self-loops at the vertices. A single loop at $v$ of weight $\alpha$ is modeled by setting $w_{vv} = 2\alpha$, because both ends of the edge contribute $\alpha$.

For a graph $G = (V, E)$ and a cut $(S, T)$ where $S, T \subseteq V$ and $S \cap T = \emptyset$ (but where $T \cup S$ does not necessarily equal all of $V$), the *cut conductance* is given by

$$\varphi(S, T) = \frac{w(S, T)}{\min\left\{ \text{vol}_G(S), \text{vol}_G(T) \right\}}. \tag{1}$$

For a subset $U \subseteq V$ we need to define the *conductance of $U$ (embedded) in $V$*. We will use this quantity to measure how quickly the `UniformGossip` algorithm proceeds in $U$, while accounting for the fact that edges in $(U, V \setminus U)$ may slow down the process. The conductance of $U$ in $G$ is defined by

$$\Phi(U) = \min_{S \subseteq U} \varphi(S, U \setminus S). \tag{2}$$

Note that the classical notion of conductance of $G$ (according to Sinclair [41]) equals $\Phi(V)$ in our notation. When we want to emphasize the ambient graph $G$ within which $U$ resides, we will write $\Phi_G(U)$.

A few arguments in this paper will benefit from the notion of a "strongly induced" subgraph of a vertex subset of an ambient graph $G$.

DEFINITION 2.2. *Let $U \subseteq V$ be a vertex subset of $G$. The strongly induced subgraph of $U$ in $G$ is a (new) graph $H$ with vertex set $U$, whose edge weight function $h : U \times U \to \mathbf{R}$ is defined by*

$$h_{uv} = \begin{cases} w_{uv} & \text{if } u \neq v, \\ w_{uu} + \sum_{x \in V \setminus U} w_{ux} & \text{if } u = v. \end{cases}$$

Note that by construction we have $\Phi_H(U) = \Phi_G(U)$. The significance of this notion is the fact that the Markov process, describing the vertex set in possession of some message $M_s$ for a starting vertex $s \in U$ in the `UniformGossip` algorithm executed on the strongly induced subgraph $H$, behaves identically to the respective

process in $G$ observed only on $U$. In particular, this definition allows us to use Theorem 1 of [23] in the following form.

LEMMA 2.3. *For any graph $G = (V, E)$, a vertex subset $U \subseteq V$ and any start vertex in $U$, the broadcast time of the* `UniformGossip` *algorithm on the strongly induced subgraph of $U$ is $O(\Phi_G(U)^{-1} \log n)$ rounds, with high probability.*

**3. NeighborExchange in $\log^3 n$ rounds.** The idea behind our algorithm for solving the NeighborExchange problem is as follows. For every graph there exists a partition into clusters whose conductance is high, and therefore the `UniformGossip` algorithm allows information to spread quickly in each cluster. The latter further implies that pairs of neighbors inside a cluster exchange their messages quickly (perhaps indirectly). What remains is to exchange messages across intercluster edges. This is done recursively. In the following subsection we describe the conductance decomposition and then in subsection 3.2 we give the details for the algorithm together with the proof of correctness.

**3.1. Conductance decomposition of a graph.** As described, our first goal is to partition the graph into clusters with large conductance while limiting the number of intercluster edges (otherwise, having each node in a separate cluster would be a trivial solution). We are going to achieve this in the following lemma, whose proof is very similar to that of Theorem 7.1 in [42]. Note that for our eventual algorithm, we are only going to need an existential proof of this clustering and not an actual algorithm for finding it.

LEMMA 3.1. *Let $S \subseteq V$ be of maximum volume such that* $\mathrm{vol}(S) \leq \mathrm{vol}(V)/2$ *and $\varphi(S, V \setminus S) \leq \xi$, for a fixed parameter $\xi \geq \Phi(G)$. If* $\mathrm{vol}(S) \leq \mathrm{vol}(V)/4$, *then* $\Phi(V \setminus S) \geq \xi/3$.

*Proof.* Assume, toward a contradiction, that $\Phi(V \setminus S) < \xi/3$. Then, there exists a cut $(P, Q)$ of $V \setminus S$ with $\varphi(P, Q) < \xi/3$ and specifically

$$\max\left\{\frac{w(P,Q)}{\mathrm{vol}(P)}, \frac{w(P,Q)}{\mathrm{vol}(Q)}\right\} \leq \frac{\xi}{3}. \tag{3}$$

Henceforth, let $Q$ be the smaller of the two, that is, $\mathrm{vol}(Q) \leq \mathrm{vol}(V \setminus S)/2$.

We are going to show that $\varphi(S \cup Q, P) \leq \xi$ and either $S \cup Q$ or $P$ should have been chosen instead of $S$.

Consider the case $\mathrm{vol}(S \cup Q) \leq \mathrm{vol}(V)/2$. In this case,

$$\begin{aligned}
\varphi(S \cup Q, P) &= \frac{w(S,P) + w(Q,P)}{\mathrm{vol}(S \cup Q)} = \frac{w(S,P) + w(Q,P)}{\mathrm{vol}(S) + \mathrm{vol}(Q)} \\
&\leq \max\left\{\frac{w(S,P)}{\mathrm{vol}(S)}, \frac{w(Q,P)}{\mathrm{vol}(Q)}\right\} \\
&\leq \max\left\{\frac{w(S,P) + w(S,Q)}{\mathrm{vol}(S)}, \frac{w(Q,P)}{\mathrm{vol}(Q)}\right\} \\
&\leq \max\left\{\xi, \xi/3\right\} = \xi.
\end{aligned}$$

This establishes a contradiction to the maximality of the set $S$, because $\varphi(S \cup Q, P) \leq \xi$ and $\mathrm{vol}(S) < \mathrm{vol}(S \cup Q) \leq \mathrm{vol}(V)/2$.

Now we consider the case $\mathrm{vol}(S \cup Q) > \mathrm{vol}(V)/2$. We argue that $\mathrm{vol}(S \cup Q)$ cannot be too large. We use that $\mathrm{vol}(Q) \leq \frac{1}{2}\mathrm{vol}(V \setminus S) = \frac{1}{2}(\mathrm{vol}(V) - \mathrm{vol}(S))$ and get that

$$\begin{aligned}\operatorname{vol}(S \cup Q) &= \operatorname{vol}(S) + \operatorname{vol}(Q) \leq \operatorname{vol}(S) + \frac{\operatorname{vol}(V) - \operatorname{vol}(S)}{2} \\ &= \frac{\operatorname{vol}(V) + \operatorname{vol}(S)}{2} \leq \frac{5}{8}\operatorname{vol}(V).\end{aligned}$$

Hence, $\operatorname{vol}(P) \geq \frac{3}{8}\operatorname{vol}(V)$. In addition, for the cut size, we have

$$\begin{aligned}w(S \cup Q, P) &= w(S, P) + w(Q, P) \\ &\leq \xi \operatorname{vol}(S) + \frac{\xi}{3}\operatorname{vol}(Q) \\ &\leq \xi \operatorname{vol}(S) + \frac{\xi}{3}\frac{\operatorname{vol}(V) - \operatorname{vol}(S)}{2} \\ &= \frac{5}{6}\xi \operatorname{vol}(S) + \frac{1}{6}\xi \operatorname{vol}(V) \\ &\leq \frac{3}{8}\xi \operatorname{vol}(V).\end{aligned}$$

This allows us to bound the cut conductance by

$$\varphi(S \cup Q, P) = \frac{w(S \cup Q, P)}{\operatorname{vol}(P)} \leq \frac{\frac{3}{8}\xi \operatorname{vol}(V)}{\frac{3}{8}\operatorname{vol}(V)} = \xi. \tag{4}$$

This establishes the desired contradiction because $\varphi(S \cup Q, P) \leq \xi$ while $\operatorname{vol}(S) \leq \frac{1}{4}\operatorname{vol}(V) < \frac{3}{8}\operatorname{vol}(V) \leq \operatorname{vol}(P) \leq \frac{1}{2}\operatorname{vol}(V)$. ∎

Lemma 3.1 says that if a graph has no sparse balanced cuts, then it has a large subgraph which has no sparse cuts. The following corollary establishes that Lemma 3.1 holds even in the case when the ambient graph is itself a subgraph of a larger graph.

COROLLARY 3.2. *Let $U \subseteq V$ and let $S \subseteq U$ be of maximum volume such that $\operatorname{vol}(S) \leq \operatorname{vol}(U)/2$ and $\varphi(S, U \setminus S) \leq \xi$, for a fixed parameter $\xi \geq \Phi(U)$. If $\operatorname{vol}(S) \leq \operatorname{vol}(U)/4$, then $\Phi(U \setminus S) \geq \xi/3$.*

*Proof.* Observe that the proof of Lemma 3.1 holds when the graph has loops, that is, $w_{uu} \neq 0$ for some $u$'s. Let $H$ be the strongly induced graph of $U$. It follows from the definition that for any two disjoint sets $A, B \subseteq U$ we have $\operatorname{vol}_G(A) = \operatorname{vol}_H(A)$ and $w(A, B) = h(A, B)$. We can therefore apply Lemma 3.1 to $H$ and deduce that the statement holds for the respective sets in $G$. ∎

We are now ready to state and analyze our clustering construction. We emphasize that this construction is neither efficient nor distributed but serves merely as a proof of existence of the partition.

The clustering algorithm for a graph $G = (V, E)$ is simply a call to `Cluster(`$G$`,`$V$`,`$\xi$`)` where `Cluster` is the following recursive subroutine:

`Cluster(`$G$`,`$U$`,`$\xi$`):`

*The inputs are a graph $G = (V, E)$, a subset $U \subseteq V$ and a parameter $0 < \xi < 1$.*

1. *Find a subset $S \subseteq U$ of maximum volume such that $\operatorname{vol}(S) \leq \operatorname{vol}(U)/2$ and $\varphi(S, U \setminus S) \leq \xi$.*
2. *If no such $S$ exists, then stop and output a single cluster $\{U\}$. Otherwise,*

3a. *If $\operatorname{vol}(S) \leq \operatorname{vol}(U)/4$, output $\{U \setminus S\} \cup$* `Cluster(`$G$`,`$S$`,`$\xi$`)`.

3b. *If $\operatorname{vol}(S) > \operatorname{vol}(U)/4$, output* `Cluster(`$G$`,`$S$`,`$\xi$`)` $\cup$ `Cluster(`$G$`,`$U \setminus S$`,`$\xi$`)`.

LEMMA 3.3. *For every $0 < \zeta < 1$, every graph $G = (V, E)$ with edge weights $w_{uv} \in \{0\} \cup [1, +\infty)$ has a partition $V = V_1 \cup \cdots \cup V_k$ such that $\Phi(V_i) \geq \frac{\zeta}{\log_{4/3} \operatorname{vol}(V)}$ for all $i$, and $\sum_{i<j} w(V_i, V_j) \leq \frac{3\zeta}{2} \operatorname{vol}(V)$.*

*Proof.* The depth $K$ of the recursion in `Cluster` is, by construction, at most $\log_{4/3} \operatorname{vol}(V)$ assuming that the smallest nonzero weight is 1. Let $\mathcal{R}_i \subseteq 2^V$ be a collection of the $U$-parameters of invocations of `Cluster` at depth $0 \leq i \leq K$ of the recursion. (So, for example, $\mathcal{R}_0 = \{V\}$.) Clearly if $|U| = 1$, the algorithm terminates. For a set $U$ let $S(U)$ be the small side of the cut produced by `Cluster(`$G$`,`$U$`,`$\xi$`)`, or $\emptyset$ if no eligible cut was found. We can then bound the total weight of cut edges as

$$\begin{aligned}\sum_{0\leq i\leq K}\sum_{U\in\mathcal{R}_i} w\big(S(U), U\setminus S(U)\big) &\leq \sum_{0\leq i\leq K}\sum_{U\in\mathcal{R}_i} \xi \operatorname{vol}\big(S(U)\big)\\ &\leq \sum_{0\leq i\leq K}\sum_{U\in\mathcal{R}_i} \frac{\xi}{2}\operatorname{vol}(U) \leq \frac{\xi}{2}\sum_{0\leq i\leq K}\sum_{U\in\mathcal{R}_i} \operatorname{vol}(U)\\ &\leq \frac{\xi}{2}\sum_{0\leq i\leq K} \operatorname{vol}(V) \leq \frac{\xi \log_{4/3}\operatorname{vol}(V)}{2}\operatorname{vol}(V),\end{aligned}$$

where we use the convention $w(\emptyset, S) = 0$. If we set $\xi = \frac{3\zeta}{\log_{4/3} \operatorname{vol}(V)}$, for some $0 < \zeta < 1$, then Corollary 3.2 establishes the lemma. ∎

In this paper, we are going to use the following specialization of this lemma, obtained by plugging in $\zeta = 1/6$.

COROLLARY 3.4. *Every unweighted graph on $m$ edges has a clustering that cuts at most $\frac{m}{2}$ edges and each cluster has conductance at least $\frac{1}{6\log_{4/3} 2m}$.*

**3.2. The `superstep` algorithm for the NEIGHBOREXCHANGE problem.** In this section, we will describe the `Superstep` algorithm, which solves the NEIGHBOREXCHANGE problem. Recall that for this problem, all vertices $v$ are assumed to possess an initial message $M_v$, and the goal is for every pair of neighbors to know each other's initial messages.

We now describe our communication protocol, which specifies a local, per-vertex rule that tells a node which edge to choose for communication at any given round. It is assumed that the node will greedily transmit all messages known to it whenever an edge is chosen for communication. The protocol described here will employ some auxiliary messages, which are needed exclusively for its internal workings.

The `Superstep` subroutine described in Figure 1 is designed to ensure that, after a single invocation, all neighbors $(u, w)$ in an undirected graph $G$ have exchanged each other's initial messages. Clearly then, $D$ invocations of `Superstep`, where $D$ is the diameter of $G$, ensure that a message starting at vertex $v$ reaches all $u \in V$, and this holds for all messages. $D$ invocations of `Superstep` thus resolve the RUMOR problem.

The `Superstep` subroutine receives as input an undirected, unweighted graph $G$ and a parameter $\tau$ which is the number of rounds for which the vertices will choose random neighbors to contact in each iteration (therefore the length of each iteration is $2\tau$). If $E$ is a set of undirected edges, let $\vec{E} = \{(u, w) : \{u, w\} \in E\}$ be the corresponding directed graph. Note that initially $F_0$ is symmetric as it includes all directed edges of $E$. A crucial aspect of the `Superstep` subroutine is that every $F_i$ is symmetric as well.

`Superstep`($G$,$\tau$):
The parameter $G = (V, E)$ is an unweighted, undirected graph, and $\tau$ is a positive integer.
Set $F_0 := \vec{E}$ and $i := 0$. While $F_i \neq \emptyset$, repeat:

1. *(First half)*
   - 1a. Initialize every vertex $v$ with a new auxiliary message $a(v)$, unique to $v$. (This message is added to the set of initial messages that $v$ happens to know currently.)
   - 1b. Perform the `UniformGossip` algorithm with respect to $F_i$ for $\tau$ rounds. And denote the outcome of the random activated edge choices by $K_i$
   - 1c. For every vertex $u$ and neighbor $w$, let $X_{uw}$ be the indicator that $u$ received $a(w)$
2. *(Second half)*
   - 2a. Initialize every vertex $v$ with a new auxiliary message $b(v)$, unique to $v$
   - 2b. Perform $K_i^{\text{rev}}$, the reverse process of the one realized in Step 1b
   - 2c. For every vertex $u$ and neighbor $w$, let $Y_{uw}$ be the indicator that $u$ received $b(w)$
3. *(Pruning)* Compute the set of pruned *directed* edges $P_i = \{(u, w) : X_{uw} + Y_{uw} > 0\}$
4. Set $F_{i+1} := F_i \setminus P_i$ and $i := i + 1$

FIG. 1. *Code for* `Superstep` *algorithm. It is easily verified that the above algorithm can be implemented in the* $\mathcal{GOSSIP}$ *model of communication.*

LEMMA 3.5. *Let $G = (V, E)$ be an undirected, unweighted graph with $|V| = n$ and $|E| = m$. Then, after one invocation of* `Superstep`*($G$,$\tau$), where $\tau = \Theta(\log^2 m)$, the following hold with probability $1 - 1/n^{\Omega(1)}$:*

(i) *Every pair of neighbors $\{u, w\} \in E$ receives each other's messages.*
(ii) *The algorithm performs $\Theta(\log^3 m)$ rounds of communication.*

Finally, our main result, Theorem 1.1, follows as a corollary of Lemma 3.5.

Our proof of Lemma 3.5 is structured as follows. Recall that if $E$ is a set of undirected edges, then $\vec{E} = \{(u, w) : \{u, w\} \in E\}$ is the corresponding directed graph. Let $\vec{E} = F_0, \ldots, F_d = \emptyset$ be the respective edge sets of each iteration in `Superstep`. We are going to show that, with probability $1 - 1/n^{\Omega(1)}$, the following invariants are maintained at each iteration:

(a) The directed edge set $F_i$ is symmetric in the sense that $(u, w) \in F_i \Rightarrow (w, u) \in F_i$.
(b) The size of $F_i$ reduces by a constant factor at each iteration. Formally, $|F_{i+1}| \leq \frac{1}{2}|F_i|$.
(c) After the $i$th iteration, for every $(u, w) \in \vec{E} \setminus F_{i+1}$, vertex $u$ has received the message of vertex $w$ and vice-versa.

In fact, invariants (a) and (c) will be shown to hold always. Since $F_d = \emptyset$, claim (c) implies part (i) of Lemma 3.5. Claim (b) implies that the maximum number of iterations is $\log 2m$. Noting that every iteration entails $2\tau$ distributed rounds establishes part (ii) of Lemma 3.5.

*Proof of claim* (a). Initially, $F_0$ is symmetric by construction. Inductively, assume that $F_i$ is symmetric. The reversal lemma applied to $K_i$ and $K_i^{\text{rev}}$ implies $X_{uw} = Y_{wu}$ for all $u, w \in V$. This in turn implies that $X_{uw} + Y_{uw} = X_{wu} + Y_{wu}$, so $P_i$ is symmetric. Since $F_i$ is symmetric by hypothesis, we can conclude that $F_{i+1} = F_i \setminus P_i$ is symmetric as well. ∎

*Proof of claim* (b). Consider the graph $G_i = (V, F_i)$ on the edge set $F_i$. Since $F_i$ is symmetric, by claim (a), we can treat $G_i$ as undirected for the purposes of analyzing the `UniformGossip` algorithm. Let $V_1 \cup \cdots \cup V_k$ be the decomposition of $G_i$ promised by Corollary 3.4. (Note that the corollary holds for disconnected graphs, which may arise.) We thus have $\Phi(V_j) \geq \frac{1}{6 \log_{4/3} 2m}$ for all $1 \leq j \leq k$.

The choice $\tau = O\big(6 \log_{4/3} 2m \cdot \log m\big)$ ensures, via Lemma 2.3, that the first `UniformGossip` execution in every iteration mixes on all $V_j$ with probability $1 - 1/n^{\Omega(1)}$. Mixing in $V_j$ means that for every internal edge $(u, w)$, where $u, w \in V_j$ and $(u, w) \in F_i$, the vertices $(u, w)$ receive each other's auxiliary messages. The latter is summarized as $X_{uw} = X_{wu} = 1$. In order to argue that the graphs are undirected we need to ensure symmetry. Applying the reversal lemma to the second execution of the `UniformGossip` algorithm, we deduce that $Y_{uw} = Y_{wu} = 1$ as well.

Since the set of pruned edges $P_i$ contains directed edges, we need to guarantee that if the direction $(u, v)$ of an edge is pruned then so is the direction $(v, u)$. The two equalities $X_{uw} = X_{wu} = 1$ and $Y_{uw} = Y_{wu} = 1$ imply, by the definition of $P_i$, that $P_i$ is a superset of the edges not cut by the decomposition $V_1 \cup \cdots \cup V_k$. Equivalently, $F_{i+1}$ is a subset of the cut edges. Corollary 3.4, however, bounds the volume of the cut edges by $\frac{1}{2} \operatorname{vol}(F_i)$, which concludes the proof of claim (b). ∎

*Proof of claim* (c). Initially, $\vec{E} \setminus F_0 = \emptyset$ and so the claim holds trivially. By induction, the claim holds for edges in $\vec{E} \setminus F_i$. And so it suffices to establish that $u$ and $v$ exchange their respective payload messages for all $(u, w) \in P_i$. However, this is equivalent to the conditions $X_{uw} + Y_{uw} > 0$, which are enforced by the definition of $P_i$. ∎

**4. Solving NeighborExchange in hereditary sparse graphs.** Next, we ask what can be achieved if instead of exchanging information indirectly as done in the `Superstep` algorithm, we exchange information only directly between neighbors. We will show in this section that this results in very simple deterministic algorithms for an important class of graphs that includes bounded genus graphs and all graphs that can be characterized by excluded minors [32, 33]. The results here will be used for the more general simulators in section 5.

As before we will focus on solving the NeighborExchange problem. One trivial way to solve this problem is for each node to contact its neighbors directly, e.g., by using a simple round robin method. This takes at most $\Delta$ time, where $\Delta$ is the maximum-degree of the network. However, in some cases direct message exchanges work better. One graph that exemplifies this is the star graph on $n$ nodes. While it takes $\Delta = n$ time to complete a round robin in the center, after just a single round of message exchanges each leaf has initiated a bidirectional link to the center and thus exchanged its messages. On the other hand, scheduling edges cannot be fast on dense graphs with many more edges than nodes. We use the following notion to measure the density of a graph.

Definition 4.1. *The* hereditary density *of a graph $G$ is the minimal integer $\delta$ such that for every subset of nodes $S$ the subgraph induced by $S$ has at most density $\delta$, that is, at most $\delta|S|$ edges.*

The following lemma shows that the hereditary density captures how efficient direct message exchanges can be on a given graph.

LEMMA 4.2. *The following holds for a graph $G$ with hereditary density $\delta$:*

1. *Any schedule of direct message exchanges that solves the* NEIGHBOREXCHANGE *problem on $G$ takes at least $\delta$ rounds.*
2. *There exists a schedule of the edges of $G$ such that each node needs only $2\delta$ direct message exchanges to solve the* NEIGHBOREXCHANGE *problem.*

*Proof.* Since the hereditary density of $G$ is $\delta$, there is a subset of nodes $S \subseteq V$ with at least $\delta|S|$ edges between nodes in $S$. In each round, each of the $|S|$ nodes is allowed to schedule at most one message exchange, so a simple pigeonhole principle argument shows that at least one node needs to initiate at least $\delta$ message exchanges.

For the second claim, we are going to show that for any $\epsilon > 0$ there is an $O(\epsilon^{-1} \log n)$-time deterministic distributed algorithm in the $\mathcal{LOCAL}$ model that assigns the edges of $G$ to nodes such that each node is assigned at most $2(1+\epsilon)\delta$ edges. Then setting $\epsilon < (3n^2)^{-1}$ makes the algorithm inefficient but finishes the existential proof since every node is assigned at most $\lfloor 2\delta + 1/2n \rfloor = \lfloor 2\delta \rfloor$ edges.

The algorithm runs in phases in which, iteratively, a node takes responsibility for some of the remaining edges connected to it. All edges that are assigned are then eliminated and so are nodes that have no unassigned incident edges. In each phase, every node of degree at most $2(1+\epsilon)\delta$ takes responsibility for all of its incident edges (breaking ties arbitrarily). At least a $1/(1+\frac{1}{\epsilon})$ fraction of the remaining nodes falls under this category in every phase. This is because otherwise, the number of edges in the subgraph would be more than $\big(|S| - |S|/(1+\frac{1}{\epsilon})\big)\big(2(1+\epsilon)\delta\big)/2 = |S|\delta$, which would contradict the fact that the hereditary density of the graph equals $\delta$. What remains after each phase is an induced subgraph which, by definition of the hereditary density, continues to have hereditary density at most $\delta$. The number of remaining nodes thus decreases by a factor of $1 - 1/(1+\frac{1}{\epsilon})$ in every phase and it takes at most $O(\log_{1+\epsilon} n)$ phases until no more nodes remain, at which point all edges have been assigned to a node. ∎

We note that the lower bound of Lemma 4.2 is tight in all graphs, that is, the upper bound of $2\delta$ can be improved to $\delta$. Graphs with hereditary density $\delta$, also known as $(0, \delta)$-sparse graphs, are thus exactly the graphs in which $\delta$ is the minimum number such that the edges can be oriented to form a directed graph with outdegree at most $\delta$. This in turn is equivalent to the *pseudoarboricity* of the graph, that is, the minimum number of pseudoforests needed to cover the graph, which can be computed in polynomial time [21]. For our purposes the (nondistributed) algorithms to compute these optimal direct message exchange schedule are too slow. Instead, we present a simple and fast algorithm, based on the $\mathcal{LOCAL}$ algorithm in Lemma 4.2, which computes a schedule that is within a factor of $2(1+\epsilon)^2$ of the optimal. We note that the `DirectExchange` algorithm presented here works in the $\mathcal{GOSSIP}$ model and furthermore does not require the hereditary density $\delta$ to be known a priori. The algorithm for an individual node $v$ is given in Figure 2. Its properties are stated in Theorem 1.2.

THEOREM 4.3 (repeated). *For any $\epsilon > 0$, the deterministic algorithm* `DirectExchange` *solves the* NEIGHBOREXCHANGE *problem in the $\mathcal{GOSSIP}$ model using $O(\frac{\delta \log n}{\epsilon^2})$ rounds, where $\delta$ is the hereditary density of the underlying topology. During the algorithm, each node initiates at most $2(1+\epsilon)^2\delta$ exchanges.*

```
Set δ' = 1 and H = ∅. H is the subset of neighbors in Γ(v) that node v
has exchanged messages with. Repeat:
    δ' = (1 + ε)δ'
    for O(1/ε · log n) times do
        if |Γ(v) \ H| ≤ δ'
            in the next δ' rounds exchange messages with all neighbors
            in Γ(v) \ H
            terminate
        else
            wait for δ' rounds
        update H
```

FIG. 2. *Code for* `DirectExchange` *algorithm.*

*Proof.* Let $\delta$ be the hereditary density of the underlying topology. Exactly as in the proof of Lemma 4.2, it holds that the algorithm terminates during the for-loop if $\delta'$ is at least $2(1+\epsilon)\delta$. This is because for this value of $\delta'$, at least a $1/(1+\frac{1}{\epsilon})$ fraction of the remaining nodes terminate at each iteration of the inner loop, as otherwise the number of edges in the current set of nodes $S$ exceeds $\delta|S|$, contradicting $\delta$ being the hereditary density. Thus, when the algorithm terminates, $\delta'$ is at most $2(1+\epsilon)^2\delta$, which is also an upper bound on the number of neighbors contacted by any node. In the $(i+1)$th-to-last iteration of the outer loop, $\delta'$ is at most $2(1+\epsilon)^2\delta/(1+\epsilon)^i$, and the running time for this phase is thus at most $2(1+\epsilon)^2\delta/(1+\epsilon)^i \cdot O(\frac{1}{\epsilon}\log n)$. Summing up over these powers of $1/(1+\epsilon)$ results in a total of at most $\delta/((1+\epsilon)-1)\cdot O(\frac{1}{\epsilon}\log n) = O(\frac{\delta \log n}{\epsilon^2})$ rounds. ∎

**5. Simulators and graph spanners.** In this section we generalize our results to arbitrary simulations of $\mathcal{LOCAL}$ algorithms in the $\mathcal{GOSSIP}$ model and point out connections to graph spanners, another well-studied subject.

Recall that we defined the NEIGHBOREXCHANGE problem exactly in such a way that it simulates in the $\mathcal{GOSSIP}$ model what is done in one round of the $\mathcal{LOCAL}$ model. With our solutions, an $O(\delta \log n)$-round algorithm and an $O(\log^3 n)$-round algorithm for the NEIGHBOREXCHANGE problem in the $\mathcal{GOSSIP}$ model, it is obvious that we can now easily convert any $T$-round algorithm for the $\mathcal{LOCAL}$ model to an algorithm in the $\mathcal{GOSSIP}$ model, e.g., by $T$ times applying the `Superstep` algorithm. In the case of the `DirectExchange` algorithm we can do even better. While it takes $O(\delta \log n)$ rounds to compute a good scheduling, once it is known it can be reused and each node can simply exchange messages with the same $O(\delta)$ nodes without incurring an additional overhead. Thus, simulating the second and any further rounds can be easily done in $O(\delta)$ rounds in the $\mathcal{GOSSIP}$ model. This means that any algorithm that takes $O(T)$ rounds to complete in the $\mathcal{LOCAL}$ model can be converted to an algorithm that takes $O(\delta T + \delta \log n)$ rounds in the $\mathcal{GOSSIP}$ model. We call this a simulation and define simulators formally as follows.

DEFINITION 5.1. *An $(\alpha, \beta)$-simulator is a way to transform any algorithm $A$ in the $\mathcal{LOCAL}$ model to an algorithm $A'$ in the $\mathcal{GOSSIP}$ model such that $A'$ computes the same output as $A$ and if $A$ takes $O(T)$ rounds then with high probability $A'$ takes at most $O(\alpha T + \beta)$ rounds.*

Phrasing our results from sections 3.2 and 4 in terms of simulators we get the following corollary.

COROLLARY 5.2. *For a graph $G$ of $n$ nodes, hereditary density $\delta$, and maximum degree $\Delta$, the following hold:* (a) *there is a randomized* $(\log^3 n, 0)$*-simulator;* (b) *there is a deterministic* $(\Delta, 0)$*-simulator;* (c) *there is a deterministic* $(2(1+\epsilon)^2\delta, O(\delta\epsilon^{-2}\log n))$*-simulator for any* $\epsilon > 0$ *or, simply, there is a* $(\delta, \delta\log n)$*-simulator.*

Note that for computations that require many rounds in the $\mathcal{LOCAL}$ model, the $(2(1+\epsilon)^2\delta, O(\delta\epsilon^{-2}\log n))$-simulator is a $\log n$-factor faster than repeatedly applying the `DirectExchange` algorithm. This raises the question of whether we can similarly improve our $(\log^3 n, 0)$-simulator to obtain a smaller multiplicative overhead for the simulation.

What we would need for this is to compute, e.g., using the `Superstep` algorithm, a schedule that can then be repeated to exchange messages between every node and its neighbors. What we are essentially asking for is a short sequence of neighbors for each node over which each node can indirectly get in contact with all its neighbors. Note that any such schedule of length $t$ must at least fulfill the property that the union of all edges used by any node is connected (if the original graph $G$ is connected) and even more that each node is connected to all its neighbors via a path of length at most $t$. Subgraphs with this property are called *spanners*. Spanners are well-studied objects, due to their extremely useful property that they approximately preserve distances while potentially being much sparser than the original graph. The quality of a spanner is described by two parameters, its number of edges and its *stretch*, which measures how well it preserves distances.

DEFINITION 5.3 (spanners). *A subgraph $S = (V, E')$ of a graph $G = (V, E)$ is called an* $(\alpha, \beta)$-stretch spanner *if any two nodes $u, v$ with distance $d$ in $G$ have distance at most $\alpha d + \beta$ in $S$.*

From the discussion above it is also clear that any solution to the NEIGHBOREXCHANGE problem in the $\mathcal{GOSSIP}$ model also computes a spanner as a byproduct.

LEMMA 5.4. *If $A$ is an algorithm in the $\mathcal{GOSSIP}$ model that solves the* NEIGHBOREXCHANGE *problem in any graph $G$ in $T$ rounds, then this algorithm can be used to compute a $(T, 0)$-stretch spanner with hereditary density $T$ in $O(T)$ rounds in the $\mathcal{GOSSIP}$ model.*

While there are spanners with better properties than the $(\log^3 n, 0)$-stretch and $\log^3 n$-density implied by Lemmas 5.4 and 3.5, our construction has the interesting property that the number of messages exchanged during the algorithm is at most $O(n\log^3 n)$, whereas all prior algorithms rely on the broadcast nature of the $\mathcal{LOCAL}$ model and therefore use already $O(n^2)$ messages in one round on a dense graph. Lemma 5.4 furthermore implies a nearly logarithmic lower bound on the time that is needed in the $\mathcal{GOSSIP}$ model to solve the NEIGHBOREXCHANGE problem since a significantly sub logarithmic simulator would imply the existence of a too good spanner.

COROLLARY 5.5. *For any algorithm in the $\mathcal{GOSSIP}$ model that solves the* NEIGHBOREXCHANGE *problem there is a graph $G$ on $n$ nodes on which this algorithm takes at least $\Omega(\frac{\log n}{\log\log n})$ rounds.*

*Proof.* Assume an algorithm takes at most $T(n)$ rounds on any graph with $n$ nodes. The edges used by the algorithm form a $T(n)$-stretch spanner with density $T(n)$, as stated in Lemma 5.4. For values of $T(n)$ which are too small it is known that such spanners do not exist [36]. More specifically it is known that there are graphs with $n$ nodes, density at least $1/4n^{1/r}$, and girth $r$, that is, the length of

the smallest cycle is $r$. In such a graph any $(r-2)$-stretch spanner has to be the original graph itself, since removing a single edge causes its end-points to have distance at least $r-1$, and thus the spanner also has density $1/4n^{1/r}$. Therefore $T(n) \geq \max_r\{\min(r-2, 1/4n^{1/r})\} = \Omega(\frac{\log n}{\log\log n})$. ∎

Interestingly, it is not only the case that efficient simulators imply good spanners but the next theorem shows as a converse that good existing spanner constructions for the $\mathcal{LOCAL}$ model can be used to improve the performance of simulators.

THEOREM 5.6. *If there is an algorithm that computes an $(\alpha, \beta)$-stretch spanner with hereditary density $\delta$ in $O(T)$ rounds in the $\mathcal{LOCAL}$ model, then this can be used along with an $(\alpha', \beta')$-simulator to construct an $(\alpha\delta, T\alpha' + \beta' + \delta\log n + \delta\beta)$-simulator.*

*Proof.* For simplicity we first assume that $\beta = 0$, that is, the spanner $S$ computed by the algorithm in the $\mathcal{LOCAL}$ model has purely multiplicative stretch $\alpha$ and hereditary density $\delta$. Our strategy is simple: We are first going to compute the good spanner by simulating the spanner creation algorithm from the $\mathcal{LOCAL}$ model using the given simulator. This takes $O(T\alpha' + \beta')$ rounds in the $\mathcal{GOSSIP}$ model. Once this spanner $S$ is computed we are only going to communicate via the edges in this spanner. Note that for any node there is a path of length at most $\alpha$ to any of its neighbors. Thus if we perform $\alpha$ rounds of $\mathcal{LOCAL}$-flooding rounds in which each node forwards all messages it knows of to all its neighbors in $S$ each node obtains the messages of all its neighbors in $G$. This corresponds exactly to a NEIGHBOREXCHANGE in $G$. Therefore if we want to simulate $T'$ rounds of an algorithm $A$ in the $\mathcal{LOCAL}$ model on $G$ we can alternatively perform $\alpha T'$ $\mathcal{LOCAL}$ computation rounds on $S$ while doing the $\mathcal{LOCAL}$ computations of $A$ every $\alpha$ rounds. This is a computation in the $\mathcal{LOCAL}$ model but on a sparse graph. We are therefore going to use the $(\delta, \delta\log n)$-simulator from Corollary 5.2 to simulate this computation which takes $O(\delta\alpha T' + \delta\log n)$ rounds in the $\mathcal{GOSSIP}$ model. Putting this together with the $O(T\alpha' + \beta')$ rounds it takes to compute the spanner $S$ we end up with $O(\delta\alpha T' + \delta\log n + T\alpha' + \beta')$ rounds in total.

In general (for example, for $\beta > \alpha$) it is not possible (see, e.g., Corollary 5.5) to simulate the $\mathcal{LOCAL}$ algorithm step by step. Instead we rely on the fact that any $\mathcal{LOCAL}$ computation over $T$ rounds can be performed by each node first gathering information of all nodes in a $T$-neighborhood and then doing $\mathcal{LOCAL}$ computations to determine the output. For this all nodes simply include all their initial knowledge (and for a randomized algorithm all the random bits they might use throughout the algorithm) in a message and flood this in $T$ rounds to all nodes in their $T$-neighborhood. Because a node now knows all information that can influence its output over a $T$-round computation it can now locally simulate the algorithm for itself and its neighbors to the extent that its output can be determined. Having this we simulate the transformed algorithm as before: We first precompute $S$ in $O(T\alpha' + \beta')$ time and then simulate the $T'$ rounds of flooding in $G$ by performing $\alpha T' + \beta$ rounds of $\mathcal{LOCAL}$-flooding in $S$. Using the $(\delta, \delta\log n)$-simulator this takes $O(\delta(\alpha T' + \beta) + \delta\log n)$ rounds in the $\mathcal{GOSSIP}$ model. ∎

COROLLARY 5.7. *There is a $(2^{\log^* n}\log n, \log^4 n)$-simulator, a $(\log n, 2^{\log^* n}\log^4 n)$-simulator, and a $(O(1), \mathit{polylog}(n))$-simulator.*

*Proof.* We are going to construct the simulators with increasingly better multiplicative overhead by applying Theorem 5.6 to existing spanner constructions [9, 16, 38, 39] for the $\mathcal{LOCAL}$ model. We first construct a $(\log^2 n, \log^4 n)$-simulator by combining our new $(\log^3 n, 0)$-simulator with the deterministic spanner construction in [9]. The construction in [9] takes $O(\log n)$ rounds in the $\mathcal{LOCAL}$ model and adds at

most one edge to each node per round. Using $\alpha = T = \delta = O(\log n)$, $\alpha' = \log^3 n$, and $\beta = \beta' = 0$ in Theorem 5.6 gives the desired $(\log^2 n, \log^4 n)$-simulator. Having this simulator, we can use [39] to improve the multiplicative overhead while keeping the additive simulation overhead the same. In [39] an $\alpha = (2^{\log^* n} \log n)$-stretch spanner with constant hereditary density $\delta = O(1)$ is constructed in $T = O(2^{\log^* n} \log n)$-time in the $\mathcal{LOCAL}$ model. Using these parameters and the $(\log^2 n, \log^4 n)$-simulator in Theorem 5.6 leads to the strictly better $(2^{\log^* n} \log n, \log^4 n)$-simulator claimed here. Having this simulator, we can use it with the randomized spanner construction in [16]. There, an $\alpha$-stretch spanner, with $\alpha = O(\log n)$, is constructed in $T = O(\log^3 n)$-time in the $\mathcal{LOCAL}$ model by extracting a subgraph with $\Omega(\log n)$ girth. Such a graph has constant hereditary density $\delta = O(1)$, as argued in [36]. Using these parameters and the $(2^{\log^* n} \log n, \log^4 n)$-simulator in Theorem 5.6 leads to the $(\log n, 2^{\log^* n} \log^4 n)$-simulator. Finally, we can use any of these simulators together with the nearly additive $(5 + \epsilon, \text{polylog}(n))$-spanner construction from [38] to obtain our last simulator. It is easy to verify that the randomized construction named $AD^{\log \log n}$ in [38] can be computed in a distributed fashion in the $\mathcal{LOCAL}$ model in $\text{polylog}(n)$ time and has hereditary density $\delta = O(1)$. This together with any of the previous simulators and Theorem 5.6 results in a $(O(1), \text{polylog}(n))$-simulator. ∎

With these various simulators it is possible to simulate a computation in the $\mathcal{LOCAL}$ model with very little (polylogarithmic) multiplicative or additive overhead in the $\mathcal{GOSSIP}$ model. Note that while the complexity of the presented simulators is incomparable, one can interleave their executions (or the executions of the simulated algorithms) and thus get the best runtime for any instance. This, together with Corollaries 5.7 and 5.2, proves our main result of Theorem 1.3, as follows.

THEOREM 5.8 (repeated). *Every algorithm in the $\mathcal{LOCAL}$ model which completes with high probability in $T = T(G)$ rounds when run on the topology $G$ can be simulated in the $\mathcal{GOSSIP}$ model in*

$$O(1) \cdot \min \left\{ \begin{array}{lll} T \cdot \log^3 n, & & \\ T \cdot 2^{\log^* n} \log n & + \log^4 n, & \\ T \cdot \log n & + 2^{\log^* n} \log^4 n, & \\ T & + \log^{O(1)} n, & \\ T \cdot \delta & + \delta \log n, & \\ T \cdot \Delta & & \end{array} \right\}$$

*rounds with high probability, where $n$ is the number of nodes, $\Delta$ the maximum degree, and $\delta$ the hereditary density of $G$.*

*Proof of Theorem* 1.3. We can simulate in the $\mathcal{GOSSIP}$ model any algorithm which completes in $T = T(G)$ rounds in the $\mathcal{LOCAL}$ model with the following complexities. The direct approach of simulating each $\mathcal{LOCAL}$ round by $\Delta$ rounds in the $\mathcal{GOSSIP}$ model, where $\Delta$ is the maximum degree in $G$, gives an $O(T \cdot \Delta)$-round algorithm. By Theorem 1.1, we obtain a simulation with a multiplicative factor of $O(\log^3 n)$ by running the NEIGHBOREXCHANGE algorithm $T$ times. Corollary 5.2 gives a simulation which requires $O(T\delta + \delta \cdot \log n)$ rounds when $\delta$ is the hereditary density of $G$. Finally, Corollary 5.7 gives the remaining three simulators, with running times of $O(T \cdot 2^{\log^* n} \log n + \log^4 n)$, $O(T \cdot \log n + 2^{\log^* n} \log^4 n)$ and $O(T, +\text{polylog}(n))$, respectively.

Interleaving all of the above simulations gives the claimed complexity. ∎

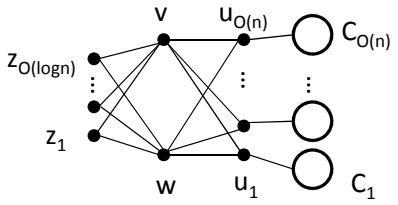


FIG. 3. *An example illustrating the problematic behavior of the natural* NEIGHBOREXCHANGE *algorithm, which repeatedly has each node choose a random neighbor whose information is still unknown. $C_i$ stands for a clique of size $O(1)$ in which every node is also connected to the node $u_i$.*

**6. Discussion: A counterexample to a simpler NEIGHBOREXCHANGE algorithm, synchronization, and fault-tolerance issues.** We first discuss a counterexample to the simpler NEIGHBOREXCHANGE algorithm proposed in section 1.2 and then briefly discuss the related issues of synchronization and fault-tolerance.

As mentioned in section 1.2 a simpler and maybe more natural candidate for a NEIGHBOREXCHANGE algorithm would be for each node to choose a neighbor uniformly at random only from among those it has not yet heard from (directly or indirectly). The counterexample given in Figure 3 demonstrates the subtle complications which can make this algorithm fail miserably.

In this example, it takes two rounds for the node $w$ to hear about the node $v$ (through nodes in $\{z_1, \ldots, z_{O(\log n)}\}$). During these rounds there is a high probability that a constant fraction of the nodes in $\{u_1, \ldots, u_{O(n)}\}$ did not yet hear from either $v$ or $w$. With high probability, a constant fraction of these will contact $w$ before contacting $v$, after which they will not contact $v$ anymore because they will have heard from it through $w$. This leaves $O(n)$ nodes which $v$ has to contact directly (since nodes in $\{z_1, \ldots, z_{O(\log n)}\}$ are no longer active since they already heard from both of their neighbors), resulting in a linear number of rounds for NEIGHBOREXCHANGE. While this specific example can still be solved quickly by requiring nodes that have heard from all their neighbors to continue the algorithm with random choices, it is pretty clear that generally no such strategy can succeed.

In fact, what our counterexample demonstrates is how crucial it is to maintain (a certain level of) symmetry and undirectedness in the underlying network to not end up with directed stars in which the centers have to contact each leaf one by one. In the `Superstep` algorithm perfect symmetry is maintained via the reversal step.

While the strong symmetry guarantee provided by the reversal step is tremendously useful its importance for the working of our algorithm is also problematic. This can be seen in two interesting further considerations. The first one is porting our algorithm to a less synchronized setting. Maintaining the guarantees of the very time sensitive reversal step in such a setting requires heavy-handed synchronization approaches that are not desirable in general. The second consideration is fault tolerance. Here we note that just a single transmission being dropped or received out of order can lead to the reversal step breaking symmetry and thus to the introduction of directed edges. In directed graphs the conductance decomposition from section 3.1 fails to hold. Even worse, not only does the analysis fail to go through, but an example similar to the one given in Figure 3 shows that a single dropped message can result in the completion time of the NEIGHBOREXCHANGEgoing from $O(\log^3 n)$ to a worst-case time of $\Omega(n)$.

To alleviate this problem it is would be great to get rid of the reversal step altogether. This has been done successfully in [4]: The basic idea is to do away with the hard decisions to "remove" edges once a message from a neighbor has been received, and instead to multiplicatively decrease the weight of such edges for the next round. This approach would introduce a slight asymmetry in each edge's weight in both directions. In order to analyze such an algorithm, it is needed to understand the behavior of `UniformGossip` in an asymmetric setting. In this setting, each vertex uses its own distribution over outgoing links when choosing a communication partner at each step. Another alternative is the deterministic approach of [28] which requires only weak symmetry requirements that are restricted to very short time-periods and are thus easily enforced asynchronously and in a fault-tolerant way.

**7. Conclusion.** This paper presents a more efficient alternative to the `UniformGossip` algorithm that allows fast rumor spreading on all graphs, with no dependence on their conductance. We then show how this leads to fast simulation in the $\mathcal{GOSSIP}$ model of any algorithm designed for the $\mathcal{LOCAL}$ model by constructing sparse spanners and conversely also to a new interesting spanner construction with low message complexity. While it is known how to obtain robust NeighborExchange, an intriguing remaining challenge is to design fault-tolerant simulators.

**Acknowledgment.** We thank the reviewers for many comments and suggestions that helped us to improve the paper.

REFERENCES

[1] S. Baswana, T. Kavitha, K. Mehlhorn, and S. Pettie, *Additive spanners and $(\alpha, \beta)$-spanners*, ACM Trans. Algorithms, 7 (2010), pp. 5:1–5:26, https://doi.org/10.1145/1868237.1868242.

[2] S. Baswana, S. Khurana, and S. Sarkar, *Fully dynamic randomized algorithms for graph spanners*, ACM Trans. Algorithms, 8 (2012), pp. 35:1–35:51, https://doi.org/10.1145/2344422.2344425.

[3] S. Boyd, A. Ghosh, B. Prabhakar, and D. Shah, *Randomized gossip algorithms*, IEEE/ACM Trans. Netw., 14 (2006), pp. 2508–2530, https://doi.org/10.1109/TIT.2006.874516.

[4] K. Censor-Hillel and G. Giakkoupis, *Fast and Robust Information Spreading*, manuscript, 2013.

[5] K. Censor-Hillel, B. Haeupler, J. Kelner, and P. Maymounkov, *Global computation in a poorly connected world: Fast rumor spreading with no dependence on conductance*, in Proceedings of the 44th Symposium on Theory of Computing, 2012, pp. 961–970.

[6] K. Censor-Hillel and H. Shachnai, *Fast information spreading in graphs with large weak conductance*, in Proceedings of the 22nd ACM-SIAM Symposium on Discrete Algorithms, 2011, pp. 440–448.

[7] F. Chierichetti, S. Lattanzi, and A. Panconesi, *Almost tight bounds for rumour spreading with conductance.*, in Proceedings of the 42nd ACM Symposium on Theory of Computing, 2010, pp. 399–408.

[8] A. Demers, D. Greene, C. Hauser, W. Irish, J. Larson, S. Shenker, H. Sturgis, D. Swinehart, and D. Terry, *Epidemic algorithms for replicated database maintenance*, in Proceedings of the 6th Annual ACM Symposium on Principles of Distributed Computing, 1987, pp. 1–12, https://doi.org/http://doi.acm.org/10.1145/41840.41841.

[9] B. Derbel, C. Gavoille, D. Peleg, and L. Viennot, *On the locality of distributed sparse spanner construction*, in Proceedings of the 27th ACM Symposium on Principles of Distributed Computing, New York, 2008, pp. 273–282, https://doi.org/10.1145/1400751.1400788.

[10] B. Derbel, M. Mosbah, and A. Zemmari, *Fast distributed graph partition and application*, in Proceedings of the 20th International Parallel and Distributed Processing Symposium, 2006, https://doi.org/10.1109/IPDPS.2006.1639362.

[11] B. Doerr and M. Fouz, *Asymptotically optimal randomized rumor spreading*, in Proceedings of the 38th International Conference on Automata, Languages and Programming, Springer-Verlag, Berlin, 2011, pp. 502–513, http://dl.acm.org/citation.cfm?id=2027223.2027274.

[12] B. Doerr, M. Fouz, and T. Friedrich, *Social networks spread rumors in sublogarithmic time*, in Proceedings of the 43rd Annual ACM Symposium on Theory of Computing, 2011, pp. 21–30.

[13] B. Doerr, M. Fouz, and T. Friedrich, *Asynchronous rumor spreading in preferential attachment graphs*, in Proceedings of SWAT, 2012, pp. 307–315.

[14] B. Doerr, T. Friedrich, and T. Sauerwald, *Quasirandom rumor spreading*, in Proceedings of the 19th Annual ACM-SIAM Symposium on Discrete Algorithms, Philadelphia, 2008, pp. 773–781.

[15] B. Doerr, T. Friedrich, and T. Sauerwald, *Quasirandom rumor spreading: Expanders, push vs. pull, and robustness*, in Proceedings of the 36th International Colloquium on Automata, Languages and Programming, 2009, pp. 366–377.

[16] D. Dubhashi, A. Mei, A. Panconesi, J. Radhakrishnan, and A. Srinivasan, *Fast distributed algorithms for (weakly) connected dominating sets and linear-size skeletons*, J. Comput. System Sci., 71 (2005), pp. 467–479, https://doi.org/http://dx.doi.org/10.1016/j.jcss.2005.04.002.

[17] U. Feige, D. Peleg, P. Raghavan, and E. Upfal, *Randomized broadcast in networks*, Random Structures Algorithms, 1 (1990), pp. 447–460, https://doi.org/10.1002/rsa.3240010406.

[18] N. Fountoulakis and A. Huber, *Quasi-random rumor spreading on the complete graph is as fast as randomized rumor spreading*, SIAM J. Discrete Math., 23 (2009), pp. 1964–1991.

[19] N. Fountoulakis, K. Panagiotou, and T. Sauerwald, *Ultra-fast rumor spreading in social networks*, in Proceedings of the 23rd Annual ACM-SIAM Symposium on Discrete Algorithms, 2012, pp. 1642–1660, https://dl.acm.org/citation.cfm?id=2095116.2095246.

[20] P. Fraigniaud and G. Giakkoupis, *On the bit communication complexity of randomized rumor spreading*, in Proceedings of the 22nd ACM Symposium on Parallelism in Algorithms and Architectures, New York, 2010, pp. 134–143, https://doi.org/http://doi.acm.org/10.1145/1810479.1810505.

[21] H. N. Gabow and H. H. Westermann, *Forests, frames, and games: Algorithms for matroid sums and applications*, Algorithmica, 7 (1992), pp. 465–497, https://doi.org/10.1007/BF01758774.

[22] C. Georgiou, S. Gilbert, R. Guerraoui, and D. R. Kowalski, *Asynchronous gossip*, J. ACM, 60 (2013), pp. 11:1–11:42, https://doi.org/10.1145/2450142.2450147.

[23] G. Giakkoupis, *Tight bounds for rumor spreading in graphs of a given conductance*, in Proceedings of the 28th International Symposium on Theoretical Aspects of Computer Science, Dagstuhl, Germany, 2011, pp. 57–68, https://doi.org/http://dx.doi.org/10.4230/LIPIcs.STACS.2011.57.

[24] G. Giakkoupis, *Tight bounds for rumor spreading with vertex expansion*, in Proceedings of SODA, 2014, pp. 801–815.

[25] G. Giakkoupis and T. Sauerwald, *Rumor spreading and vertex expansion*, in Proceedings of the 23rd Annual ACM-SIAM Symposium on Discrete Algorithms, 2012, pp. 1623–1641.

[26] G. Giakkoupis, T. Sauerwald, H. Sun, and P. Woelfel, *Low randomness rumor spreading via hashing*, in Proceedings of STACS, 2012, pp. 314–325.

[27] G. Giakkoupis and P. Woelfel, *On the randomness requirements of rumor spreading*, in Proceedings of the 22nd ACM-SIAM Symposium on Discrete Algorithms, 2011, pp. 449–461.

[28] B. Haeupler, *Simple, fast and deterministic gossip and rumor spreading*, in Proceedings of the 24th Annual ACM-SIAM Symposium on Discrete Algorithms, 2013, pp. 705–716.

[29] R. Karp, C. Schindelhauer, S. Shenker, and B. Vocking, *Randomized rumor spreading*, in Proceedings of the 41st Annual Symposium on Foundations of Computer Science, IEEE, Washington, DC, 2000.

[30] D. Kempe, J. Kleinberg, and A. Demers, *Spatial gossip and resource location protocols*, in Proceedings of the 23rd Annual ACM Symposium on Theory of Computing, New York, 2001, pp. 163–172, https://doi.org/http://doi.acm.org/10.1145/380752.380796.

[31] D. Kempe and J. M. Kleinberg, *Protocols and impossibility results for gossip-based communication mechanisms*, in Proceedings of the 43rd Symposium on Foundations of Computer Science, IEEE, Washington, DC, 2002, pp. 471–480, https://portal.acm.org/citation.cfm?id=645413.652161.

[32] A. Kostochka, *Lower bound of the hadwiger number of graphs by their average degree*, Combinatorica, 4 (1984), pp. 307–316, https://dx.doi.org/10.1007/BF02579141. 10.1007/BF02579141.

[33] W. Mader, *Homomorphieeigenschaften und mittlere kantendichte von graphen*, Math. Ann., 174 (1967), pp. 265–268, https://dx.doi.org/10.1007/BF01364272. 10.1007/BF01364272.

[34] D. Mosk-Aoyama and D. Shah, *Computing separable functions via gossip*, in Proceedings of the 25th Annual ACM Symposium on Principles of Distributed Computing, New York, 2006, pp. 113–122, https://doi.org/http://doi.acm.org/10.1145/1146381.1146401.
[35] D. Peleg, *Distributed Computing: A Locality-Sensitive Approach*, SIAM, Philadelphia, 2000.
[36] D. Peleg and A. A. Schäffer, *Graph spanners*, J. Graph Theory, 13 (1989), pp. 99–116.
[37] D. Peleg and J. D. Ullman, *An optimal synchronizer for the hypercube*, SIAM J. Comput., 18 (1989), pp. 740–747.
[38] S. Pettie, *Low distortion spanners*, ACM Trans. Algorithms, 6 (2009), pp. 7:1–7:22.
[39] S. Pettie, *Distributed algorithms for ultrasparse spanners and linear size skeletons*, Distributed Comput., 22 (2010), pp. 147–166.
[40] T. Sauerwald and A. Stauffer, *Rumor spreading and vertex expansion on regular graphs*, in Proceedings of SODA, 2011, pp. 462–475.
[41] A. Sinclair, *Algorithms for Random Generation and Counting: A Markov Chain Approach*, Birkhauser Verlag, Basel, Switzerland, 1993.
[42] D. A. Spielman and S. Teng, *Spectral sparsification of graphs*, SIAM J. Comput., 40 (2011), pp. 981–1025, https://doi.org/10.1137/08074489X.
[43] S. Verma and W. T. Ooi, *Controlling gossip protocol infection pattern using adaptive fanout*, in Proceedings of the 25th IEEE International Conference on Distributed Computing Systems, 2005, pp. 665–674, https://doi.org/10.1109/ICDCS.2005.20.