Forwarding Without Repeating: 
Efficient Rumor Spreading in Bounded-Degree Graphs

Vincent Gripon, Vitaly Skachek, and Michael Rabbat
Department of Electrical and Computer Engineering
McGill University, Montréal, Canada
Email: vincent-gripon@ens-cachan.org, vitaly.skachek@gmail.com, michael.rabbat@mcgill.ca

Abstract

We study a gossip protocol called forwarding without repeating (fwr). The objective is to spread multiple rumors over a graph as efficiently as possible. fwr accomplishes this by having nodes record which messages they have forwarded to each neighbor, so that each message is forwarded at most once to each neighbor. We prove that fwr spreads a rumor over a strongly connected digraph, with high probability, in time which is within a constant factor of optimal for digraphs with bounded out-degree. Moreover, on digraphs with bounded out-degree and bounded number of rumors, the number of transmissions required by fwr is arbitrarily better than that of existing approaches. Specifically, fwr requires $O(n)$ messages on bounded-degree graphs with $n$ nodes, whereas classical forwarding and an approach based on network coding both require $\omega(n)$ messages. Our results are obtained using combinatorial and probabilistic arguments. Notably, they do not depend on expansion properties of the underlying graph, and consequently the message complexity of fwr is arbitrarily better than classical forwarding even on constant-degree expander graphs, as $n \to \infty$. In resource-constrained applications, where each transmission consumes battery power and bandwidth, our results suggest that using a small amount of memory at each node leads to a significant savings.
1 Introduction

We consider the now widely-studied problem of spreading rumors over a graph. Initially a subset of nodes have a rumor and we are interested in how long it takes to spread these messages to all nodes in the graph. In this paper we are particularly interested in a variant of the classical push-based forwarding algorithm. In the variant, which we refer to as forwarding without repeating, a node already holding one rumor forwards it to a neighbor only one time. Thus, each node must record which rumors it has sent to which neighbors. In bounded degree networks, if the number of rumors is bounded, we show that the number of transmissions required for forwarding without repeating to spread the rumors to all nodes is infinitely smaller than that of standard (push-based) forwarding. Moreover, the delay of forwarding without repeating is within a constant factor of optimal for any strongly connected graph.

Our motivation comes from the variety of resource-constrained applications, where reducing the number of transmissions is important for efficiency of the system. These include efficient broadcast of code updates in networks of embedded wireless sensing and/or control devices [12, 13] and content dissemination in mobile social networks [9]. In these applications, devices have limited onboard energy resources and limited bandwidth. Reducing the number of transmissions can both free up communication resources and conserve energy resources. The forwarding without repeating algorithm considered here requires the use of additional memory at each node. However, in networks with bounded degree and where the goal is to spread a bounded number of rumors, we argue that using additional memory resources at each node may be beneficial if it leads to considerable savings in the number of transmissions.

The literature on rumor spreading is vast. Here we briefly review the developments which are most relevant to our discussion. The seminal work of Frieze and Grimmet [8] introduces the problem of rumor spreading (a.k.a. gossip) and studies push-based gossip protocols on complete graphs. Demers et al. [6] consider the use of gossip algorithms for lazy updating of distributed databases. They characterize the performance of push-, pull-, and push-pull algorithms, again on complete graphs. Chierichetti, Lattanzi, and Panconesi [4] provide the best-known upper bound on the number of iterations required for synchronous gossip (where each node communicates with one neighbor at each iteration) to spread a rumor over the entire network. They show that in a network of \( n \) nodes with conductance \( \phi \), the push-pull strategy reaches every node within \( O\left(\frac{\log^2 n}{\phi} \cdot \log n\right) \) rounds, with high probability. In addition, they show that in graphs which satisfy a regularity condition (all nodes have the same degree, to within a multiplicative factor), the push and pull strategies satisfy similar bounds on their own. Relatively little work has focused on bounding the number of transmissions required to spread rumors (with the exceptions of [10] and [7], which characterize the message complexity for complete graphs and Erdos-Renyi random graphs, respectively). Most work in this direction is of the form: if a network of \( n \) nodes requires \( O(r) \) rounds to spread a message, then the number of transmissions is no more than \( O(n \cdot r) \).

In a slightly different direction, algebraic gossip protocols are proposed and studied in [5]. There the problem of spreading multiple messages is considered, and a network coding approach is adopted. Rumors are each finite-length bitstrings, and rather than gossiping on individual strings, the authors consider schemes where nodes form and transmit random linear combinations of the messages they have already received at each iteration. After every node has received sufficiently many of these coded strings, it can recover the individual rumors by solving a system of linear equations. In graphs with good expansion properties, this leads to a scheme where all nodes receive all messages in less time than it would take to spread all messages individually (without coding).

In this paper we compare three approaches to rumor spreading: classical (push-based) forwarding (for), the network coding approach (nca), and forwarding without repeating (fwr). We prove
the following results:

- First, we analyze \texttt{fwr} in the synchronous time setting and we show that the number of rounds required for it to converge is within a constant factor of the diameter of the network, which is a trivial lower bound for rumor spreading on any graph.

- Next, we analyze all three algorithms in the asynchronous time setting, where only a single pair of neighboring nodes exchange messages in each round. A trivial lower bound on the number of rounds required in this setting is \( n \), the number of nodes in the network. For strongly connected (di)graphs with bounded out-degree, we show that \texttt{fwr} is within a constant factor of this lower bound. Moreover, when considering any family of graphs with bounded degree, we show that both \texttt{for} and \texttt{nca} require a number of rounds which is \( \Omega(n) \) (for \( n \to \infty \)). Since the number of transmissions is proportional to the number of rounds in the asynchronous setting, this implies that \texttt{fwr} is arbitrarily more efficient than both \texttt{for} and \texttt{nca} in large networks in terms of message complexity.

Our proofs use a combination of combinatorial and probabilistic arguments. Notably, our results do not involve assumptions about the conductance or other expansion-related properties of the underlying graph. Hence, these conclusions apply even to the family of constant-degree expander graphs [15]. Another notable feature of this work is that, in contrast to most previous work which assumes the graph is undirected, our results are obtained in the general setting of strongly connected directed graphs.

2 Notation, Assumptions, and Problem Formulation

Consider a connected directed graph \( \mathcal{G} = (\mathcal{V}, \mathcal{E}) \), where \( \mathcal{V} \) is the finite set of vertices, \( |\mathcal{V}| = n \), and \( \mathcal{E} \subseteq \{(u, v) : u, v \in \mathcal{V}, u \neq v\} \) is the set of directed edges. Each vertex represents an agent in a distributed system (e.g., terminals in a communication network). Edges represent communication media (e.g., wires in an Ethernet network, proximity in a wireless network). We say that a vertex \( v' \) is a successor of the vertex \( v \) if \( (v, v') \in \mathcal{E} \). We also say that \( v \) is a predecessor of \( v' \). A neighbor of \( v \) is a vertex that is either its predecessor or its successor.

We impose the gossip transmitter constraint, which implies that at any given time, any vertex in the network can only send a bitstring to at most one of its successors (see also the definition in [14]). Conversely, a vertex may receive multiple bitstrings from different predecessors at the same time. This assumption is reasonable considering a wired network, for instance, where multiple incoming messages can be buffered.

The out-degree \( d^{\text{out}}(v) \) (respectively, in-degree \( d^{\text{in}}(v) \)) of a vertex \( v \) is the number of successors (respectively, predecessors) it has, namely,

\[
\begin{align*}
\quad d^{\text{out}}(v) &= |\{v' : (v, v') \in \mathcal{E}\}| \quad \text{and} \quad d^{\text{in}}(v) = |\{v' : (v', v) \in \mathcal{E}\}|. 
\end{align*}
\]

The out-degree and in-degree of the graph \( \mathcal{G} \) are the maximum out-degree and the maximum in-degree, respectively, taken over all vertices in \( \mathcal{G} \):

\[
\begin{align*}
\quad d^{\text{out}}(\mathcal{G}) &= \max_{v \in \mathcal{V}} d^{\text{out}}(v) \quad \text{and} \quad d^{\text{in}}(\mathcal{G}) = \max_{v \in \mathcal{V}} d^{\text{in}}(v).
\end{align*}
\]

A path in \( \mathcal{G} \) is a sequence of vertices \( v_0, v_1, \ldots, v_\ell \) such that for all \( i = 0, 1, \cdots, \ell - 1 \) holds \( (v_i, v_{i+1}) \in \mathcal{E} \); \( \ell \) is called the length of the path. A shortest path between \( v \) and \( v' \) is a path such that the first vertex is \( v_0 = v \), the last is \( v_\ell = v' \) and the length of the path is minimum over all
paths from \( v \) to \( v' \). We refer to the distance between \( v \) and \( v' \) as the length of the shortest path.
(Note that, in general, this distance is not a proper metric; it is not necessarily symmetric.) The diameter \( D(G) \) is defined as the maximum length of a shortest path between any two vertices in \( G \).
If there exists a path between all \( v \in \mathcal{V} \) and \( v' \in \mathcal{V} \), we say that the graph is strongly connected. If \( n \geq 2 \), a strongly connected graph is such that for all \( v \in \mathcal{V} \) holds \( d^{\text{out}}(v) \geq 1 \) and \( d^{\text{in}}(v) \geq 1 \).
In the sequel, the graph \( G \) is always assumed to be strongly connected.

To perform an asymptotic analysis on the performance of the algorithms in the paper, we consider a family \( \mathcal{F} = \{ G_i \}_{i=1}^{\infty} \) of graphs with increasing number of vertices. We say that the family of graphs \( \mathcal{F} \) has a bounded out-degree if
\[
\exists d \in \mathbb{N} \text{ such that } \forall G_i \in \mathcal{F} : d^{\text{out}}(G_i) \leq d.
\]
Similarly, the family \( \mathcal{F} \) is said to have a bounded in-degree if
\[
\exists \hat{d} \in \mathbb{N} \text{ such that } \forall G_i \in \mathcal{F} : d^{\text{in}}(G_i) \leq \hat{d}.
\]
Examples of families of bounded degree graphs include: a family of directed symmetric chain graphs with \( d = \hat{d} = 2 \); a family of \( q \)-ary rooted directed trees with \( d = q \) and \( \hat{d} = 1 \); a family of directed symmetric toroidal lattices with \( d = \hat{d} = 4 \); and a family of ring graphs with \( d = \hat{d} = 2 \).

Throughout this paper, we consider the following communication scenario: initially, each vertex has only the knowledge of its successors and, possibly, its predecessors (the global structure of the graph is unknown). The representation each vertex has of its neighbors is relative, meaning that there is no global indexing of nodes in the graph.\(^1\) Also, \( m \) distinct bitstrings are initially dispersed throughout the network (each bitstring is initially held by one vertex, different bitstrings may be held by different initial vertices). The bitstrings are assumed to be drawn from a very large set, such as \( \mathbb{K}^t \), where \( \mathbb{K} \) is a finite field (not necessarily binary) and \( t \) is an integer.

The goal of the covering problem is to deliver all \( m \) bitstrings to all vertices in the graph. The transmission can be carried out either in synchronous or in asynchronous mode. In either setting, at most one bitstring is transmitted at a given time from a vertex to one of its successors. Typically, at each point of time, a vertex can transmit to its successors at most one bitstring.

In the synchronous model, the network is provided with an universal clock, such that all the transmissions proceed in synchronized rounds, and all vertices can transmit at each round.

In the asynchronous model, each vertex is equipped with a clock that ticks according to a rate-1 Poisson point process. The clocks at different nodes are assumed to be independent of each other. When the clock ticks at a given vertex, that vertex can transmit (although it may also decide not to do so). Via the additive property of Poisson processes, the collection of clock ticks across the network can be viewed as being generated by a single rate-\( n \) Poisson process, where each clock tick is assigned uniformly and independently to one of the vertices. Hence, the next clock tick is always uniform and i.i.d. over the vertices in the network.

In both synchronous and asynchronous setups, we consider a call model; i.e., where the transmitting vertex calls a communicating partner chosen from all its successors. As it was observed in \cite{11}, the gossip algorithm typically consists of two decision layers. The first layer, a basic gossip algorithm, is related to the decision made by a vertex about which of its successors to contact when its clock ticks. This decision can be made in either deterministic or randomized manner. For example, vertex can select one of its successors, at random using uniform distribution.

\(^1\)This setup is referred to as anonymous networking in the literature, see, e.g., \cite{1}.
The second layer, a gossip-based protocol, is related to what content is transmitted to the successor and how the internal status of the transmitter and receiver is updated. This content can be just one of the bitstrings the transmitting vertex already has in its possession, or it can be a linear combination of some bitstrings, akin the network coding (see [5]).

We consider three problems that are specific cases of what is usually called gossip covering or information spreading problems. Randomized gossip algorithms are usually employed to solve these problems. The three problems are the following.

1. **Expected covering.** In expected covering, the objective is that each vertex in $G$ receives all bitstrings at least once in expectation after an infinite number of clock ticks.

2. **Almost sure covering.** In almost sure covering, the objective is that, after an infinite number of clock ticks, each vertex in $G$ has received all the bitstrings with probability one.

3. **Sure covering.** In sure covering, the objective is to guarantee (i.e., for every sample paths) that each vertex in $G$ receives all the bitstrings after a finite number of transmissions.

There exists a hierarchy between those three problems as stated in the following theorem, which is proved in Appendix A.

**Theorem 1.** Sure covering problem is strictly more difficult than almost sure covering problem that is itself strictly more difficult than expected covering problem.

By slightly abusing the language, we sometimes say the function $f(\cdot)$ is a bound for the covering problem, if $f(\cdot)$ is a bound for all three covering problems.

We define two quantities characterizing the gossip algorithms: the delay and the number of transmissions.

**Definition 1.** In a synchronous gossip algorithm: the delay is the total number of transmission rounds.

In an asynchronous gossip algorithm: the delay is the total number of times any vertex is allowed to transmit (i.e., the number of global clock ticks).

**Definition 2.** The number of transmissions in a (synchronous or asynchronous) gossip algorithm is the total number of actual transmissions used in the algorithm.

The number of transmissions in the asynchronous time setting is always smaller or equal to the delay. It is also smaller or equal to $n$ times the delay of the same algorithm in synchronous time setting. The delay characterizes the time required to solve one of the covering problems. The number of transmissions is a different measure, which is significant if the main objective is to save non-necessary transmissions (for instance, the power consumption, given that each transmission consumes a constant amount of energy). Note that these quantities tend to infinity as $n$ grows. In that case we are interested in comparing the asymptotic behavior of ratios of these quantities for different algorithms.

### 3 Algorithms

We consider three different algorithms: **for, nca** and **fwr**. These algorithms are executed at each vertex in the process of bitstring transmission and receiving. We denote by $S(v)$, $P(v)$ and $C(v)$, respectively, the set of successors, the set of predecessors, and the set of bitstrings known to a vertex $v$ at some time during execution of the algorithm.
3.1 Classical forwarding (\texttt{for})

The \texttt{for} algorithm is straightforward. Each vertex $v$ chooses one of its successors randomly and uniformly from $S(v)$, and also chooses one bitstring randomly and uniformly from $C(v)$, and sends the chosen bitstring to the chosen successor. If it owns no bitstring, then it performs no transmission. Note that in some literature this algorithm is referred to as the push version of \texttt{for}. A pseudo code version of \texttt{for} is presented in Algorithm 1.

**Algorithm 1:** Classical forwarding (\texttt{for}) algorithm for vertex $v$.

Sending a bitstring:
\begin{verbatim}
begin
  if $C(v) \neq \emptyset$ then
    select $v'$ at random in $S(v)$;
    select $c$ at random in $C(v)$;
    send $c$ to $v'$;
  end
end
\end{verbatim}

Receiving a bitstring $c$:
\begin{verbatim}
begin
  $C(v) \leftarrow C(v) \cup \{c\}$;
end
\end{verbatim}

3.2 The network coding approach (\texttt{nca})

In the network coding approach, the bitstrings are assumed to be vectors over the finite field $\mathbb{K}$. In each transmission step, the vertex sends a random linear combination of the bitstrings it already has in its possession. The coefficients of the vectors in this linear combination are selected randomly and uniformly from $\mathbb{K}$. Denote by $c_1, c_2, \cdots, c_m$ the original bitstrings.

In this approach, the receiving vertex has to be able to recover the original bitstrings from the received linear combinations. To do so, it must obtain $m$ linearly independent vectors. Moreover, the coefficients of the encoding must also be known to the receiver. Thus, the transmitted message contains both a header with the coefficients and the information payload. In other words, the message is a pair $(c, w)$ where $w = (w_1, w_2, \cdots, w_m)$ is a vector of $m$ elements in $\mathbb{K}$, and $c = \sum_{i=1}^{m} w_i c_i$.

The receiving vertex stores in its memory the list of pairs received so far,

$$\mathcal{R}(v) = \{ (\hat{c}_i, \hat{w}_i) \}_{i=1}^{r},$$

where $r$ is the current size of the list, and $\hat{w}_i = (\hat{w}_{i,1}, \hat{w}_{i,2}, \cdots, \hat{w}_{i,m})$. For convenience, define the $r \times m$ matrix $W = (\hat{w}_{i,j})_{i=1,2,\cdots,r; j=1,2,\cdots,m}$.

As the size of $\mathcal{R}(v)$ grows, the vertex may try to solve the corresponding system of linear equations in order to retrieve the original bitstrings. Pseudo code for \texttt{nca} is presented in Algorithm 2.

**Lemma 1**: For any graph, the delay and the number of transmissions required by \texttt{nca} are at least that required by \texttt{for} for the same objective (expected or almost sure covering) with a single bitstring.
Sending a bitstring:
begin
\begin{itemize}
\item draw a vector $\alpha = (\alpha_1, \alpha_2, \ldots, \alpha_r)$ uniformly at random in $\mathbb{K}^r$;
\item select a successor $v'$ at random in $\mathcal{S}(v)$;
\item send $(c = \sum_{i=1}^{r} \alpha_i \hat{c}_i, \ w = \alpha W)$ to $v'$;
\end{itemize}
end

Receiving a bitstring $(c, w)$:
begin
\begin{itemize}
\item add $(c, w)$ to $\mathcal{R}(v)$;
\item $r \leftarrow r + 1$;
\item check for a full-rank $m \times m$ submatrix $Z$ of $W$;
\item if $Z$ is invertible then
\begin{itemize}
\item inverse $Z$ and find the initial bitstrings;
\end{itemize}
end
\end{itemize}
end

Algorithm 2: Network coding approach (nca) algorithm for node $v$.

Proof. Let us consider nca on a graph with $m$ bitstrings. In this case, typically the vertex sends out a linear combination of the received messages. It is straightforward to see that the propagation of messages in nca is analogous to the propagation of an initial bitstring in for.

Since it is clear that a vertex that receives no message cannot retrieve the initial bitstrings, the delay and the number of transmissions needed in nca is greater than or equal to that of for.

Note that in both for and nca the selection of the receiving vertex is fully random. Therefore, these algorithms cannot solve the sure covering problem for most of non-trivial graphs.

3.3 Forwarding without repeating (fwr)

When taking a closer look at for, it appears that time and transmissions are wasted when a vertex receives a message it already has. This situation, in particular, happens when the same predecessor sends the same message to the same successor twice. As we show below, a simple amendment to for, which prevents repeated transmissions of the same bitstring, can lead to a meaningful improvement.

In the fwr algorithm, each vertex stores in its memory the list of pairs $\mathcal{L}(v) \subseteq \mathcal{C}(v) \times \mathcal{S}(v)$ of bitstrings and successors it has sent messages to, including which specific bitstrings were sent, and it also records the predecessors from which it has received messages. By using this data, the vertex will not send a bitstring $c$ to a successor $v'$ if the pair $(c, v')$ already appears in $\mathcal{L}(v)$. Thus some redundant transmissions may be avoided. Note that we slightly abuse the notation by treating $\mathcal{L}(v)$ as a set.

Pseudo code version for fwr is given in Algorithm 3. Note that for the sake of simplicity, Algorithm 3 is not presented in a way consistent with the separation in two distinct layers (basic gossip algorithm and gossip-based protocol). However, it is straightforward to change the presentation to make it consistent with that model.

When a vertex receives a bitstring $c$ from predecessor $v'$, it stores the pair $(c, v')$ in the list $\mathcal{L}(v)$. However, in the analysis in this paper, for the sake of simplicity we assume a slightly weaker version of the algorithm fwr. More specifically, we assume that upon receiving a message, the vertex performs no update of the list $\mathcal{L}(v)$.
Sending a bitstring:
\[
\text{begin} \\
\quad \text{if } (\mathcal{C}(v) \times \mathcal{S}(v)) \setminus \mathcal{L}(v) \neq \emptyset \text{ then} \\
\quad \quad \text{select a pair } (c, v') \text{ uniformly at random in } (\mathcal{C}(v) \times \mathcal{S}(v)) \setminus \mathcal{L}(v); \\
\quad \quad \text{send the bitstring } c \text{ to } v'; \\
\quad \quad \text{add } (c, v') \text{ to } \mathcal{L}(v); \\
\text{end}
\]

Receiving a bitstring \(c\) from \(v'\):
\[
\text{begin} \\
\quad \text{add } (c, v') \text{ to } \mathcal{L}(v); \\
\text{end}
\]

Algorithm 3: Forwarding without repeating (fwr) algorithm for node \(v\).

Since fwr can be viewed as an improved version of for, where some redundant transmissions are avoided, it is immediate that the delay and the number of transmissions required using fwr is less or equal to those required when using for.

In the following sections, we derive some bounds for covering problems for both synchronous and asynchronous settings.

4 Synchronous time setting

First, we derive a lower bound on the covering delay for any algorithm and any \(m\) under the synchronous update model. Assume that only one bitstring is used in the network. The number of vertices that have obtained the bitstring cannot grow by more than a factor of two during one step of the algorithm, which corresponds to the case where all vertices having the bitstring send it to distinct vertices that do not have it yet. It follows that the delay for solving the covering problem using any algorithm on \(\mathcal{G}, D(\mathcal{G})\), is at least:

\[
D(\mathcal{G}) \geq \lceil \log_2(n) \rceil.
\]  

(1)

This lower bound is not tight for all graphs. An alternative lower bound depends on the bitstring diameter of a graph. The bitstring diameter \(D_c(\mathcal{G})\) of the graph is the maximum length of a shortest path between a vertex that initially has a bitstring and any other vertex.

Theorem 2. In the synchronous time setting, a lower bound for covering delay is:

\[
D(\mathcal{G}) \geq D_c(\mathcal{G}).
\]

The proof of Theorem 2 appears in Appendix B.

By using fwr, it is immediate that in at most \(m \cdot d^{out}(v)\) steps (corresponding to the case where \(v\) has all \(m\) bitstrings), all the successors of \(v\) will obtain the bitstrings known to \(v\). By using this principle recursively, one can derive the following upper bound on the covering delay:

Lemma 2. When using fwr on a graph \(\mathcal{G}\) in synchronous settings, the covering delay is at most:

\[
D(\mathcal{G}) \leq m \cdot d^{out}(\mathcal{G}) \cdot D_c(\mathcal{G}).
\]  

(2)
It follows from (1) and Theorem 2 that the delay in $f_{wr}$ is at a constant factor $m \cdot d^{out}(G)$ from the optimal number of steps, for any family of graph with bounded out-degree and bounded number of initial bitstrings.

For specific graphs we can provide tighter bound on the sure covering time, as we state in the following theorem, which is proved in Appendix C.

**Theorem 3.** Let $G$ be a directed tree with root $v_0$, and assume that $v_0$ has the bitstring $c$. Consider synchronous gossip algorithm $f_{wr}$ applied to this graph. Then, the number of transmission rounds, $\mathcal{N}(G)$, needed for sure covering of $G$ is upper bounded by

$$\mathcal{N}(G) \leq \max_{\Phi} \left\{ \sum_{v \in \Phi} d^{out}(v) \right\},$$

where $\Phi$ runs over all paths from $v_0$ to the leaves of $G$, and where $d^{out}(v)$ denotes the out-degree of $v$ in the tree $G$.

## 5 Asynchronous time setting

Similar to the scenario of the synchronous time setting, we can derive a trivial lower bound on the covering number of transmissions in asynchronous mode on graph $G$. Moreover, since the number of transmissions is smaller than the delay, this bound also holds for the covering delay. This lower bound is

$$\mathcal{N}(G) \geq n - 1,$$

and is a direct consequence of the fact at most one new vertex receives the bitstring at each asynchronous step (and $n - 1$ vertices initially do not have one of the bitstrings).

As far as the number of transmissions is concerned, we can derive some upper bounds for $f_{wr}$. Those are presented in Theorem 4 (which is proved in Appendix D) and Corollary 1.

**Theorem 4.** The number of transmissions required to solve the covering problem using $f_{wr}$ on $G$ is

$$\mathcal{N}(G) \leq m|E|.$$

**Corollary 1.** As a direct consequence of Theorem 4, we have:

$$\mathcal{N}(G) \leq mn \cdot d^{out}(G).$$

It follows from (3) and Corollary 1 that $f_{wr}$ is asymptotically optimal (up to a multiplicative constant) in the number of required transmissions when applied to a family of graphs with bounded out degree and with a bounded number of initial bitstrings.

Let us now focus on estimating bounds for the number of transmissions required when using $f_{wr}$ for any family of graph with bounded in-degree and bounded number of initial bitstrings. (See Appendix E for an analysis tailored specifically to the family of chain graphs.)
**Theorem 5.** Assume that for is used in asynchronous time setting on the graph \( G \). Then, there exists vertex \( v \in \mathcal{V} \), such that the probability that \( v \) received all bitstrings after at most \( s \) transmissions, \( P_s^*(v) \), satisfies:

\[
P_s^*(v) \leq 1 - \exp\left( -\frac{2s \cdot d^\text{out}(G) \cdot d^\text{in}(v)}{n} \right).
\]

**Proof.** First, note that the expression does not depend on the number of initial bitstrings. It is obvious that a larger number of bitstrings requires more transmissions using for and thus we focus on the simple case where \( m = 1 \). Let \( c \) be the corresponding bitstring.

It will be convenient to say that a vertex of a graph is contaminated if it has obtained the bitstring \( c \). After \( s \) transmissions, the set of contaminated vertices in the graph \( G \) is denoted \( C_s \). Assume that \( G \) contains at least 2 vertices, and take some \( k \in \mathbb{N} \) such that \( n \geq k \geq 1 \). Run for until the point when \(|C_s| = k\) for the first time, and let \( s \) be the corresponding number of transmissions (\( s \) is well defined since the size of \( C_s \) increases by at most one after each transmission). The probability that the next vertex to receive the bitstring at the iteration \( s + 1 \) is a given vertex \( v \) can be upper bounded. As a matter of fact, the algorithm will use any edge at random among those starting at vertices in \( C_s \). We note that, since \(|C_s| = k\) and since the graph is strongly connected, the number of such edges is at least \( k \) (recall that for each \( v \in C_s \), we have \( d^\text{out}(v) \geq 1 \)). At most \( d^\text{in}(v) \) of these edges are connected to a given vertex \( v \notin C_s \). Note that in contrast to the vertices that are drawn uniformly in the asynchronous time setting version of for, the probability to use one edge is not uniform if the degrees of vertices are not equal. However, the ratio between the probability to choose the most probable edge and the less probable edge is at most \( d^\text{out}(G) \). Therefore, the conditional probability that the vertex \( v \) will receive the bitstring at the transmission \( s + 1 \) is

\[
P_{s+1}(v) \leq \frac{|\text{edges connecting a vertex in } C_s \text{ to } v| \cdot d^\text{out}(G)}{|\text{edges starting at vertices in } C_s|} \leq \frac{d^\text{out}(G) \cdot d^\text{in}(v)}{k}.
\]

(This probability is conditioned on the assumption that \(|C_s| = k\), and that \( v \notin C_s \), as we mentioned above.)

Now, run for until the point when \(|C_s| \geq n/2\) for the first time, and let \( s_0 \) be the corresponding number of transmissions. Select a vertex \( v \) that was not contaminated in the course of the first \( s_0 \) transmissions (which is well defined since \( n \geq 2 \)). The conditional probability (conditioned on the assumptions \(|C_{s_0}| \geq n/2 \) and \( v \notin C_s \)) that \( v \) is contaminated after \( s_0 + 1 \) transmissions is therefore:

\[
P_{s_0+1}(v) \leq \frac{2 \cdot d^\text{out}(G) \cdot d^\text{in}(v)}{n}. \tag{4}
\]

Since all transmissions are independent of each other, the probability that \( v \) is contaminated after \( s = s_0 + s' \) transmissions is:

\[
P_s^*(v) \leq 1 - \left( 1 - \frac{2 \cdot d^\text{out}(G) \cdot d^\text{in}(v)}{n} \right)^{s'} \leq 1 - \exp\left( \frac{-2s' \cdot d^\text{out}(G) \cdot d^\text{in}(v)}{n} \right).
\]

Since \( s' \leq s \), this completes the proof. \( \square \)
Expected covering | for | nca | fwr
---|---|---|---
ω(n) | ω(n) | Θ(n)
Almost sure covering | ω(n) | ω(n) | Θ(n)
Sure covering | N/A | N/A | Θ(n)

Table 1: Comparison of the number of transmissions for the different algorithms and different problems when considering asynchronous time setting and a family of graphs with bounded in- and out-degree and bounded number of initial bitstrings.

From Theorem 5 we arrive at the following conclusion.

**Corollary 2.** Suppose that the algorithm for is applied in the asynchronous setting to a family of graphs with bounded in- and out-degree. Then the number of transmissions $s$ for almost sure covering of the graph satisfies:

$$s = \omega(n).$$

**Proof.** Observe that almost sure covering implies that every vertex $v \in G_i$ receives all the $m$ messages with probability one. By picking a vertex $v$ and applying analysis as in the proof of Theorem 5, we obtain that

$$\frac{2s \cdot d^{out}(G) \cdot d^{in}(v)}{n} \to \infty.$$

Denote by $F = \{G_i\}_{i=1}^\infty$ the family of graphs of bounded degree. Then,

$$\exists d \in \mathbb{N} \text{ such that } \forall i, \forall v \in G_i : d^{in}(v) \leq d^{in}(G_i) \leq \hat{d}$$

and

$$\exists d \in \mathbb{N} \text{ such that } \forall i : d^{out}(G_i) \leq d,$$

and thus we obtain

$$d \cdot \hat{d} \cdot \frac{s}{n} \to \infty. \quad (6)$$

If $\lim_{n \to \infty} \frac{s}{bn} \leq 1$, for some constant $b \geq 0$, condition (6) does not hold. Therefore, we conclude that $s$ grows faster than $bn$ for any $b \geq 0$, i.e. $s = \omega(n)$.

Using Corollaries 1, 2 and Lemma 1 we conclude that the ratio of the number of transmissions needed to assure almost sure covering using fwr to that using for or that using nca tends to zero when considering a family of graphs with bounded in- and out-degree and bounded number of bitstrings. Table 1 below summarizes the main results of this extended abstract.

## A Proof of Theorem 1

First, we prove the loose inclusions:

- It is clear that sure covering is more difficult than almost sure covering since if all vertices have all bitstrings, they also have all of them with probability one.
• Suppose that $\text{alg}$ is an algorithm that solves almost sure covering. After using $\text{alg}$ on a graph, let us focus on a bitstring $c \in K^t$ and a vertex $v \in V$. Denote by $T_c(v)$ the random variable that counts the number of times $v$ received the bitstring $c$ during the execution of $\text{alg}$ for infinite number of steps. The probability that $v$ received $c$ at least one time is:

$$P(T_c(v) \geq 1) = \sum_{k=1}^{\infty} P(T_c(v) = k) = 1.$$ 

On the other hand, the expected number of times $v$ received $c$ is:

$$\text{Ex}[T_c(v)] = \sum_{k=1}^{\infty} k \cdot P(T_c(v) = k) = \sum_{k=1}^{\infty} P(T_c(v) = k) + \sum_{k=2}^{\infty} (k-1) \cdot P(T_c(v) = k) \geq \sum_{k=1}^{\infty} P(T_c(v) = k) = 1.$$

Thus $\text{alg}$ always solves expected covering.

Next, let us prove the strict part of the theorem statement:

• Consider a graph that consists of two vertices $v$ and $v'$, and such that $\mathcal{E} = \{(v, v'), (v', v)\}$. Suppose that $v$ initially has some bitstring $c$. We use the following algorithm: when $v$ can transmit (which happens infinitely often under both the synchronous and asynchronous model), it chooses with constant probability $0 < \alpha < 1$ to send its bitstring $c$ to $v'$. This algorithm trivially solves almost sure covering as the probability that $v'$ receives the bitstring $c$ after $s$ steps is:

$$P_{v'}(s) = 1 - (1 - \alpha)^s \to 1.$$

On the other hand, it may happen with probability 0 that at each step, $v$ does not communicate with $v'$ and thus this algorithm does not solve the sure covering problem.

• Using the same graph, let us now consider the following algorithm: the first time $v$ transmits, it draws a random variable with probability 1/2 to be one and probability 1/2 to be zero. If the obtained value is one, then $v$ sends its bitstring to $v'$ exactly two times in a row; if it is zero, then it never sends its bitstring to $v'$. Trivially, the probability that $v'$ receives the bitstring $c$ after $s$ steps is:

$$P_{v'}(s) = 1 - (1 - \alpha)^s \to 1.$$

On the other hand, the expected number of times $v'$ receives the bitstring is $2 \cdot 1/2 = 1$, which means that this algorithm solves the expected covering problem.

### B Proof of Theorem 2

Let $v \in V$ be a vertex that initially has some bitstring $c$ and let $v' \in V$ be another vertex, such that there exists a shortest path starting at $v$ and ending at $v'$ of length $D_c(G)$. We use the fact that if the probability that $v'$ received $c$ is zero, then the expected number of times $v'$ received $c$ is also zero. We now proceed by induction:
Figure 1: Chain graph where vertex $v_i$ is connected to vertices $v_{i-1}$ and $v_{i+1}$.

- At the beginning, the only vertex in the graph that has a nonzero probability to have $c$ is $v_0$.
- After step $s$, only the vertices that are at a distance less than or equal to $s$ from $v$ have a nonzero probability of having received $c$. At step $s + 1$, the gossip transmitter constraint implies that only the successors of vertices that have $c$ may have a nonzero probability to receive $c$.

### C Proof of Theorem 3

First, observe that the graph $G$ is covered if all its leaves have obtained the bitstring $c$. This is due to the fact that every vertex in the tree $G$ is located on the path from the root $v_0$ to some leaf.

Second, let us estimate the number of transmission rounds required to deliver $c$ to a particular leaf $v$. Let $\Phi$ be a path from $v_0$ to $v$, where $\Phi = (v_0, v_1, v_2, \ldots, v_\ell = v)$. In the worst case scenario, $v_0$ will send $c$ to all its other successors before it sends it to $v_1$. Then, in the words case scenario, $v_1$ sends $c$ to all its other successors before it sends it to $v_2$, and so on. In total, bitstring $c$ will be sent to $\sum_{v' \in \Phi} d^{out}(v')$ vertices when it reaches $v$.

The final expression is obtained by taking the leaf $v$ which maximizes the required expression.

### D Proof of Theorem 4

To obtain this result, we first observe that the quantities (for all $v \in V$) $m|S(v)| - |L(v)|$ are non-increasing with the number of transmissions. More precisely, at each transmission, one of the quantities $m|S(v)| - |L(v)|$ decreases by one. Therefore, an upper bound for the number of steps before all $L(v)$’s are empty is:

$$N(G) \leq \sum_{v \in V} m|S(v)| - |L(v)|$$

$$\leq m \sum_{v \in V} |S(v)|$$

$$\leq m|E|$$

### E Analysis of for on the chain graph

**Theorem 6.** Solving expected covering problem using for on a family of graphs with bounded out-degree in the asynchronous time setting requires $\omega(n)$ transmissions.

**Proof.** Take the symmetric directed chain (di)graph depicted in Figure 1 that consists of $n + 1$ vertices. This graph is defined by $E = \{(v_i, v_{i+1})\}_{i=0}^{n-1} \cup \{(v_{i+1}, v_i)\}_{i=0}^{n-1}$. If the vertex $v_0$ owns the bitstring $c$, it takes at least $n$ transmissions to deliver this bitstring to $v_n$.

Assume that the vertex $v_1$ has the only bitstring $c$, and so $m = 1$. Note that the number of vertices having the bitstring $c$ at the end of step $s$, which we denote by $k_s$, grows at most by one at each transmission. Let $s_0$ be the smallest number of transmissions such that $k_{s_0} = \lceil n/2 \rceil$. 

12
Since the set of vertices possessing the bitstring is connected, the set of vertices having the bitstring is
\[ \bigcup_{i=1}^{\lceil n/2 \rceil} v_i. \]
This set might also contain \( v_0 \), but then we ignore \( v_0 \) for the sake of simplicity of the analysis.

Denote by \( E_i(s) \) the event that \( v_i \) receives the bitstring \( c \) at step \( s \). We also denote by \( P[E_i(s)] \) the conditional probability of the event \( E_i(s) \) given that \( k_{s_0} = \lceil n/2 \rceil \) as above and \( s \geq s_0 \).

We obtain that, for \( s \geq s_0 \) and \( i = \lceil n/2 \rceil + 1 \),
\[
P[E_i(s)] \leq \frac{1}{k_{s_0}} \cdot \frac{1}{2},
\]
since \( v_{i-1} \) is the next vertex to transmit with probability \( 1/k_{s_0} \), and \( v_{i-1} \) sends the bitstring to \( v_i \) with probability \( 1/2 \). Since \( k_{s_0} = \lceil n/2 \rceil \), this implies that
\[
P[E_i(s)] \leq \frac{1}{n}.
\]

Since the transmissions are independent, the probability that \( v_i \) receives the bitstring in the first \( s \) transmissions given that it did not receive it in the first \( s_0 \) transmissions, \( s_0 \leq s \), is:
\[
P\left[ \bigcup_{s'=s_0+1}^{s} E_i(s) \right] \leq 1 - \left( 1 - \frac{1}{n} \right)^{s-s_0}.
\]

Take \( s = b \cdot n + s_0 \) for some constant \( b \geq 0 \) independent of \( n \). It is obvious that \( P[\bigcup_{s'=s_0+1}^{s} E_i(s)] \) is an increasing function of \( s \), and therefore, for any \( s \leq b \cdot n + s_0 \), we have:
\[
P\left[ \bigcup_{s'=s_0+1}^{s} E_i(s) \right] \leq 1 - \left( 1 - \frac{1}{n} \right)^{bn}.
\]

Since the events \( E_{i-1}(s), E_i(s) \) and \( E_{i+1}(s) \) form a Markov chain, we can now upper-bound \( P[E_{i+1}(s)] \):
\[
P[E_{i+1}(s)] \leq \left[ 1 - \left( 1 - \frac{1}{n} \right)^{bn} \right] \frac{1}{n}.
\]

Similarly, this yields:
\[
P\left[ \bigcup_{s'=s_0+1}^{s} E_{i+1}(s) \right] \leq \left[ 1 - \left( 1 - \frac{1}{n} \right)^{bn} \right]^2.
\]

This leads inductively to the formula:
\[
P\left[ \bigcup_{s'=s_0+1}^{s} E_{n-1}(s) \right] \leq \left[ 1 - \left( 1 - \frac{1}{n} \right)^{bn} \right]^{n-1-i}.
\]

Now, observe that
\[
\left[ 1 - \left( 1 - \frac{1}{n} \right)^{bn} \right]^{n-1-i} \sim e^{-(n/2)e^{-b}}.
\]
Assume that the algorithm runs for $bn + s_0$ transmissions, where $b > 0$ is a constant. To upper bound the expected number of times the bitstring $c$ is sent to $v_n$ during the execution of the algorithm, we note that the number of times $v_{n-1}$ transmits $c$ to $v_n$ is less than $bn$. Then, the expected number of times that $c$ is sent to $v_n$ is:

$$\leq bn \cdot e^{-(n/2)e^{-b}}.$$  

This value vanishes exponentially fast for sufficiently large values of $n$. We conclude that the expected number of times that $c$ is sent to $v_n$ is very close to zero. Therefore, if the number of transmissions in the algorithm is linear in $n$, the algorithm will not achieve an expected covering of the chain graph.

References

[1] H. Attiya, M. Snir, and M.K. Warmuth, “Computing on an anonymous ring”, *Journal of the ACM*, vol. 35, no. 4, October 1988, pp. 845–875.

[2] C. Avin and G. Ercal, “On the cover time and mixing time of random geometric graphs”, *Theoretical Computer Science*, vol. 380, Issue 1-2, July 2007, pp. 2–22.

[3] K. Censor-Hillel and H. Shachnai, “Fast information spreading in graphs with large weak conductance”, *Proceedings of the Twenty-Second Annual ACM-SIAM Symposium on Discrete Algorithms (SODA’11)*, San Francisco, California, Jan. 2011, pp. 440–448.

[4] F. Chierichetti, S. Lattanzi, and A. Panconesi, “Rumour spreading and graph conductance”, *Proceedings of the Twenty-First Annual ACM-SIAM Symposium on Discrete Algorithms (SODA’10)*, Austin, Texas, Jan. 2010, pp. 1657–1663.

[5] S. Deb, M. Médard, and C. Choute, “Algebraic Gossip: A Network Coding Approach to Optimal Multiple Rumor Mongering”, *IEEE Trans. Inform. Theory*, vol. 52, no. 6, June 2006, pp. 2486–2507.

[6] A.J. Demers, D.H. Greene, C. Hauser, W. Irish, J. Larson, S. Shenker, H.E. Sturgis, D.C. Swinehart, and D.B. Terry, “Epidemic algorithms for replicated database maintenance”, *Proceedings of the Sixth Annual ACM Symposium on Principles of Distributed Computing (PODC’87)*, 1987, pp. 1–12.

[7] R. Elsässer, “On the communication complexity of randomized broadcasting in random-like graphs”, *Proceedings of the 18th Annual ACM Symposium on Parallel Algorithms and Architectures (SPAA’06)*, 2006, pp. 148–157.

[8] A. Frieze and G. Grimmett, “The shortest-path problem for graphs with random arc-lengths”, *Discrete Applied Mathematics*, vol. 10, 1985, pp. 57–77.

[9] S. Ioannidis and A. Chaintreau, “On the strength of weak ties in mobile social networks”, *Proceedings of the ACM Workshop on Social Network Systems (SNS)*, March 2009.

[10] R. Karp, C. Schindelhauer, S. Shenker, and B. Vöcking, “Randomized rumor spreading”, *Proceedings of the IEEE Conference on Foundations of Computer Science (FOCS’00)*, 2000.

[11] D. Kempe, J. M. Kleinberg, and A. J. Demers, “Spatial gossip and resource location protocols,” *in Proc. ACM Symp. Theory of Computing*, Heraklion, Crete, Greece, Jul. 2001.
[12] P. Levis, N. Patel, D. Culler, and S. Shenker, “Trickle: A self-regulating algorithm for code propagation and maintenance in wireless sensor networks”, Proceedings of the USENIX NSDI Conference, San Francisco, March 2004.

[13] M. Mesbahi and M. Egerstedt, Graph Theoretic Methods in Multiagent Networks, Princeton University Press, 2010.

[14] D. Mosk-Aoyama and D. Shah, “Fast distributed algorithms for computing separable functions”, IEEE Trans. Inform. Theory, vol. 54, no. 7, July 2008, pp. 2997–3007.

[15] O. Reingold, S. Vadhan, and A. Widgerson, “Entropy waves, the zig-zag graph product, and new constant-degree expanders”, Annals of Mathematics, vol. 155, no. 2, 2002, pp. 157–187.