Dealing with Unreliable Agents in Dynamic Gossip

Line van den Berg[1](✉) and Malvin Gattinger[2]

[1] Univ. Grenoble Alpes, Inria, CNRS, Saint-Martin-d'Hères, France
line.van-den-berg@inria.fr

[2] University of Groningen, Groningen, The Netherlands
malvin@w4eg.eu

Abstract. Gossip describes the spread of information throughout a network of agents. It investigates how agents, each starting with a unique secret, can efficiently make peer-to-peer calls so that ultimately everyone knows all secrets. In Dynamic Gossip, agents share phone numbers in addition to secrets, which allows the network to grow at run-time.

Most gossip protocols assume that all agents are reliable, but this is not given for many practical applications. We drop this assumption and study Dynamic Gossip with unreliable agents. The aim is then for agents to learn all secrets of the reliable agents and to identify the unreliable agents.

We show that with unreliable agents classic results on Dynamic Gossip no longer hold. Specifically, the Learn New Secrets protocol is no longer characterised by the same class of graphs, so-called sun graphs. In addition, we show that unreliable agents that do not initiate communication are harder to identify than agents that do. This has paradoxical consequences for measures against unreliability, for example to combat the spread of fake news in social networks.

1 Introduction

The internet has led to great changes in the distribution of news. Recently, 'fake news' received attention, possibly having influenced the 2016 US presidential election [1]. Besides the challenge to identify fake news, a question is how to treat it: should false information be removed or is marking it as false sufficient?

Dynamic Gossip is a formal model how information can spread throughout a changing network of agents. It investigates how agents, each with a unique secret, decide, based on their own knowledge about the network, what calls to make so that ultimately everyone knows all the secrets. A gossip protocol can help agents to decide on a call sequence to perform. Examples from the literature are ANY ("call any agent"), CMO ("call me once") and LNS ("learn new secrets") [7]. In a dynamic setting, additional to secrets, agents share phone numbers, allowing the network to grow at run-time.

Traditionally these systems assume that *everybody is reliable*, but this assumption is not justified for many practical applications. Therefore, we adapt

© Springer Nature Switzerland AG 2020
M. A. Martins and I. Sedlár (Eds.): DaLí 2020, LNCS 12569, pp. 51–67, 2020.
https://doi.org/10.1007/978-3-030-65840-3_4

Dynamic Gossip to account for unreliable agents. Of course, the possibilities for agents to be unreliable are numerous: agents can lie about their own secret or about secrets of others, agents can have a memory of whom they have lied to or not, agents can always lie or with a certain probability, and agents can lie merely about secrets or also about phone numbers, etc. With any such form of unreliability, the aim of the reliable agents should still be to learn all the secrets (of the reliable agents) and, in addition, to identify the unreliable agents.

We show that, already with relatively simple unreliable agent a known result in Dynamic Gossip from [7] breaks down. Specifically, the Learn New Secrets protocol is no longer characterised by sun graphs. This emphasises the need to discard the assumption that everyone is reliable for any practical application. In addition, we show that unreliable agents that do not initiate communication are harder to identify than those that do. This has seemingly paradoxical consequences for security measures taken against unreliable agents: blocking as a measure against false information has the adverse effect of securing the anonymity of the unreliable agents. New protocols are needed to properly cope with unreliable agents in Dynamic Gossip.

Our article is structured as follows. We give a short summary of related work in Sect. 2. In Sect. 3 we recall the definitions of Dynamic Gossip. We then define Unreliable Gossip and unreliable agents in Sect. 4 and 5, respectively. The new setting then motivates a new notion of success which we define and examine in Sect. 6. We conclude with future work ideas and a discussion the relevance of Unreliable Gossip for social networks in Sect. 7.

2 Related Work

Gossip has first been studied in combinatorics and graph-theory [11]. The classical question, also known as the "telephone problem" is: Given n agents who each start with a unique secret, how many phone calls are needed to spread all secrets? For $n > 3$ agents that all have the phone number of all other agents, $2n - 4$ calls are necessary and sufficient to make everyone learn all secrets [15]. For networks in which not every agent has the phone number of all other agents, these numbers are naturally higher [10]. Besides communication networks, gossip has also been used for the study of epidemics [9], power grids [17] and neural networks [17].

Most results on classical gossip assume a *central* and *all-knowing scheduler* deciding who should call whom and when. This is not realistic for practical applications in which agents have to decide autonomously what (communication) action to take. Hence, *distributed* gossip has been studied in which agents decide autonomously, on their own, whom to call using *epistemic* protocols [2,3].

More recently, another assumption has been lifted, namely the assumption that the graph representing who can call whom is constant, i.e. agents have a static phone book or contact list. In *dynamic* gossip agents also exchange phone numbers, adding edges in the reachability graph [7]. This means that the network may grow at run-time. This is the setting which we use and extend here.

Most work on the classical telephone problem and on Dynamic Gossip assumes that all agents are reliable and follow the same protocol. However, gossip with unreliability has been studied extensively in other areas. One direction of research is about settings where communication links are unreliable, as studied in [14] and [16]. In contrast, here we assume that communication works perfectly but that agents are unreliable. Our setting is thus more comparable to having faulty or malicious agents, as in distributed storage [6] or consensus protocols [4].

Similar to our work is also the proposal of 'corrected gossip' in [12]. The authors study failing nodes and define a gossip protocol which tries to reduce latency of the total group communication. A big difference to our setting is that their networks are static and no links are added at run-time.

3 Dynamic Gossip

We now give a short introduction to Dynamic Gossip, following [7]. We assume a finite set of agents, $\mathcal{A}$. Initially, each agent knows only their own secret and some set of phone numbers including their own. If an agent a has the phone number of an agent b, then the phone call ab can take place. During a call, the two agents exchange secrets and phone numbers — including those they learned in previous calls. One might wonder what else agents learn in such a call, but for all results we discuss here higher-order knowledge such as "a knows that b knows the secret of c" is irrelevant, hence we will not model it and refer to [8].

Example 1. Suppose agent a knows the number of b, and agents b and c know each other's number and no other numbers are known. We draw this situation below. Note that we use dashed arrows for the binary relation of knowing the number of someone (N). Now if a calls b then a and b learn each other's secret, which we draw with solid arrows (S). We also add another dashed arrow: in the call ab agent a also learns the number of c.

$$\begin{tikzcd}
a \arrow[r, dashed] & b \arrow[r, dashed, leftrightarrow] & c \arrow[r, phantom, "\xrightarrow{ab}"] & a \arrow[r, leftrightarrow] \arrow[rr, dashed, bend left] & b \arrow[r, dashed, leftrightarrow] & c
\end{tikzcd}$$

We formally define gossip graphs, calls and sequences as follows.

Definition 1 (Gossip Graph). *For any set $\mathcal{A}$, let $I_{\mathcal{A}} := \{(a,a) \mid a \in \mathcal{A}\}$. A* gossip graph *is a triple $G = (\mathcal{A}, N, S)$ where $\mathcal{A}$ is a finite set of agents, $N \subseteq \mathcal{A} \times \mathcal{A}$ and $S \subseteq \mathcal{A} \times \mathcal{A}$ such that $I_{\mathcal{A}} \subseteq N$ and $I_{\mathcal{A}} \subseteq S$. Given any G, let $S_a := \{b \in \mathcal{A} \mid (a,b) \in S\}$ and $N_a := \{b \in \mathcal{A} \mid (a,b) \in N\}$.*

A graph is initial *iff $S = I_{\mathcal{A}}$. A graph is* complete *iff $S = \mathcal{A} \times \mathcal{A}$.*

An agent a is an expert *iff $S_a = \mathcal{A}$. An agent a is* terminal *iff $N_a = \{a\}$.*

We say that "agent a knows the number of agent b" iff $(a,b) \in N$. Similarly, we say "agent a knows the secret of agent b" iff $(a,b) \in S$.

We now define calls ab, in which agent a and b share all their information.

Definition 2 (Call). *Suppose $G = (\mathcal{A}, N, S)$, $a, b \in \mathcal{A}$ and $(a, b) \in N$. The* call *ab maps G to $G^{ab} := (\mathcal{A}, N^{ab}, S^{ab})$ where*

$$N_c^{ab} := \begin{cases} N_a \cup N_b & \textit{if } c \in \{a, b\} \\ N_c & \textit{otherwise} \end{cases} \quad \textit{and} \quad S_c^{ab} := \begin{cases} S_a \cup S_b & \textit{if } c \in \{a, b\} \\ S_c & \textit{otherwise} \end{cases}$$

Definition 3 (Call sequences). *A* call sequence *σ is a sequence of calls. We use the following notation: ϵ is the empty sequence and $\sigma; \tau$ is the concatenation of two sequences σ and τ; Moreover, $\sigma \sqsubseteq \tau$ denotes that σ is a prefix of τ.*

We say that call ab is possible *on a graph $G = (\mathcal{A}, N, S)$ iff $(a, b) \in N$. The call sequence ϵ is possible on any graph, and a call sequence $ab; \sigma$ is possible on G iff the call ab is possible on G and σ is possible on G^{ab}. If a call sequence σ is possible on a graph G, then G^σ is defined by: $G^\epsilon := G$ and $G^{ab;\sigma} := (G^{ab})^\sigma$.*

It is an easy exercise to show by induction on σ that "if a knows the secret of b, then a also knows the number of b" is an invariant when making calls.

Lemma 1. *For any initial graph $G = (\mathcal{A}, N, S)$ and any call sequence σ that is possible on G, we have in the resulting graph $G^\sigma = (\mathcal{A}, N^\sigma, S^\sigma)$ that $S^\sigma \subseteq N^\sigma$.*

A protocol for Dynamic Gossip is a rule how agents decide whom they should call. The goal of a gossip protocol is to reach a complete graph, where everybody knows all secrets. Moreover, good protocols will use fewer calls and avoid superfluous or redundant calls. Here we will focus on the LNS protocol from [7]. For a general definition of protocols in a formal language, see [8].

Definition 4 (LNS Protocol). *A call ab is* LNS-permitted *iff $(a, b) \in N$ and $(a, b) \notin S$.*

We now define when a protocol is successful on a graph. Intuitively, this means all possible executions of the protocol lead to a complete graph.

Definition 5 (Success). *Let P_G bet the set of all call sequences possible on G and permitted by protocol P. We also call such call sequences P-*permitted.

Let a graph $G = (\mathcal{A}, N, S)$ and a protocol P be given. A finite call sequence $\sigma \in P_G$ is successful *iff G^σ is complete. A sequence σ is P-*maximal *on G iff σ is P-permitted on G and there is no call P-permitted on G^σ, i.e. no call ab can be added to σ such that $\sigma; ab$ is still P-permitted.*

- *P is* strongly successful *on G if all P-maximal $\sigma \in P_G$ are successful.*
- *P is* weakly successful *on G if there is a $\sigma \in P_G$ that is successful.*
- *P is* unsuccessful *on G if there is no $\sigma \in P_G$ that is successful.*

Given a certain class $\mathcal{G}$ of networks (graphs) and a protocol P, we can ask the question: is P (strongly, weakly, un-) successful on $\mathcal{G}$. That is, does P lead to a complete network? This question, the *gossip problem*, is used to characterise networks both by their graph-theoretical properties and by the protocols that are (strongly, weakly, un-) successful on them.

It is easy to see that on any graph that consists of disconnected parts no protocol is successful. Hence, graphs need to be weakly connected to allow any of the protocols to be successful [7].

Definition 6. *A graph $G = (\mathcal{A}, N, S)$ is* weakly connected *iff for all agents $a, b \in \mathcal{A}$ there is a undirected N-path between a and b. We say that G is* strongly connected *iff for all agents $a, b \in \mathcal{A}$ there is an N-path from a to b.*

A graph $G = (\mathcal{A}, N, S)$ is a sun graph *iff N is strongly connected on $s(G)$, where $s(G)$ is the result of removing all terminal agents from G.*

Informally, one can think of sun graphs as 'almost' strongly connected graphs.

Example 2. The following graph is a sun graph: if we remove the only terminal agent a, then we obtain a strongly connected graph (consisting of b and c).

$$\begin{tikzcd} a & b \arrow[l, dashed] \arrow[r, leftrightarrow, dashed] & c \end{tikzcd}$$

Theorem 1 (Theorem 13 in [7]). *Suppose G is an initial gossip graph. Then LNS is strongly successful on G iff G is a sun graph.*

4 Unreliable Gossip

It is easy to define reliable agents: they do exactly what they are expected to do. In particular, reliable agents communicate truthfully about their own secret, about secrets of others and share all the phone numbers they have.

However, when unreliability is allowed, there are numerous different options. There may be noise on the communication channel causing the communication between agents to fail; agents may (intentionally or unintentionally) follow a different protocol; agents may actively spread lies, either about their own secret, about other agents' secrets or both; agents may sabotage connections between other agents; unreliable agents may form coalitions to manipulate the network; the degree of unreliability may evolve over time, via peer pressure or other mechanisms; unreliable agents might have a memory of whom they have lied to; etc. This gives rise to many different types of unreliable behaviour.

In this article we only consider a basic form of unreliability: unreliability in the form of unintended random memoryless noise. A real-world example for this kind of unreliability could be a network of sensors that communicate with each other, but where one or more of the sensors are faulty. We therefore assume:

- Agents all follow the same protocol;
- Unreliable agents only lie about their own secret;
- Connections are not sabotaged;
- Unreliability does not evolve;
- Unreliable agents do not remember to whom they lied;
- Agents consider all new information as true until proven otherwise.

In the standard model of (dynamic) gossip, an agent either knows a secret or not. For settings with unreliable agents we need more: agents can also have obtained a wrong secret and thus have a false belief.

To model this, we let secrets be bits and replace the former set of secrets S_a with two sets: X_a for agents of which a received secret 1, and Y_a for agents of which a received secret 0. When an agent is in either X_a or Y_a, then a considers that agent to be reliable. But when an agent is both in X_a and Y_a, then a will consider that agent unreliable.

Definition 7 (UG Graph). *A gossip graph with unreliable agents, short* Unreliable Gossip graph *or* UG graph, *is a quadruple $G = (\mathcal{A}, R, N, S)$ where A is a finite set of agents, $R \subseteq \mathcal{A}$ is the set of reliable agents, $N \subseteq \mathcal{A} \times \mathcal{A}$ is the network relation and $S \colon \mathcal{A} \to \mathcal{P}(\mathcal{A}) \times \mathcal{P}(\mathcal{A})$ assigns to each agent $a \in \mathcal{A}$ a pair (X_a, Y_a). We say that a* has a positive secret of b *iff $b \in X_a$, that a* has a negative secret of b *iff $b \in Y_a$, and that a* knows that b is unreliable *iff $b \in X_a \cap Y_a$.*

We also write S_a for $X_a \cup Y_a$, which intuitively is the set of all agents of which a knows any secret. When all agents are reliable ($R = \mathcal{A}$), a UG graph can be identified with a gossip graph by setting $S_a := X_a \cup Y_a$ for each $a \in \mathcal{A}$.

Definition 8 (Initial UG Graph). *A UG graph $G = (\mathcal{A}, R, N, S)$ is* initial *iff for all $a \in \mathcal{A}$ we have $X_a \cup Y_a = \{a\}$.*

In a regular call ab, where both agents speak the truth, information is shared as follows. This means that both agents update their contact lists (N_a and N_b, respectively) and update their sets X, Y by taking unions. In particular, if before the call agent a had a positive and secret agent b had a negative secret of some agent c, then after the call both a and b know that agent c is unreliable.

Definition 9 (UG Call between reliable agents). *Let $G = (\mathcal{A}, R, N, S)$ be a UG Graph and let $a, b \in \mathcal{A}$ such that $(a, b) \in N$. The* call *ab maps G to $G^{ab} = (\mathcal{A}, R, N^{ab}, S^{ab})$ where N^{ab} is as in Definition 2 and*

$$S_c^{ab} := \begin{cases} (X_a \cup X_b, Y_a \cup Y_b) & \text{if } c \in \{a, b\} \\ (X_c, Y_c) & \text{otherwise} \end{cases}$$

Analogous to Definition 3 we write G^σ for the result of executing a sequence of calls σ on a UG graph G.

Note that in the definition of a call, agents are naive: they consider all new information completely trustworthy and update their knowledge accordingly. In other settings where unreliable agents may lie about secrets of other agents, one can imagine that agents would adopt a more sceptic approach or prefer first hand information (an agent sharing their own secret) over second hand information (and agent sharing a secret of another agent).

5 Unreliable Agents

We now formally define unreliable agents that satisfy the constraints given in Sect. 4. An unreliable agent may report a wrong value of their own secret in a call. We do not assume any rules about when and how often an unreliable agent reports the wrong value of their secret, only that the probability to lie is non-zero (for when the probability is zero, it is a reliable agent).

In addition to the call ab from Definition 9, we now define three calls Ab, aB or AB in which respectively a, b or both agents report the wrong value of their own secret. That is, the agents denoted with a capital letter are lying about their own secrets in this call. For example, in a call Ab all secrets are shared normally, *apart from agent a's secret.* More specifically, if $a \in X_a$ then the new set of secrets for b is not given by merging X_a with X_b, and Y_a with Y_b, but by merging $X_a \setminus \{a\}$ with X_b, and $Y_a \cup \{a\}$ with Y_b. The lying of agent a is thus represented by acting as if her own secret was in Y_a and not in X_a (or vice versa).

Note that Ab, Ba and AB can only occur if, respectively, a, b or both agents do not belong to the set of reliable agents R. On the other hand, note that in the call Ab agent b does not necessarily belong to R, but might still be unreliable and just happen to speak the truth in this call.

Definition 10 (UG Call with unreliable agents). *Let $G = (\mathcal{A}, R, N, S)$ be a UG Graph and let $a, b \in \mathcal{A}$ such that $(a, b) \in N$. We define four calls.*

ab *The call ab maps G to $G^{ab} = (\mathcal{A}, R, N^{ab}, S^{ab})$ from Definition 2.*

Ab *Suppose $a \notin R$. The* call *Ab maps G to $G^{Ab} = (\mathcal{A}, R, N^{Ab}, S^{Ab})$ where $N^{Ab} := N^{ab}$ from Definition 2, and for agents a and b:*

$$S_a^{Ab} := (X_a \cup X_b, Y_a \cup Y_b) \tag{1}$$

$$S_b^{Ab} := \begin{cases} ((X_a \setminus \{a\}) \cup X_b, Y_a \cup \{a\} \cup Y_b) & \textit{if } a \in X_a \setminus Y_a \\ (X_a \cup \{a\} \cup X_b, (Y_a \setminus \{a\}) \cup Y_b) & \textit{if } a \in Y_a \setminus X_a \\ (X_a \cup X_b, Y_a \cup Y_b) & \textit{if } a \in X_a \cap Y_a \end{cases} \tag{2}$$

and $S_c^{Ab} := (X_c, Y_c)$ for all other agents $c \notin \{a, b\}$.

aB *Vice versa, suppose $b \notin R$. The* call *aB maps G to G^{aB} which is defined symmetrically, i.e. the same as G^{Ba}.*

AB *Finally, suppose $a \notin R$ and $b \notin R$. The* call *AB maps G to $G^{AB} = (\mathcal{A}, R, N^{AB}, S^{AB})$ where $N^{AB} := N^{ab}$ from Definition 2 and for a and b:*

$$S_a^{AB} := \begin{cases} (X_a \cup (X_b \setminus \{b\}), Y_a \cup Y_b \cup \{b\}) & \textit{if } b \in X_b \setminus Y_b \\ (X_a \cup X_b \cup \{b\}, Y_a \cup (Y_b \setminus \{b\})) & \textit{if } b \in Y_b \setminus X_b \\ (X_a \cup X_b, Y_a \cup Y_b) & \textit{if } b \in X_b \cap Y_b \end{cases} \tag{3}$$

$$S_b^{AB} := \begin{cases} ((X_a \setminus \{a\}) \cup X_b, Y_a \cup \{a\} \cup Y_b) & \textit{if } a \in X_a \setminus Y_a \\ (X_a \cup \{a\} \cup X_b, (Y_a \setminus \{a\}) \cup Y_b) & \textit{if } a \in Y_a \setminus X_a \\ (X_a \cup X_b, Y_a \cup Y_b) & \textit{if } a \in X_a \cap Y_a \end{cases} \tag{4}$$

and $S_c^{AB} := (X_c, Y_c)$ for all other agents $c \notin \{a, b\}$.

Analogous to Definition 3 we write G^σ for the result of executing a sequence of reliable or unreliable calls σ on a UG graph G.

We stress that an unreliable agent will not always report the wrong value. In fact, then it would be the same as a reliable agent with the opposite secret value, and the other agents would never find out that the unreliable agent is lying.

To illustrate the different types of calls, consider the following example.

Example 3. Consider the UG graph $G = (\mathcal{A}, R, N, S)$ where $\mathcal{A} = \{a, b, c, d\}$, $R = \{c, d\}$, $N = \mathcal{A} \times \mathcal{A}$ and $S_x = (X_x, Y_x) = (\{x\}, \varnothing)$ for each $x \in \mathcal{A}$. The (LNS-permitted) call sequence $AB; ac; Ad; cd; bc$ changes G as follows:

	(X_a, Y_a)	(X_b, Y_b)	(X_c, Y_c)	(X_d, Y_d)
	$(\{a\}, \varnothing)$	$(\{b\}, \varnothing)$	$(\{c\}, \varnothing)$	$(\{d\}, \varnothing)$
$\xrightarrow{AB}$	$(\{a\}, \{b\})$	$(\{b\}, \{a\})$	$(\{c\}, \varnothing)$	$(\{d\}, \varnothing)$
$\xrightarrow{ac}$	$(\{a, c\}, \{b\})$	$(\{b\}, \{a\})$	$(\{a, c\}, \{b\})$	$(\{d\}, \varnothing)$
$\xrightarrow{Ad}$	$(\{a, c, d\}, \{b\})$	$(\{b\}, \{a\})$	$(\{a, c\}, \{b\})$	$(\{c, d\}, \{a, b\})$
$\xrightarrow{cd}$	$(\{a, c, d\}, \{b\})$	$(\{b\}, \{a\})$	$(\{a, c, d\}, \{a, b\})$	$(\{a, c, d\}, \{a, b\})$
$\xrightarrow{bc}$	$(\{a, c, d\}, \{b\})$	$(\{a, b, c, d\}, \{a, b\})$	$(\{a, b, c, d\}, \{a, b\})$	$(\{a, c, d\}, \{a, b\})$

In particular, after the fourth call cd the agents c and d learn that a is unreliable. However, even after the last call, agent d does not know this about b and no more call is permitted according to the LNS protocol.

Interestingly, a consequence of Definition 10 is that agents may find out themselves that they are unreliable. This is what happens after the call bc in Example 3 for agent b: after this call, $b \in X_b \cap Y_b$, hence she considers herself unreliable. But this also informs her that she is uncovered by agent c, who learns the same information about the unreliability of b. If now another agent e enters the network and the call be (or Be) takes place, e will be informed by agent b of her own unreliability. This results from Definition 10: the last clauses of Eqs. 2, 3 and 4 enforce that, in a call between a and b, whenever a is uncovered, i.e. $a \in X_a \cap Y_a$, the sets X_a and X_b and Y_a and Y_b are merged without adjustments. Hence afterwards $a \in X_b \cap Y_b$, i.e. b learns that a is unreliable.

In our setting where unreliability is unintended random memoryless noise this definition is not problematic, but in fact can help the network to perform better. In a network of sensors for instance, the unreliable sensor could then give a signal that it needs to be fixed.

If agents are intentionally unreliable, it might be more realistic to change their behavior once they learn they are uncovered. To model this we could easily change the last clauses of Eqs. 2, 3 and 4 in Definition 10 to

$$(X_a \cup X_b, Y_a \setminus \{a\} \cup Y_b) \tag{2'}$$

$$(X_a \cup X_b, Y_a \setminus \{a\} \cup Y_b) \tag{3'}$$

$$(X_a \cup X_b, Y_a \cup Y_b \setminus \{b\}) \tag{4'}$$

respectively. Similarly, we could do the same but remove a (resp. b) from X_a (resp. X_b) instead of Y_a (resp. Y_b), but the effect would be analogous. In that situation, an uncovered agent will only continue to communicate one value of her secret (here X_a). In other words, once uncovered she will change her behavior.

A simple example of an unreliable agent is an *alternating bluffer* that "lies" in every second call. It provides a first approach to random unintended noise, but it is deterministic and thus easier to simulate and reason about. We note that agent a in Example 3 behaves as an alternating bluffer.

In our model there is no "curing" or "going back" from unreliability. Once an agent is unreliable and consequently (possibly) uncovered, there is no way for agents to change their behavior. This is sufficient to introduce unreliability into Dynamic Gossip and explore whether the known results continue to hold. But of course, for practical applications, it would be desirable to enable agents to be cured. For example for the application of this framework to the spread of diseases [9]. An important question is then how agents can convince others that they have improved their behavior, from unreliable to reliable.

6 Unreliable Success

We now define what it means to be successful in Unreliable Gossip. Completeness on UG graphs is reached when all agents know all secrets, now in the sense that each agent knows at least one secret of each other agent. We note that this is equivalent to completeness on gossip graphs as defined in Definition 1 with $S_a = X_a \cup Y_a$.

Definition 11. *A UG graph $G = (\mathcal{A}, R, N, S)$ is* complete *iff for all agents $a \in \mathcal{A}$ we have $X_a \cup Y_a = \mathcal{A}$.*

However, for Unreliable Gossip this kind of completeness and success according to Definition 5 is not a useful goal. Instead, *the aim of the reliable agents should be to reach completeness among themselves and to identify all unreliable agents*. We now define *reliable completeness* formally and argue that it is a more intuitive goal in the setting of Unreliable Gossip than (mere) completeness.

Definition 12. *A UG graph $G = (\mathcal{A}, R, N, S)$ is* reliably complete *iff for all $a \in R$ we have (i) $X_a \cup Y_a \setminus (X_a \cap Y_a) = R$, and (ii) $X_a \cap Y_a = \mathcal{A} \setminus R$.*

That is, a graph is reliably complete iff each reliable agent (i) knows the secrets of all reliable agents and (ii) knows for all unreliable agents that they are unreliable. We note that in the presence of (ii) the condition (i) is equivalent to $X_a \cup Y_a = \mathcal{A}$. To make it easier to refer to the second condition we also say that *an agent a identifies the unreliable agents* iff $X_a \cap Y_a = \mathcal{A} \setminus R$.

Note that reliably complete does not imply complete, because in a reliably complete graph the unreliable agents do not have to know all secrets. Reliable agents should learn all secrets and identify all unreliable agents, but we do not care at all about what unreliable agents learn. Also vice versa, completeness does not imply reliable completeness, because completeness says nothing about knowing which other agents are unreliable.

In order to compare completeness on unreliable networks to completeness on normal networks, we define *reliable counter-graphs* and *reliable subgraphs.*

Definition 13. *Let* $G = (\mathcal{A}, R, N, S)$ *be a UG Graph. Then we define its* reliable counter-graph $G^* := (\mathcal{A}, N, S^*)$ *where* $S^*_a := (X_a \cup Y_a) \setminus (X_a \cap Y_a)$*. And we define its* reliable subgraph $G|_R := (\mathcal{A}|_R, N|_R, S|_R)$ *where* $\mathcal{A}|_R := R$*,* $N|_R := N \cap (R \times R)$ *and* $(S|_R)_a := (X_a \cup Y_a) \cap R$*.*

A sanity check shows that indeed both G^* and $G|_R$ are gossip graphs. Definition 13 allows us to rephrase the definition of reliable completeness: a graph G is reliably complete if and only if the reliable subgraph of G is complete and all reliable agents identify the unreliable agents.

To conclude this section, we define success for Unreliable Gossip, both for the original notion of completeness and reliable completeness.

Definition 14. *Suppose we have a UG graph* G *and a call sequence* σ *which can be executed on* G*. We say that* σ *is* successful *on* G *iff* G^σ *is complete and we say that* σ *is* reliably successful *on* G *iff* G^σ *is reliably complete.*

A protocol is (reliably) weakly/strongly/un-successful *on a graph* G *iff all/some/no sequences permitted by the protocol and executable on* G *are (reliably) successful on* G*.*

6.1 LNS Is Not Reliably Successful on Sun Graphs

Here we show that, already with a small amount of unreliability, for example in the form of the alternating bluffer, a known result about LNS [7] fails to hold. Specifically, we show that on UG graphs that are sun graphs with only terminal unreliable agents, LNS fails to identify the unreliable agents in the sense that it is not reliably successful as defined in the previous section. Before the general result we give an example where the classification of LNS fails to hold.

Example 4. Consider again the sun graph from Example 2 and suppose a is unreliable. Now consider the sequence $bc; ba; cA$. This is an LNS sequence resulting in a complete graph. However, if a is an alternating bluffer, then b will learn one value of the secret of a and c the other. Formally, in the resulting graph $G^{bc;ba;cA}$ we have $a \in X_c \setminus Y_c$ and $a \in Y_b \setminus X_b$. Unfortunately, LNS allows no further calls. Hence b and c may no longer communicate and will not notice that a is unreliable.

Consider $ba; Ac; bc$. This is also an LNS sequence which can be executed on the graph above. But in this case b and c talk to each other *after* having learned different values from a and will thus find out that a is unreliable. Formally, in the resulting graph $G^{ba;Ac;bc}$ we have $a \in (X_b \cap Y_b)$ and $a \in (Z_c \cap Y_c)$.

Hence, whether b and c find out that a is unreliable depends on the sequence.

Example 4 already suffices to show that LNS is not reliably successful on all sun graphs when we have unreliable agents. However, we now prove something slightly stronger, namely that for all graphs of a similar shape there is a maximal sequence which is not successful.

Theorem 2. *Consider any initial UG graph with at least one unreliable agent. If all unreliable agents are terminal then LNS is not reliably strongly successful.*

Intuitively, Theorem 2 holds because there are call sequences in which the unreliable agents are called too late, so that the reliable agents cannot verify the secrets of these unreliable agents with each other. This is the case in Example 4: the reliable agents b and c first learn each others' secrets before calling the unreliable agent a. But then b and c cannot call each other again in LNS and hence cannot verify the secret of a with each other. That is why they fail to identify c as unreliable.

We now first prove a lemma.

Lemma 2. *Suppose $G = (\mathcal{A}, R, N, S)$ is an initial UG graph with at least one unreliable agent. Moreover, suppose that all unreliable agents in G are terminal. Then for any LNS-permitted call sequence σ we have: if there is a prefix $\tau \sqsubseteq \sigma$ such that $G^\tau|_R$ is complete but G^τ is not reliably complete, then also G^σ is not reliably complete and thus σ is not reliably successful on G.*

Lemma 2 states that any LNS sequence cannot become reliably successful any more as soon as it reaches a complete reliable subgraph. Intuitively, once the reliable subgraph becomes complete, the reliable agents can no longer call each other to compare secrets they received from the unreliable agents.

Proof (*of Lemma 2*). Let G be an initial UG graph with at least one unreliable agent and where all unreliable agents are terminal. Let σ be an LNS-permitted call sequence with a prefix $\tau \sqsubseteq \sigma$ such that $G^\tau|_R$ is complete. Then $\forall r \in R : X_r^\tau \cup Y_r^\tau \supseteq R$ and therefore also $\forall r \in R : X_r^\sigma \cup Y_r^\sigma \supseteq R$ because no contradictory information can be learned about reliable agents.

Now note that after the call sequence τ no more calls from an unreliable agent to a reliable agent can take place: just after τ the unreliable agents are still terminal, and in all later calls where they learn the number of a reliable agent they will also learn the secret of that same agent (because $G^\tau|_R$ is complete). Moreover, we can ignore calls between unreliable agents because they do not affect reliable completeness.

Hence, let ab be the last call to take place in σ from a reliable agent a to an unreliable agent b. Let $\sigma \setminus ab$ denote the sequence without this last call. That means before the call a knew no secret of b, i.e. $b \notin X_a^{\sigma\setminus ab} \cup Y_a^{\sigma\setminus ab}$. But then, because a will not be involved in any later calls, we have that $b \in X_a^\sigma \cup Y_a^\sigma$. Hence agent b will not be identified by agent a and σ is not reliably successful on G.

Proof (of Theorem 2). Let $G = (\mathcal{A}, R, N, S)$ be an initial UG graph that is a sun graph where all unreliable agents are terminal. Because all unreliable agents are terminal, the reliable subgraph $G|_R$ of G must be a sun graph too. By Theorem 1, any maximal LNS-permitted call sequence τ consisting of calls ab with $a, b \in R$ will complete $G|_R$, i.e. $G^\tau|_R$ is complete. Now by Lemma 2, any LNS-permitted call sequence σ extending τ will fail to identify all unreliable agents and hence fail to reliably complete the network.

Thus, we cannot extend the sun graph characterisation of LNS to Unreliable Gossip. This already holds for a small amount of unreliability: one terminal alternating bluffer. Of course, this is because we now also demand that reliable agents identify the unreliable agents. If we only care about completeness in the original sense, then LNS is still strongly successful on UG graphs *with respect to the reliable agents*. In particular, even if unreliable agents are involved in earlier calls (i.e. if there is no τ as in the proof above), the reliable subgraph will still be completed.

6.2 Blocking Unreliable Agents Hides and Helps Them

How can we "repair" LNS to deal with unreliable agents? Intuitively, blocking unreliable agents seems a good measure against the spread of false information in networks because it would prevent unreliable agents from spreading their false information. This would mean that, when an unreliable agent performs a call to another agent, her call will be rejected.

By blocking unreliable agents, their communicative power is restricted: they will not be able to initiate calls – whenever they do, they are rejected. Of course, conceptually, there is a difference between blocked agents and agents that are not able to initiate communication. The latter may rather occur whenever their communicating device is broken. Yet, mathematically, these situations are analogous: in both situations, the unreliable agents cannot successfully make a call to another agent. Therefore we evaluate the following protocol that limits the unreliable agents in their ability to make calls to discuss whether blocking unreliable agents is indeed a good measure.

Definition 15 (Protocol LNSR). *A call ab is* LNSR-permitted *iff $(a, b) \in N$, $a \in R$ and $(a, b) \notin S$.*

But, against the intuition, the protocol LNSR does not only prevent false information from spreading, it might also prevent unreliable agents from being detected by the reliable agents. Specifically, we prove that unreliable agents that are not allowed to initiate any form of communication are harder to identify than unreliable agents that are. In other words, unreliability can be easier detected when it is spread more. Therefore the restriction to disable, via blocking, the unreliable agents from initiating calls is not desirable.

Theorem 3. *LNSR is a proper strengthening of LNS in the following sense:*

(i) For any UG graph G we have: If LNSR is (reliably) weakly successful on G, then also LNS is (reliably) weakly successful on G.

(ii) There is a UG graph G where LNSR is not reliably weakly unsuccessful, but where LNS is reliably weakly successful.

Proof. (i) Note that any LNSR-permitted call sequence σ is also LNS-permitted. If LNSR is (reliably) weakly successful on some UG graph G, then there is an LNSR-permitted call sequence σ such that G^σ is (reliably) complete. But then σ is also LNS-permitted, and hence LNS is also (reliably) weakly successful on G.

(ii) Consider the UG graph $G = (\mathcal{A}, R, N, S)$ below with $\mathcal{A} = \{a, b, c\}$, $R = \{a, b\}$, $N = \{(b, a), (b, c)\}$ and $S_x = (\{x\}, \varnothing)$ for all $x \in \mathcal{A}$.

$$\begin{tikzcd} a & b \arrow[l, dashed] \arrow[r, dashed] & c \end{tikzcd}$$

Then the following are all the LNS-permitted call sequences on G. For each sequence we list four variants, depending on where c is lying.

1. $ba; ac; bc$ or $ba; aC; bc$ or $ba; ac; bC$ or $ba; aC; bC$
2. $ba; bc; ac$ or $ba; bC; ac$ or $ba; bc; aC$ or $ba; bC; aC$
3. $bc; ca; ba$ or $bC; ca; ba$ $(*)$ or $bc; Ca; ba$ or $bC; Ca; ba$
4. $bc; ba; ca$ or $bC; ba; ca$ or $bc; ba; Ca$ or $bC; ba; Ca$

Only the call sequences under 1 and 2 are LNSR-permitted. But only the sequence marked with $*$ reliably completes the network: first the agents a and b need to learn different values from agent c and after that they should communicate with each other to learn that c is unreliable. None of the other sequences reliably complete the network and in particular no LNSR-permitted call sequence reliably completes G. Hence LNS is reliably weakly successful on G, but $LNSR$ is not.

It is crucial in part (ii) of Theorem 3 that reliable agents are the last to communicate in order to identify the unreliable agent as such. Thus the success of the protocol is dependent on the call sequence, and in particular on the position of calls between reliable agents: they need to verify the secrets of the unreliable agents. But, agents do not know which agents are the unreliable agents (this is the goal of the protocol), hence they do not know which secrets need to be verified nor with whom to verify this.

This problem of verification is similar to the Byzantine Generals Problem [13] developed to describe a situation in which agents must agree on a joint strategy to avoid catastrophic failure of the system, but where some of the agents or some are unreliable. In a simple form, multiple generals are threatened by a common enemy and they each have to decide whether to attack or to retreat with a preferred outcome of a coordinated attack or coordinated retreat. A good solution to the problem is an algorithm that can both guarantee that all reliable generals decide upon the same plan and that a small number of unreliable generals cannot cause the reliable generals to adopt a bad strategy. Such solutions have been studied in the literature under the name of Byzantine Fault Tolerance, starting with [6] and more recently including [4].

Theorem 3 illustrates that there are networks on which unreliable agents remain unidentified when they are not allowed to initiate calls, but can be identified when they do initiate calls. This has direct consequences for the security measure to block unreliable agents and raises questions about their effectiveness for real-life applications and gossip-like settings. For example, a faulty sensor should not be shut down immediately but continue to communicate such that it will be identified as faulty by a larger number of other sensors. As another example, fake news articles shared in social networks will be easier to uncover and identify if they are *not* removed or blocked, but instead marked as fake and continued to be actively shared.

Formally, we define the ideas of blocking and deleting as follows. Deleting means that an agent removes those agents she knows to be unreliable from her own phone book. Blocking means that, in addition to deleting, the agent removes her own number from the phone book of agents she knows to be unreliable.

Definition 16 (Delete and Block). *Let $G = (\mathcal{A}, R, N, S)$ be a UG graph and let $a \in \mathcal{A}$. The* delete *action λ_a maps G to $G^{\lambda_a} = (\mathcal{A}, R, N^{\lambda_a}, S)$ and the* block *action μ_a maps G to $G^{\mu_a} = (\mathcal{A}, R, N^{\mu_a}, S)$, which are defined by*

$$N_c^{\lambda_a} := \begin{cases} N_c & \textit{if } c \neq a \\ N_c \setminus (X_c \cap Y_c) & \textit{if } c = a \end{cases}$$

$$N_c^{\mu_a} = \begin{cases} N_c & \textit{if } c \neq a \textit{ and } c \notin X_a \cap Y_a \\ N_c \setminus \{a\} & \textit{if } c \neq a \textit{ and } c \in X_a \cap Y_a \\ N_c \setminus (X_c \cap Y_c) & \textit{if } c = a \end{cases}$$

As Theorem 3 shows, blocking unreliable agents, though seemingly a good approach to prevent the spread of false information, comes at a cost. Blocking unreliable agents seems analogous to restricting their communicative power because the effects are the same: unreliable agents will not be able to initiate communication. This is exactly what has been shown to help them remain unidentified.

However, in contrast to LNSR, let us now assume that agents only block other agents once they have identified them as unreliable. They will then be able to forward the information that agents are unreliable to others. The following example illustrates how this can prevent the unwanted effect of hiding unreliable agents.

Example 5. Consider the following network of four agents with one unreliable agent, agent b, i.e. $\mathcal{A} = \{a, b, c, d\}$ and $R = \{a, c, d\}$:

$$\begin{tikzcd} a \arrow[r, dashed] \arrow[rr, dashed, bend left] & b & c \arrow[l, dashed] \arrow[r, dashed] & d \end{tikzcd}$$

Suppose that the reliable agents block the unreliable agents as soon as they identify them. Consider the LNS-permitted call sequence $ab; cB; ac; cd$. After the subsequence $ab; cB; ac$ the agents a and c will identify agent b as unreliable and this will then be communicated to agent d in the final call cd. Hence also agent d will block b after this and the whole sequence is reliably successful.

A disadvantage of both LNSR and blocking known-to-be-unreliable agents is that it might exclude other reliable agents "behind" unreliable ones. Whenever there is a reliable agent a that is only able to communicate with an unreliable agent, blocking this unreliable agent also prevents agent a to contact the rest of the network. She is therefore excluded from the rest of the network. Consider the following example.

Example 6. Let $G = (\mathcal{A}, R, N, S)$ be the network drawn below where $R = \mathcal{A} \setminus \{b\}$ and $\mathcal{A} \setminus \{a, b\}$ forms a complete cluster, i.e. $\forall r \in \mathcal{A} \setminus \{a, b\}$: $X_r \cup Y_r = \mathcal{A} \setminus \{b\}$. Suppose further that all agents in the cluster consider b unreliable, i.e. $\forall r \in \mathcal{A} \setminus \{a, b\} : b \in X_r \cap Y_r$, and have no information about a, that b has all the information about $\mathcal{A} \setminus \{a\}$ and that a only has the phone number of b, as drawn below. Then blocking agent b effectively blocks agent a and the network will not be reliably completed. We argue that this is a realistic scenario for LNS: the call ab might come too late in the call sequence. Then, because the other reliable agents block agent b, agent a is also blocked indirectly.

$$\begin{tikzcd} A \setminus \{a, b\} \arrow[r, leftrightarrow] & b & a \arrow[l, dashed] \end{tikzcd}$$

7 Discussion and Conclusion

We extended the formal model of Dynamic Gossip from [7] to include unreliable agents. To better capture success in Dynamic Gossip with unreliable agents we defined the notion of reliable success: all reliable agents should learn all secrets and they should identify the unreliable agents. We have then shown that, already with a single unreliable agent, we cannot extend the results about the success of the LNS protocol: LNS is successful in the old sense, but not reliably successful on sun graphs with unreliable terminal agents. This shows that the assumption that *everybody is reliable* is crucial for the success of LNS and that LNS should be adapted for practical applications where agents might fail.

We then examined a way to counter the spread of false information, namely to restrict communication of unreliable agents. It turns out that unreliable information that is not actively spread is harder to identify than unreliable information that is actively spread. This has seemingly paradoxical consequences for measures against unreliable agents: blocking can have a contrary effect and help unreliable agents to remain unidentified. Thus, there is a pay-off between identifying and containing false information.

Our framework and in particular the alternating bluffer are of course simplistic and there are many ways to extend this work: agents can also be unreliable or (with intent) lie about other agents' secrets, about phone numbers, about their own knowledge, etc. Yet, we see this work as a starting point for the discussion of reliability and unreliability in dynamic gossip and its real life applications. We thus end this article with the following open questions.

- What is the class of unreliable gossip graphs characterized by LNS?
- Is there any limitation on the position of unreliable agents in this class?
- Is there an LNS weakening or strengthening (in the sense of [8]) that performs better in situations with unreliability?

Further research will show how Dynamic Gossip protocols can be adapted to deal with other forms of unreliability.

Finally, we want to stress that this work is not purely theoretical: social media and the spread of fake news can be seen as an instance of gossip with unreliable agents. Some social networks already use hybrid strategies where false information is not blocked but just marked as such.

Acknowledgements. This work is based on the master's thesis of the first author [5], supervised by Jan van Eijck. We thank Hans van Ditmarsch and the anonymous reviewers at the DaLí workshop for helpful feedback.

References

1. Allcott, H., Gentzkow, M.: Social media and fake news in the 2016 election. J. Econ. Perspect. **31**(2), 211–268 (2017). https://doi.org/10.1257/jep.31.2.211
2. Apt, K.R., Grossi, D., van der Hoek, W.: Epistemic protocols for distributed gossiping. In: Proceedings TARK 2015. EPTCS, vol. 215, pp. 51–66 (2015). https://doi.org/10.4204/EPTCS.215.5
3. Attamah, M., van Ditmarsch, H., Grossi, D., van der Hoek, W.: Knowledge and gossip. In: Proceedings of the Twenty-first European Conference on Artificial Intelligence, pp. 21–26 (2014). https://doi.org/10.3233/978-1-61499-419-0-21
4. Baird, L.: The swirlds hashgraph consensus algorithm: fair, fast, Byzantine fault tolerance (2017). https://www.swirlds.com/downloads/SWIRLDS-TR-2016-01.pdf
5. van den Berg, L.: Unreliable gossip (2018). https://eprints.illc.uva.nl/1597/, Master's thesis, University of Amsterdam
6. Castro, M., Liskov, B.: Practical Byzantine fault tolerance. In: Proceedings of the Third Symposium on Operating Systems Design and Implementation, OSDI 1999, pp. 173–186 (1999). https://www.usenix.org/legacy/events/osdi99/castro.html
7. van Ditmarsch, H., van Eijck, J., Pardo, P., Ramezanian, R., Schwarzentruber, F.: Dynamic gossip. Bull. Iran. Math. Soc. **45**(3), 701–728 (2018). https://doi.org/10.1007/s41980-018-0160-4
8. van Ditmarsch, H., Gattinger, M., Kuijer, L.B., Pardo, P.: Strengthening gossip protocols using protocol-dependent knowledge. J. Appl. Logics - IfCoLog J. Logics Appl. **6**(1) (2019). https://arxiv.org/abs/1907.12321
9. Eugster, P.T., Guerraoui, R., Kermarrec, A.M., Massoulié, L.: Epidemic information dissemination in distributed systems. Computer **37**, 60–67 (2004). https://doi.org/10.1109/MC.2004.1297243
10. Harary, F., Schwenk, A.J.: The communication problem on graphs and digraphs. J. Franklin Inst. **297**, 491–495 (1974). https://doi.org/10.1016/0016-0032(74)90126-4

11. Hedetniemi, S.M., Hedetniemi, S.T., Liestman, A.L.: A survey of gossiping and broadcasting in communication networks. Networks **18**(4), 319–349 (1988). https://doi.org/10.1002/net.3230180406
12. Hoefler, T., Barak, A., Shiloh, A., Drezner, Z.: Corrected gossip algorithms for fast reliable broadcast on unreliable systems. In: 2017 IEEE International Parallel and Distributed Processing Symposium (IPDPS), pp. 357–366 (2017). https://doi.org/10.1109/IPDPS.2017.36
13. Lamport, L., Shostak, R., Pease, M.: The Byzantine generals problem. ACM Trans. Program. Lang. Syst. (TOPLAS) **4**(3), 382–401 (1982). https://doi.org/10.1145/3335772.3335936
14. Shi, G., Johansson, M., Johansson, K.H.: Randomized gossiping with unreliable communication: dependent or independent node updates. In: 2012 IEEE 51st IEEE Conference on Decision and Control (CDC), pp. 4846–4851 (2012). https://doi.org/10.1109/CDC.2012.6426729
15. Tijdeman, R.: On a telephone problem. Nieuw Archief voor Wiskunde **3**(19), 188–192 (1971)
16. Wang, H., Liao, X., Wang, Z., Huang, T., Chen, G.: Distributed parameter estimation in unreliable sensor networks via broadcast gossip algorithms. Neural Netw. **73**, 1–9 (2016). https://doi.org/10.1016/j.neunet.2015.09.008
17. Watts, D.J., Strogatz, S.H.: Collective dynamics of ‘small-world’ networks. Nature **393**(6684), 440–442 (1998). https://doi.org/10.1038/30918