

An Analytical Model of Information Dissemination for a Gossip-based Protocol

Rena Bakhshi ^{a,*} Daniela Gavidia ^a Wan Fokkink ^a
Maarten van Steen ^a

^a*Department of Computer Science, Vrije Universiteit Amsterdam, The Netherlands*

Abstract

We develop an analytical model of information dissemination for a gossiping protocol that combines both pull and push approaches. With this model we analyse how fast an item is replicated through a network, and how fast the item covers the network. We also determine the optimal size of the exchange buffer, to obtain fast replication. Our results are confirmed by large-scale simulation experiments.

1 Introduction

Today, large-scale distributed systems consisting of thousands of nodes are commonplace, due to the wide availability of high-performance and low-cost devices. In practice, these systems are often diagnosed through performing simulations to discover correlations between design parameters and observed behaviour. Such experimental results provide essential data on system behaviour, and can aid in understanding the emergent behaviour of the system. However, experiments can be time consuming and the infinite space of the parameter settings for probabilistic systems is often too large to be explored experimentally. Consequently, the experiments do not always clarify how parameter settings influence the extra-functional properties of the system. As a result, it is very difficult to predict what the effects of certain design decisions are, as it is practically infeasible to explore the full range of input data. A challenge is to develop analytical models that capture (part of) the behaviour

* Corresponding author

Email addresses: rbakhshi@few.vu.nl (Rena Bakhshi), daniela@few.vu.nl (Daniela Gavidia), wanf@few.vu.nl (Wan Fokkink), steen@few.vu.nl (Maarten van Steen).

of a system, and then subsequently optimise design parameters following an analytical rather than an experimental approach.

We are interested in developing and validating analytical models for gossip-based systems (cf. [5]). These systems rely on epidemic techniques for the communication and exchange of information. These communication protocols, while having simple specifications, show complex and often unexpected behaviour when executed on a large scale (e.g. [22,14]). Our analytical models of gossip protocols need to be realistic, yet, sufficiently abstract to allow for easy prediction of systems behaviour. By ‘realistic’ we mean that they can be applied to large-scale systems and can capture functional and extra-functional behaviour such as replication, coverage, convergence, and other system dynamics (see [16]). Such models are amenable for mathematical analysis, to make precise predictions. Furthermore, we will exploit the fact that because an analytical model presents an abstraction of the original protocol, a simulation of the model tends to be much more efficient (in computation time and memory consumption) than a simulation of an implementation of this protocol.

In this paper, we develop an analytical model of an epidemic protocol by Gavidia et al. [17], which was originally developed to disseminate data items to a collection of small devices in a decentralised fashion. A decentralised solution considerably decreases the probability of information loss or unavailability that may occur due to a single point of failure, or high latency due to the overload of a node. Nodes executing the protocol periodically contact each other, and exchange data items. Concisely, a node initiates a contact with a random neighbour, pulls a random subset of items from the contacted node, simultaneously pushing its own random subset of items. This push/pull approach has a better performance than a pure push or pull approach [22,23]. The amount of information exchanged during each contact between two communicating nodes is limited. Replication ensures the availability of the data items even in the face of dynamic behaviour. Thus, nodes not only conserve the data collectively stored in the network, but also relocate it in a random fashion; hence, nodes will eventually see all data items.

The central point of our study is a thorough probabilistic analysis of information dissemination in a large-scale network using the aforementioned protocol. Our modelling framework for a gossip protocol differs from the ones, presented by other papers in that we do not follow the traditional modelling using the mathematical theory of epidemics [16]. Instead, the behaviour of the protocol is modelled at an abstract level as pairwise node interactions. When two neighbouring nodes interact with each other (gossip), they may undergo a state transition (exchange items) with a certain probability. The transition probabilities depend on the probability that a given item in a node’s local storage has been replaced by another item after the exchange. We calculated accurate values for these probabilities, yielding a rather complicated expres-

sion. This expression depends not only on the amount of items, message size and local storage size, but also on the amount of items both gossiping nodes have in common, in particular, how many of such items the contacted node receives during the exchange. The expression is complex because it incorporates the expected value of the amount of such items, and a connection between the parameters is not obvious. We also determined a close approximation that is expressed by a much simpler formula, as well as a correction factor for this approximation allowing for precise error estimations. Thus we obtain a better understanding of the emergent behaviour of the protocol and how parameter settings influence its extra-functional behaviour.

We investigated two properties characterising the protocol, namely, the number of nodes that have ‘seen’ a given item over time (coverage), and the number of replicas of this item in the network at a certain moment in time (replication). Using the values of the transition probabilities, we determined the optimal number of items to exchange per gossip, for a fast convergence of coverage and replication. Moreover, we determined formulas that capture the dissemination of an item in a fully connected network. All our modelling and analysis results are confirmed by large-scale simulations, in which simulations based on our analytical models are compared with running the actual protocol. To the best of our knowledge, we are the first to develop an accurate, realistic formal model that can be used to optimally design and fine-tune a given gossip protocol. In this sense, our main contribution is demonstrating the feasibility of a model-driven approach to developing real-world gossip protocols.

The paper is structured as follows ¹. The remainder of this introduction discusses related work. Sec. 2 explains the shuffle protocol. In Sec. 3 the analytical model is developed. Sec. 4 discusses the results of our experimental evaluations. Sec. 5 presents a round-based perspective of replication and coverage. In Sec. 6 we discuss the difference (in terms of time complexity) between simulation of the formal model and of the actual protocol. And Sec. 7 contains the conclusions and future work.

Related work

Two areas of research are most relevant to our paper: rigorous analysis of gossip (and related) protocols, and results from mathematical theory of epidemics [4,10]. The results from epidemics are often used in the analysis of

¹ The current paper extends the results previously published in [6], notably the calculation of the precise formula for the probability of dropping an item. It also introduces both a simple and more refined model of the coverage property, accompanied by a thorough simulation study. In addition, we make a comparison between the complexity of the shuffle protocol and the model.

gossip protocols [16] (e.g. the traditional gossiping paper by Demers et al. [12]). We restrict our overview to the most relevant publications from the area of (analysis of) gossip protocols.

Several works have focused on gossip-based membership management protocols. Allavena et al. [2] proposed a gossip-based membership management protocol and analysed the evolution of the number of links between two nodes executing the protocol. The states of the associated Markov chain are the number of links between pairs of nodes. From the designed Markov chain they calculated the expected time until a network partition occurs. This case study also includes a model of the system under churn. A goal of that paper is to show the effect of mixing both pull and push approaches.

Eugster et al. [15] presented a lightweight probabilistic broadcast algorithm, and analysed the evolution of processes that gossip one message. The states of the associated Markov chain are the number of processes that propagate one gossip message. From the designed Markov chain, the authors computed the distribution of the gossiping nodes. Their analysis has shown that the expected number of rounds to propagate the message to the entire system does not depend on the out-degree of nodes. These results are based on the analysis assumption that the individual out-degrees are uniform. However, this simplification has shown to be valid only for small systems (cf. [22]).

Bonnet [8] studied the evolution of the in-degree distribution of nodes executing the Cyclon protocol [29]. The states of the associated Markov chain are the fraction of nodes with a specific in-degree distribution. From the designed Markov chain the author determined the distribution to which the protocol converges.

There are a number of theoretical results on gossip protocols, targeted to a distributed aggregation. In these protocols, a set of data is distributed over the nodes of a network and the nodes compute an aggregate of the data set. Kempe et al. [24] proposed a push-only gossip-based aggregation protocol for the fully connected network. In this paper, the authors used Gaussian mixture modelling [13,26]. A performance of the protocol has been measured by how quickly a data originating with a node diffuses through a network (for uniform gossip). Each node locally maintains an aggregation vector $v_{t,i}$. A state of the associated Markov chain is the fraction of the vector node i sends to other node. From the designed Markov chain, the authors studied the convergence rate. In addition, the authors showed that the diffusion speed for flooding corresponds to the mixing time of a random walk on the network. Validation of the theoretical results with practical experiments is left as a future work.

The protocol [24] has been further tailored by Boyd et al. [9] to work on an arbitrarily connected network. In their analysis, the Markov chain is defined

by a weighted random walk on the graph. Every time step, a pair of nodes (connected by an edge) communicates with a transition probability, and sets their values equal to the average of their current values. A state of the associated Markov chain is a vector of values at the end of the time step. The authors considered the optimisation of the neighbour selection probabilities for each node, to find the fastest-mixing Markov chain (for fast convergence of the algorithm) on the graph.

Jelasyty et al. [21] proposed a push-pull solution for aggregation in large dynamic networks, supported by a performance analysis of the protocol. A state of the system is represented by a vector, the elements of which correspond to the values at the nodes, a target value of the protocol calculated from the vector elements, and a measure of homogeneity characterising the quality of local approximations. The vector evolves at every step of the system according to some distribution. In the analysis, the authors considered different strategies (e.g., neighbour selection) to optimise the protocol implementation, and calculated the expected values for the above mentioned protocol parameters.

Deb et al. [11] studied the adaptation of random network coding to gossip protocols. The authors analysed the expected time and message complexity of two gossip protocols for message transmission with pure push and pure pull communication models.

2 A Gossip-based Protocol for Information Dissemination

This section describes the shuffle protocol, originally introduced in [17] for the dissemination of information in a wireless environment. The protocol itself is, at heart, a simple push-pull gossip protocol which can be used also in wired networks. The protocol disseminates data items of general interest to a collection of nodes. The protocol relies on replication to ensure the availability of data items in the face of dynamic behaviour.

The system consists of a collection of nodes, each of which contributes a limited amount of storage space (which we will refer to as the node's cache) to store data items. The nodes periodically swap (shuffle) data items from their cache with a randomly chosen neighbour. In this way, nodes update their caches on a regular basis, allowing nodes to gradually discover new items as they are disseminated through the network.

Items can be published by any user of the system, and are propagated through the network. An item is a piece of information, and for each item several copies may exist in the network. As items are gossiped between neighbouring nodes, replication may occur when a node has available storage space to keep a copy

of an item it just gossiped to a neighbour.

2.1 Protocol assumptions

All nodes have a common agreement on the frequency of gossiping. However, there is no agreement on when to gossip.

In terms of storage space, we assume that all nodes dedicate the same amount of storage space to keep items locally, and that all items are of the same size. Therefore, we say that each node has a cache size of c . When shuffling, each node sends a fixed number s of the c items in the cache.

The gossip exchange is performed as an atomic procedure, meaning that once a node initiates an exchange with another node, this pair of nodes cannot become involved in another exchange until the current exchange is finished.

2.2 Description

Nodes executing the shuffle protocol initiate a shuffle periodically. In order to execute the protocol, the initiating node needs to contact a gossiping partner. Such a random peer is delivered by an underlying layer that keeps track of the neighbourhood membership. In a wired environment, this service could be provided by, for instance, a peer sampling service [22] running at each node. For wireless environments, the neighbourhood is determined by the radio connectivity between nodes.

We describe the protocol from the point of view of each participating node. We refer to [17] for a more detailed description.

Node A initiates the shuffle by executing the following steps:

- (1) picks a neighbouring node B uniformly at random;
- (2) selects randomly s items from the local cache, and sends a copy of these items to B ;
- (3) receives s items from the local cache of B ;
- (4) checks whether any of the received items are already in its cache; if so, these received items are eliminated;
- (5) adds the rest of the received items to the local cache; if the total number of items exceeds cache size c , removes items at random among the ones that were sent by A to B , but not those that were also received by A from B , until the cache contains c items.

In response to being contacted by A , node B executes the following steps:

- (1) receives s items from the local cache of A ;
- (2) selects randomly s items from its local cache, and sends a copy of these items to A ;
- (3) checks whether any of the received items are already in its cache; if so, these received items are eliminated;
- (4) adds the rest of the received items to the local cache; if the total number of items exceeds cache size c , removes items at random among the ones that were sent by B to A , but not those that were also received by B from A , until the cache contains c items.

According to the protocol, each node agrees to keep the items received from a neighbour. Given the limited storage space available in each node, keeping the items received during an exchange implies discarding some items that the node has in its cache. By picking the items to be discarded from the ones that have been sent to the neighbour, the conservation of data in the network is ensured.

2.3 Properties

We are interested in the characteristics of the dissemination of data items when the protocol is executed at a large scale, i.e. with a large set of nodes. For this reason, we focus on two properties that can be observed in large deployments: i) the number of replicas of an item in the network, and ii) the coverage achieved by an item over time.

2.3.1 Replication

This property is defined as the fraction of nodes that hold a copy of a generic item d in their cache, at a given moment. After an item is introduced into the network, with every shuffle involving a node that has the item in its cache, there is a chance that a new copy of the item will be created, or that the item will be discarded. As a result, with every passing round the number of copies in the network for a particular item fluctuates. Given that the storage space at the nodes is limited, items are in constant competition to place copies in the network. Since competition is fair (all items have the same chance of being replicated or discarded), eventually the storage capacity is evenly divided between the existing items. To be more precise, consider a network of N nodes, in which n different items have been published in total. Since there are $N \cdot c$ cache entries in the network in total, the average number of copies that an individual item has in the network will converge to $\frac{N \cdot c}{n}$. So the fraction of nodes that have a replica of an item in their cache will converge to $\frac{c}{n}$ on average.

2.3.2 Coverage

This property is defined as the fraction of nodes in the network that have seen a generic item d since it was introduced into the network. As explained earlier, several copies of an item are generated after the item is first published. Due to the periodic nature of the protocol, these copies continually move through the network. This results in nodes discovering item d over several rounds. With each passing round, more nodes will have seen d . Eventually, d will have been seen by all nodes (i.e., the coverage is equal to 1). The speed at which the coverage grows is influenced by several factors (as will be explained later on) including the number of different items in the network (i.e. competition), cache size, and the size of the exchange buffer.

2.4 Experimental observations

Before moving on to the analysis of the protocol, we would like to focus on an important aspect that we have observed during our extensive simulation study of the shuffle protocol: the tendency of items to replicate to even levels. In other words, a newly published item generates replicas in the network over time until its number of replicas reaches a level comparable to the number of replicas of other items. This comes as a consequence of the random selection of items to gossip and to keep in local storage. By applying random selection, no item is favoured over the others resulting in a fair division of the storage space. That is, once the system has reached equilibrium, each item in the network will have, on average, the same number of replicas ($\frac{N \cdot c}{n}$).

Figure 1 shows a set of experiments where the distribution of replicas for all the items in the system is tracked over several gossip rounds. In all cases, the network consists of $N = 2500$ nodes with a cache size of $c = 100$ and each node sends $s = 50$ items when it gossips. The number of different items that are inserted in the network is $n = 500$. Each graph shows three curves corresponding to the following initial scenarios:

- Nodes are arranged in a 50×50 grid. The insertion of items to be gossiped occurs simultaneously at round 0. All nodes start with an empty cache, except for 500 nodes randomly chosen nodes. A unique item is placed in the cache of each of these 500 nodes. As a result, at round 0 our network contains 500 different items and each of these items has a single replica.
- Nodes are arranged in a 50×50 grid. The insertion of items to be gossiped occurs at randomly chosen times before round 100. At round 100, all 500 items will be present in the network. However, items that were inserted earlier will have more replicas due to having been shuffled for a longer time.
- Topology with higher density at the center. The insertion of items to be

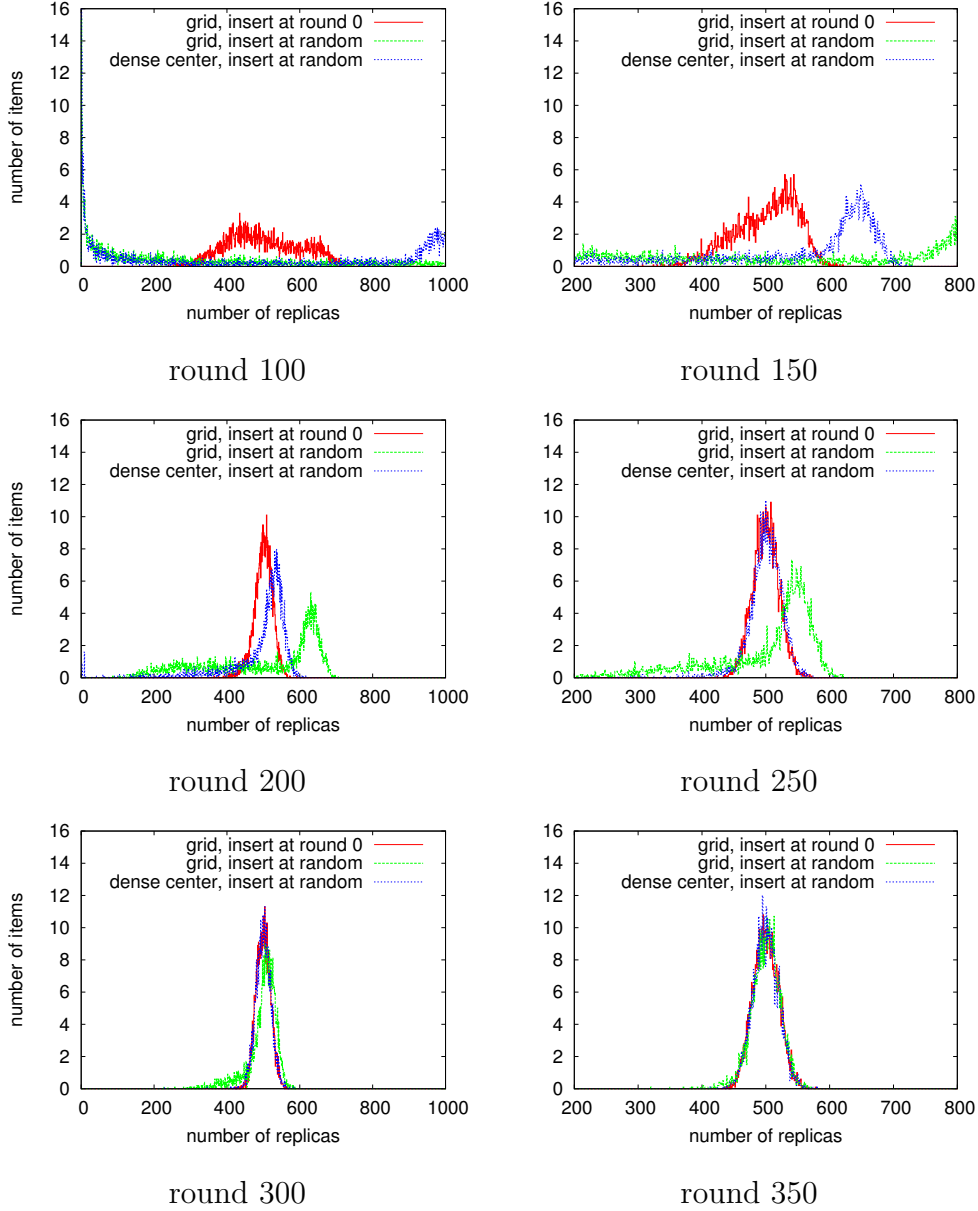


Fig. 1. Distribution of replicas over time.

gossiped occurs at randomly chosen times before round 100. In this topology, nodes near the center have more neighbours to gossip with.

As can be seen from the graphs, the different initial conditions result in different replication patterns early on. However, note how as time progresses the replication patterns become more and more similar. By round 350, it is clear that items have on average $\frac{N \cdot c}{n} = 500$ replicas, regardless of the initial conditions of the experiment. This convergence to an equilibrium where the storage space ($N \cdot c$) is evenly divided between the number of items present (n) is a result of the repeated execution of the protocol.

In the shuffle protocol, there is no loss of information during the gossip exchange. We assume that the shuffle operation between two nodes is atomic and since nodes swap items, no item can possibly disappear once inserted into the network. For this reason, once the system has reached equilibrium in terms of replication of items and assuming that there is a path between any two nodes in the network, the replicas will continue to be shuffled, moving from one node to another in a random fashion. Under these conditions, it is reasonable to assume that the probability of finding any given item in a node's cache is the same for all items in the networks. To be more specific, based on our observations we make the assumption that items are uniformly distributed throughout the network and that any item can be found in a node's cache with probability $\frac{c}{n}$. This is the starting point for our probabilistic analysis. The observation of uniform distribution is not entirely unexpected. Similar observations about uniform distribution after repeated shuffling have been made regarding the shuffling of decks of cards in [7,3].

3 An Analytical Model of Information Dissemination

We analyse dissemination of a generic item d in a network in which the nodes execute the shuffling protocol.

3.1 Probabilities of state transitions

We present a model of the shuffle protocol that captures the presence or absence of a generic item d after shuffling of two nodes A and B . There are four possible states of the caches of A and B before the shuffle: both hold d , either A 's or B 's cache holds d , or neither cache holds d .

We use the notation $P(a_2b_2|a_1b_1)$ for the probability that from state a_1b_1 after a shuffle we get to state a_2b_2 , with $a_i, b_i \in \{0, 1\}$. The indices a_1, a_2 and b_1, b_2 indicate the presence (if equal to 1) or the absence (if equal to 0) of a generic item d in the cache of an initiator A and the contacted node B , respectively. For example, $P(01|10)$ means that node A had d before the shuffle, which then moved to the cache of B , afterwards. Due to the symmetry of information exchange between nodes A and B in the shuffle protocol,

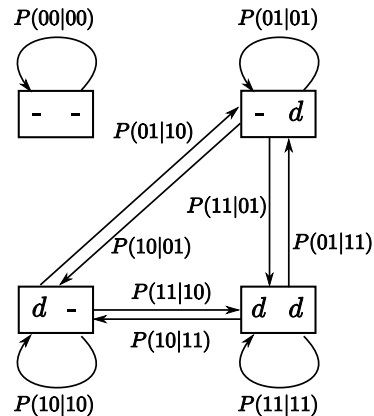


Fig. 2. Symbolic representation for caches of gossiping nodes.

$$P(a_2b_2|a_1b_1) = P(b_2a_2|b_1a_1).$$

Fig. 2 depicts all possible outcomes for the caches of gossiping nodes as a state transition diagram. If before the exchange A and B do not have d ($a_1b_1 = 00$), then clearly after the exchange A and B still do not have d ($a_2b_2 = 00$). Otherwise, if A or B has d ($a_1 = 1 \vee b_1 = 1$), the shuffle protocol guarantees that after the exchange A or B still has d ($a_2 = 1 \vee b_2 = 1$). Therefore, the state $(-, -)$ has a self-transition, and no other outgoing or incoming transitions.

We now determine values for all probabilities $P(a_2b_2|a_1b_1)$. They are expressed in terms of probabilities P_{select} and P_{drop} . The probability P_{select} expresses the chance of an item to be selected by a node from its local cache when engaged in an exchange. The probability P_{drop} represents a probability that an item which can be overwritten (meaning it is in the exchange buffer of its node, but not of the other node in the shuffle) is indeed overwritten by an item received by its node in the shuffle. Due to the symmetry of the protocol, these probabilities are the same for both initiating and contacted nodes. In Sec. 3.2, we will calculate P_{select} and P_{drop} . We write $P_{\neg select}$ for $1 - P_{select}$ and $P_{\neg drop}$ for $1 - P_{drop}$.

Scenario 1 ($a_1b_1 = 00$) Before shuffling, neither node A nor node B have d in their cache.

$a_2b_2 = 00$: neither node A nor node B have item d after a shuffle because neither of them had it in the caches before the shuffle: $P(00|00) = 1$

$a_2b_2 \in \{01, 10, 11\}$: cannot occur, because none of the nodes have item d .

Scenario 2 ($a_1b_1 = 01$) Before shuffling, a copy of d is only in the cache of node B .

$a_2b_2 = 01$: node A does not have d because node B had d but did not select it (to send) and, thus, B did not overwrite d , i.e. the probability is $P(01|01) = P_{\neg select}$

$a_2b_2 = 10$: only node A has d because node B selected d and dropped it; that is, the probability is $P(10|01) = P_{select} \cdot P_{drop}$

$a_2b_2 = 11$: both nodes A and B have a copy of d because node B selected d and kept it; that is, $P(11|01) = P_{select} \cdot P_{\neg drop}$

$a_2b_2 = 00$: cannot occur as completely discarding d is not possible in the protocol; that is, if either nodes send an item, its partner keeps this copy as well, and if an item is not among the selected for a shuffle, the item is not replaced by another one (see Sec. 2.2).

Scenario 3 ($a_1b_1 = 10$) Before shuffling, d is only in the cache of node A .

Due to the symmetry of nodes A and B , this scenario is symmetric to the previous one with $P(a_2b_2|10) = P(b_2a_2|01)$.

Scenario 4 ($a_1b_1 = 11$) Before shuffling, d is in the cache of node A as well as in the cache of node B .

$a_2b_2 = 01$: only node B has d because node A selected d and dropped it and node B did not select d ; that is, $P(01|11) = P_{select} \cdot P_{drop} \cdot P_{\neg select}$

$a_2b_2 = 10$: this outcome is symmetric to the previous one: $P(10|11) = P_{\neg select} \cdot P_{select} \cdot P_{drop}$

$a_2b_2 = 11$: after the shuffle both nodes A and B have d , because:

- * nodes A and B had d but both did not select it, i.e. $P_{\neg select} \cdot P_{\neg select}$;
- * both nodes A and B selected d (thus, both kept it), i.e. $P_{select} \cdot P_{select}$;
- * node A selected d and kept it and node B did not select d : $P_{select} \cdot P_{\neg drop} \cdot P_{\neg select}$;
- * symmetric case with the previous one: $P_{\neg select} \cdot P_{select} \cdot P_{\neg drop}$.

Thus, $P(11|11) = P_{\neg select} \cdot P_{\neg select} + P_{select} \cdot P_{select} + 2 \cdot P_{select} \cdot P_{\neg select} \cdot P_{\neg drop}$

$a_2b_2 = 00$: cannot occur, discarding of an item is not permitted by the protocol (see Sec. 2.2).

3.2 Probabilities of selecting and dropping an item

The following analysis assumes that all node caches are full (that is, the network is already running for a while). Moreover, we assume a uniform distribution of items over the network; this assumption is supported by experiments in [17,22].

Consider nodes A and B engaged in a shuffle, and let B receive the exchange buffer S_A from A . Let k be the number of duplicates (see Fig. 3), i.e. the items of an intersection of the node cache C_B and the exchange buffer of its gossiping partner S_A (i.e. $S_A \cap C_B$). Recall from Sec. 2.1 that C_A and C_B contain the same number of items for all A and B , and likewise for S_A and S_B ; we use c and s for these values. The total number of different items in the network is denoted as n .

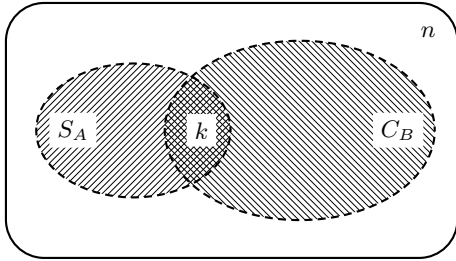


Fig. 3. k items in $S_A \cap C_B$

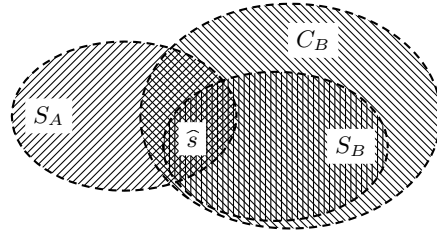


Fig. 4. \hat{s} items in $S_A \cap S_B$

The probability of selecting an item d in the cache is the number of selected items (i.e. s) divided by the total number of items in the cache (i.e. c): $P_{select} = \frac{s}{c}$. Thus, the probability that an item d in the cache is not selected is: $P_{\neg select} =$

$$1 - P_{select} = \frac{c-s}{c}.$$

Consider Figs. 3 and 4. The shuffle protocol demands that all items in S_A are kept in C_B after the shuffle. This implies that: a) all items in $S_A \setminus C_B$ will overwrite items in $S_B \subseteq C_B$, and b) all items in $S_A \cap C_B$ are kept in C_B . Thus, the probability that an item from S_B will be overwritten is determined by the probability that an item from S_A is in C_B , but not in S_B . Namely, the items in $S_B \setminus S_A$ provide a space in the cache for items from $S_A \setminus C_B$. We would like to express the probability P_{drop} of a selected item d in $S_B \setminus S_A$ (or $S_A \setminus S_B$) to be overwritten by another item in C_B (or C_A). Due to symmetry, this probability is the same for A and B ; therefore, we calculate only the probability that an item in $S_B \setminus S_A$ is dropped from C_B . The expected value of this probability depends on how many duplicates a node receives from its gossiping partner²:

$$E[P_{drop}] = \begin{cases} \sum_{k=0}^s (P_{drop}^{|S_A \cap C_B|=k} \cdot P_{|S_A \cap C_B|=k}) & \text{if } s + c \leq n \\ \sum_{k=(s+c)-n}^s (P_{drop}^{|S_A \cap C_B|=k} \cdot P_{|S_A \cap C_B|=k}) & \text{otherwise} \end{cases}$$

where $P_{|S_A \cap C_B|=k}$ is the probability of having exactly k items in $S_A \cap C_B$, and $P_{drop}^{|S_A \cap C_B|=k}$ is the probability that an item in $S_B \setminus S_A$ is dropped from C_B given k duplicates in $S_A \cap C_B$. The case distinction is because if $s + c > n$, then clearly there are at least $(s + c) - n$ items in $S_A \cap C_B$.

From the $\binom{n}{s}$ possible sets S_A , we compute how many have k items in common with C_B . Firstly, there are $\binom{c}{k}$ ways to choose k such items in C_B . Secondly, there are $\binom{n-c}{s-k}$ ways to choose the remaining $s-k$ items outside C_B . So in total, $\binom{c}{k} \cdot \binom{n-c}{s-k}$ possible sets S_A have k items in common with C_B . Hence, under the assumption of a uniform distribution of the data items over the caches of the nodes,³ $P_{|S_A \cap C_B|=k} = \binom{c}{k} \frac{\binom{n-c}{s-k}}{\binom{n}{s}}$. The expected value of $P_{drop}^{|S_A \cap C_B|=k}$ is⁴:

$$E[P_{drop}^{|S_A \cap C_B|=k}] = \begin{cases} \sum_{\hat{s}=0}^k P_{drop}^{|S_A \cap S_B|=\hat{s}} \cdot P_{|S_A \cap S_B|=\hat{s}} & \text{if } s + k \leq c \\ \sum_{\hat{s}=(s+k)-c}^k P_{drop}^{|S_A \cap S_B|=\hat{s}} \cdot P_{|S_A \cap S_B|=\hat{s}} & \text{otherwise} \end{cases}$$

where \hat{s} is the number of items in $S_A \cap S_B$ (see Fig. 4). The case distinction is because if $s + k > c$ (with k the number of items in $S_A \cap C_B$), then clearly

² The other case is presented for the sake of completeness.

³ Here we use a generalisation of the usual definition of binomial coefficients to negative integers. That is, for all m and $l \geq 0$, $\binom{m}{l} = (-1)^l \binom{-m+l-1}{l}$ (cf. [20])

⁴ The other case is presented for the sake of completeness.

there are at least $(s + k) - c$ items in $S_A \cap S_B$.

Among the s items in S_B , there are \hat{s} items also in S_A , and thus only the $s - \hat{s}$ items in $S_B \setminus S_A$ can be dropped from C_B . $P_{drop}^{|S_A \cap S_B|=\hat{s}}$ is the probability that an item in $S_B \setminus S_A$ is dropped from C_B , given \hat{s} items in $S_A \cap S_B$:

$$P_{drop}^{|S_A \cap S_B|=\hat{s}} = \begin{cases} 0 & \text{if } s = \hat{s} \\ \frac{s-k}{s-\hat{s}} & \text{otherwise} \end{cases}$$

$P_{|S_A \cap S_B|=\hat{s}}$ is the probability of having exactly \hat{s} items in $S_A \cap S_B$: $P_{|S_A \cap S_B|=\hat{s}} = \binom{s}{\hat{s}} \frac{\binom{c-s}{k-\hat{s}}}{\binom{c}{k}}$. The intuition behind this expected value of $P_{|S_A \cap S_B|=\hat{s}}$ is similar to the one of $P_{|S_A \cap C_B|=k}$. From the $\binom{c}{k}$ possible sets S_A , we compute how many have \hat{s} items in common with S_B . That is, there are $\binom{s}{\hat{s}}$ ways to choose \hat{s} items in S_B , and $\binom{c-s}{k-\hat{s}}$ ways to choose the remaining $k - \hat{s}$ items outside S_B .

Let us assume $s + c \leq n$ and $s + k \leq c$. Then, substituting in the expression for $E[P_{drop}]$ in case $s + c \leq n$, and noting that in the summand $k = s$ the factor $P_{drop}^{|S_A \cap S_B|=s}$ is equal to zero, we get:

$$\begin{aligned} E[P_{drop}] &= \sum_{k=0}^{s-1} \binom{c}{k} \frac{\binom{n-c}{s-k}}{\binom{n}{s}} \sum_{\hat{s}=0}^k \frac{s-k}{s-\hat{s}} \binom{s}{\hat{s}} \frac{\binom{c-s}{k-\hat{s}}}{\binom{c}{k}} \\ &= \frac{n-c}{\binom{n}{s}} \sum_{k=0}^{s-1} \binom{(n-c)-1}{(s-k)-1} \sum_{\hat{s}=0}^k \frac{\binom{c-s}{k-\hat{s}} \binom{s}{\hat{s}}}{s-\hat{s}} \end{aligned} \quad (1)$$

The probability of keeping an item d in $S_B \setminus S_A \subseteq C_B$ can be expressed as $P_{-drop} = 1 - P_{drop}$.

3.3 Simplification of P_{drop}

In order to gain a clearer insight into the emergent behaviour of the gossiping protocol we make an effort to simplify the formula for the probability P_{drop} of an item in $S_B \setminus S_A$ to be dropped from C_B after a shuffle. Therefore, we re-examine the relationships between the k duplicates received from a neighbour, the \hat{s} items of the overlap $S_A \cap S_B$, and P_{drop} . Let's estimate $P_{drop}^{|S_A \cap C_B|=k}$ by considering each item from S_A separately, and calculating the probability that the item is a duplicate (i.e., is also in C_B). The probability of an item from S_A to be a duplicate (also present in C_B) is $\frac{c}{n}$. In view of the uniform distribution of items over the network, the items in a node's cache are a random sample from the universe of n data items; so all items in S_A have the same chance

to be a duplicate. Thus, the expected number k of items in $S_A \cap C_B$ can be estimated by $s \cdot \frac{c}{n}$. The expected number \hat{s} of items in $S_A \cap S_B$ can be estimated by $k \cdot \frac{s}{c}$, because only the k items in $S_A \cap C_B$ may end up in $S_A \cap S_B$; $\frac{s}{c}$ captures the probability that an item from C_B is also selected to be in S_B . It follows that the probability of an item in $S_B \setminus S_A$ to be dropped from C_B after the shuffle is $P_{drop} = \frac{s-k}{s-s} = \frac{s-s \cdot \frac{c}{n}}{s-s \cdot \frac{c}{n} \cdot \frac{s}{c}} = \frac{n-c}{n-s}$. The complementary probability of keeping an item is $P_{-drop} = 1 - \frac{n-c}{n-s} = \frac{c-s}{n-s}$. These estimates are valid for general $s \leq c \leq n$.

Substituting the expressions for P_{select} and the simplified P_{drop} into the formulas for the transition probabilities in Fig. 2, we obtain:

$$\begin{aligned} P(01|01) &= P(10|10) = \frac{c-s}{c} & P(01|11) &= P(10|11) = \frac{s}{c} \frac{c-s}{c} \frac{n-c}{n-s} \\ P(10|01) &= P(01|10) = \frac{s}{c} \frac{n-c}{n-s} & P(11|11) &= 1 - 2 \frac{s}{c} \frac{c-s}{c} \frac{n-c}{n-s} \\ P(11|01) &= P(11|10) = \frac{s}{c} \frac{c-s}{n-s} \end{aligned}$$

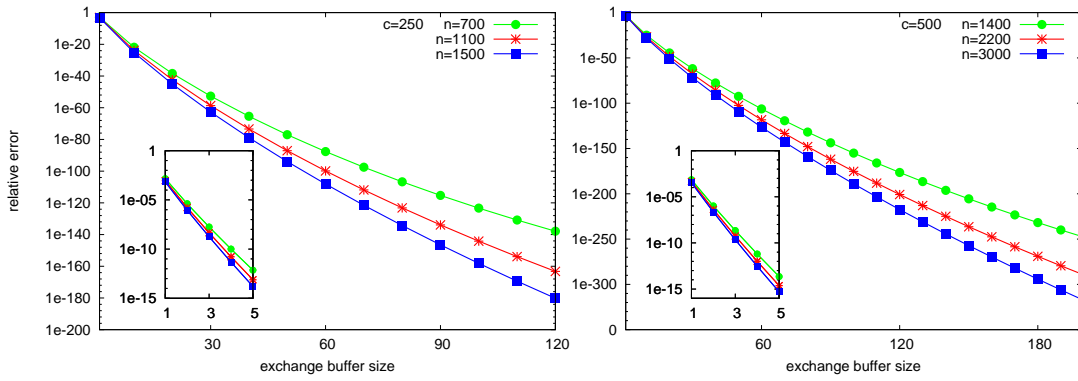


Fig. 5. The relative error of the difference of the accurate P_{drop} and its approximation, for $c = 250$ (left) and $c = 500$ (right) and different values of n .

In order to verify the accuracy of the proposed simplification for $E[P_{drop}]$, we compare the simplification and the accurate formula (1) for different values of n . We plot the difference of the accurate P_{drop} and the simplification, for cache sizes $c = 250$ and $c = 500$ (Fig. 5). These figures show that the simplification gives a close approximation of the accurate formula for P_{drop} (note the log y-scale).

3.4 Correction factor

We now investigate how closely the simplified formula of P_{drop} , that is $\frac{n-c}{n-s}$ (here referred to as $S(n, c, s)$) approximates formula (1) (here referred to as

$E(n, c, s)$). We compared the difference between these two formulas using a Java package based on common fractions, which provides loss-less calculation [18]. We observe that the inverse of the difference of the inverse values of both formulas, i.e. $e_{c,s}(n) = (E(n, c, s)^{-1} - S(n, c, s)^{-1})^{-1}$, exhibits a certain pattern for different values of n , c and s .

For $s = 1$ and arbitrary values of n and c , $E(n, c, 1) = \frac{n-c}{n}$, whereas $S(n, c, 1) = \frac{n-c}{n-1}$. This leads us to investigate the correction factor θ as in $E(n, c, s) = \frac{n-c}{(n-s)+\theta}$. For $s = 1$ clearly the factor $\theta = 1$. Yet, for $s > 1$ the situation is more complicated. We therefore calculate the first, the second and other (forward) differences⁵ over n .

For $s = 1$, the result of the first difference of the function $e_{c,1}(n)$ is 1. However, for $s = 2$ the first difference is, e.g. if $c = 4$: $e_{4,2}(7) - e_{4,2}(6) = 3.5$, $e_{4,2}(8) - e_{4,2}(7) = 4$, $e_{4,2}(9) - e_{4,2}(8) = 4.5$ and so on. Thus, we observe that the second difference of $e_{c,2}(n)$ is $\frac{1}{2}$. By calculating higher differences for $s > 2$, we conclude that the s -th difference of the function $e_{c,s}(n)$ is always $\frac{1}{s}$.

Moreover, at the point $n = 0$ the first, \dots , s -th differences of the function $e_{c,s}$ exhibit a pattern similar to the Pascal triangle [28]. That is, for $d \geq 1$ the d -th difference is $(\Delta^d e_{c,s})(0) = \frac{1}{s \cdot \binom{s-1}{d}}$ (assuming $\binom{a}{b} = 0$, whenever $b > a$). Knowing the initial difference at point $n = 0$, we were able to use the Newton forward difference equation [1] to derive the following formula for $n > 0$: $E[P_{drop}] = \frac{n-c}{(n-s)+\frac{1}{\gamma}}$, where

$$\gamma = \sum_{d=0}^{s-1} \frac{\binom{n}{d}}{s \cdot \binom{s-1}{d}} = \frac{\binom{n}{s}}{(n-s)+1} \cdot \sum_{d=0}^{s-1} \frac{1}{\binom{n-d}{(s-1)-d}} \quad (2)$$

In this equation the sum is finite because due to the observation that the s -th difference is constant $\frac{1}{s}$, all higher differences are 0.

Thus, the correction factor is $\theta = \frac{1}{\gamma}$. Extensive experiments with Mathematica and Matlab indicate that $\frac{n-c}{(n-s)+\frac{1}{\gamma}}$ and formula (1) indeed coincide. We can also see from Fig. 5 that the correction factor is small.

3.5 Optimal size for the exchange buffer

We study the optimal value for fast convergence of replication and coverage with respect to an item d . Since d is introduced at only one node in the

⁵ A forward difference of discrete function $f : \mathbb{Z} \rightarrow \mathbb{Z}$ is a function $\Delta f : \mathbb{Z} \rightarrow \mathbb{Z}$ with $\Delta f(n) = f(n+1) - f(n)$ (cf. [1]).

network, one needs to maximise the chance that an item is duplicated. That is, the probabilities $P(11|01)$ and $P(11|10)$ should be maximised (then $P(01|11)$ and $P(10|11)$ are maximised as well, intuitively because for each duplicated item in a shuffle, another item must be dropped).

We give a rigorous argument for this claim in the case of a fully connected network. Let α be a replication of the distinguished item at a given moment, and $0 \leq \alpha \leq 1$. In the long run, α converges to $\frac{c}{n}$. Suppose that a shuffle takes place. The chance that after the shuffle there is one more copy of the distinguished item in the system is

$$(P(11|01) + P(11|10)) \cdot \alpha \cdot (1 - \alpha) = 2 \cdot P(11|01) \cdot \alpha \cdot (1 - \alpha)$$

Here $\alpha \cdot (1 - \alpha)$ expresses the chance that exactly one of the nodes in the shuffle contains a copy of the item. Likewise, the chance that after the shuffle a copy of the item has been removed from the system is

$$(P(01|11) + P(10|11)) \cdot \alpha^2 = 2 \cdot P(01|11) \cdot \alpha^2 = 2 \cdot \frac{n - c}{c} \cdot P(11|01) \cdot \alpha^2$$

So after a shuffle, the change in the number of copies of the item in the system is on average

$$2 \cdot P(11|01) \cdot \alpha \cdot (1 - \alpha) - 2 \cdot \frac{n - c}{c} \cdot P(11|01) \cdot \alpha^2 = 2 \cdot P(11|01) \cdot \alpha \cdot \left(1 - \frac{n}{c} \cdot \alpha\right)$$

So as long as $\alpha < \frac{c}{n}$, maximising $P(11|01)$ maximises replication of the item.

These probabilities both equal $\frac{s}{c} \frac{c-s}{n-s}$; we compute when the s -derivative of this formula is zero. This yields the equation $s^2 - 2ns + nc = 0$; taking into the account that $s \leq n$, the only solution of this equation is $s = n - \sqrt{n(n - c)}$. We conclude that this is the optimal value for s to obtain fast convergence of replication (see Fig. 6). This will also be confirmed by the experiments and analyses in the following sections.

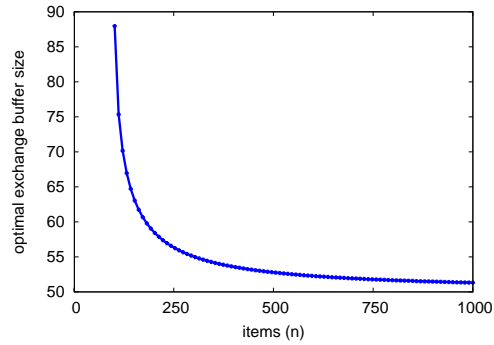


Fig. 6. Optimal value of exchange buffer size, depending on n .

4 Experimental Evaluation

In order to test the validity of the analytical model of information spread based on the shuffle protocol presented in the previous section, we follow an

experimental approach. We compare properties observed while running the shuffle protocol in a large-scale deployment with simulations of the model under the same conditions. These experiments show that the analytical model indeed captures information spread of the shuffle protocol. We note that a simulation of the analytical model is much more efficient (in computation time and memory consumption) than a simulation of the implementation of the shuffle protocol. We discuss this further in Sec. 6.

The experiments simulate the case where a new item d is introduced at one node in a network, in which all caches are full and uniformly populated by $n = 500$ items. We use an event-based simulator that takes as input the topology of the network to determine which pairs of nodes can gossip. The experiments are performed on a network of $N = 2500$ nodes, arranged in a square grid topology (50×50), where each node can communicate only with its four immediate neighbours (to the North, South, East and West). This configuration of nodes is arbitrary, except for the fact that we require a large number of nodes for the observation of emergent behaviour. Our aim is to validate the correctness of our analytical model, not to test the endless possibilities of network configurations. The model and the shuffle protocol do not make any assumptions about the network. The network configuration is provided by the simulation environment and can easily be changed into something different, e.g. another network topology. For this reason, we have chosen this large grid for testing, although other configurations could have been possible. In the experiments that follow, after each gossiping round, we measure the total number of occurrences of d in the network (replication), and how many nodes in total have seen d (coverage).

Simulations with the shuffle protocol Each node in the network has a cache size of $c = 100$, and sends s items when gossiping. In each round, every node randomly selects one of its neighbours and shuffles. In order to make a fair comparison with the simulations with the model, we let the nodes gossip for 1000 rounds before initiating the measurements of the properties. After this start-up period of 1000 rounds, items are replicated and the replicas fill the caches of all nodes fulfilling the uniform distribution requirement of the model. At round 1000, item d is inserted into the network at a random location. From that moment on we track its replication and coverage.

Simulations with the model For the simulations with the model, n , c and s are only system parameters. Instead of maintaining a cache, each node in the network only maintains a variable that represents whether it holds item d or not (state 1 or 0, respectively). Nodes update their state in pairs according to the transition probabilities introduced before (Fig. 2). This mimics an actual exchange of items between a pair of nodes according to the shuffle protocol.

While in the protocol this results in both nodes updating the contents of their caches, in a simulation using the analytical model updating the state of a node refers to updating only one variable: whether the node is in possession of the item d or not. To sum up, we use transition probabilities to update the state of one variable. Since we do not need a start-up time for the simulations with the model, at round 0 we set the state of a random node to 1 (while all the other have state 0) and track the state of the nodes for the remainder of the simulation.

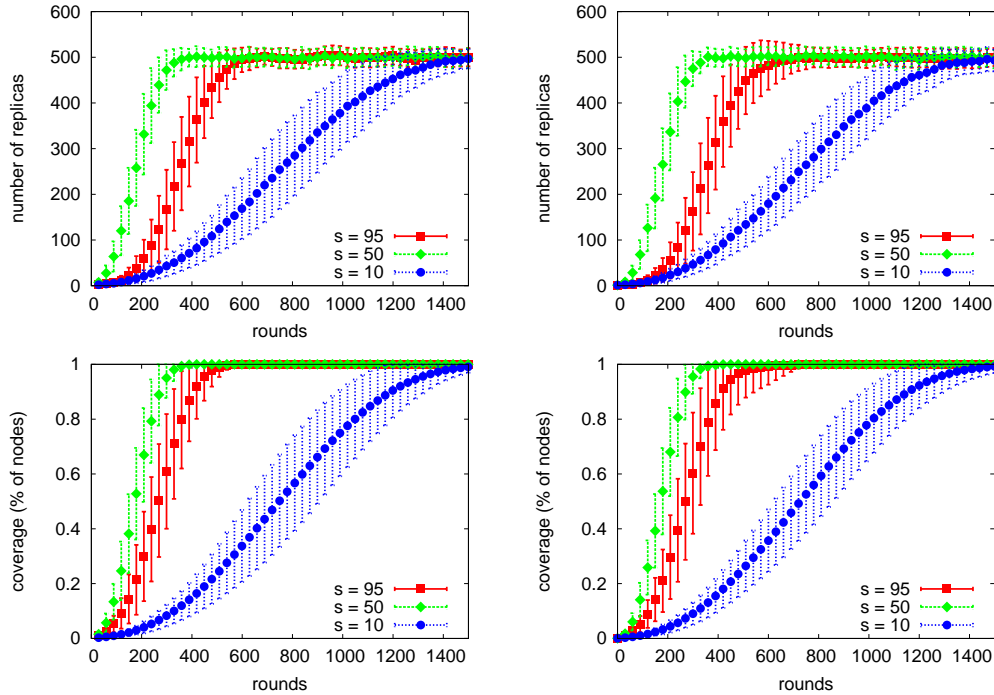


Fig. 7. The shuffle protocol (left) and the model (right), for $N = 2500$, $n = 500$, $c = 100$ and different values of s .

Fig. 7 shows the behaviour of both the shuffle protocol and the analytical model in terms of replication (upper row of Fig. 7) and coverage (lower row of Fig. 7) of d , for various values of s . Each curve in the graphs represents the average and standard deviation calculated over 100 runs. The experiments with the model calculate P_{drop} using the simplified formula $\frac{n-c}{n-s}$ described in Sec. 3.3. It can be observed very clearly that the results obtained from the model (right) resemble closely the ones from executing the protocol (left).

We note that in all cases, the network converges to a situation in which there are 500 copies of d , meaning that replication is $\frac{500}{2500} = 0.2$; this agrees with the fact that $\frac{c}{n} = \frac{100}{500} = 0.2$. Moreover, our experiments show that replication and coverage display the fastest convergence when $s = 50$; this agrees with the fact that $n - \sqrt{n(n-c)} = 500 - \sqrt{500 \cdot 400} \approx 50$ (cf. Sec. 3.5).

5 Modelling of Protocol Properties with Differential Equations

In this section, we exploit the analytical model of information dissemination to perform a mathematical analysis of replication and coverage with regard to the shuffle protocol. For the particular case of a network with full connectivity, we can find explicit expressions for the dissemination of a generic item d in terms of the probabilities presented in Sec. 3. We construct two differential equations that capture replication and coverage of item d from a round-based perspective. The advantage of this approach is that we can determine the long-term behaviour of the system as a function of the parameters. Differential equations have been previously used, for example, in [27,19,25] to model probabilistic protocols for large-scale distributed systems.

The characteristics of the replication and coverage are influenced by the topology of the network. It is the topology of the network that will dictate the likelihood of any two nodes to gossip. In order to model the properties of the protocol, it is crucial that we know the probabilities of a node in a given state (0 or 1) to interact with another node in a given state. For this reason, we choose to model the properties for a fully-connected network, where a node can gossip with any other node in the network. This topology allows us to easily calculate the probability of a node in state 1 to interact with a node in state 0 or to interact with another node in state 1. Knowing this, we concentrate on applying the state transition probabilities that we calculated in Section 3. The aim of this section is to provide an example of how the transition probabilities can be used to model the properties of dissemination for a specific topology.

5.1 Replication

One node introduces a new item d into the network at time $t = 0$, by placing d into its cache. From that moment on, d is replicated as a consequence of gossiping among nodes.

Let $x(t)$ represent the percentage of nodes in the network that have d in their cache at time t , where each gossip round takes one time unit. The variation in x per time unit $\frac{dx}{dt}$ can be derived based on the probability that an item d will replicate or disappear after an exchange between two nodes, where at least one of the nodes has d in its cache:

$$\frac{dx}{dt} = [P(11|10) + P(11|01)] \cdot (1 - x) \cdot x - [P(10|11) + P(01|11)] \cdot x \cdot x$$

The first term represents duplication of d when a node that has d in its cache initiates the shuffle, and contacts a node that does not have the item. The second term represents the opposite situation, when a node that does not

have the item d initiates a shuffle with a node that has d . The third and fourth term in the equation represent the cases where both gossiping nodes have d in their cache, and after the exchange only one copy of d remains. Substituting $P(11|10) = P(11|01) = \frac{s}{c} \frac{c-s}{n-s}$ and $P(10|11) = P(01|11) = \frac{s}{c} \frac{n-c}{n-s} \frac{c-s}{c}$, we obtain

$$\frac{dx}{dt} = 2 \cdot \frac{s}{c} \cdot \frac{c-s}{n-s} \cdot x \cdot \left(1 - \frac{n}{c} \cdot x\right)$$

The solution of this equation, taking into account that $x(0) = \frac{1}{N}$ with N the number of nodes in the network, is

$$x(t) = \frac{e^{\alpha t}}{\left(N - \frac{n}{c}\right) + \frac{n}{c} e^{\alpha t}} \quad (3)$$

where α denotes $2 \frac{s}{c} \frac{c-s}{n-s}$. By imposing stationarity, i.e. $\frac{dx}{dt} = 0$, we find the stationary solution $\frac{c}{n}$. Hence, this calculation confirms the observation in Sec. 2.3 that the network converges to a situation in which replication of d is $\frac{c}{n}$.

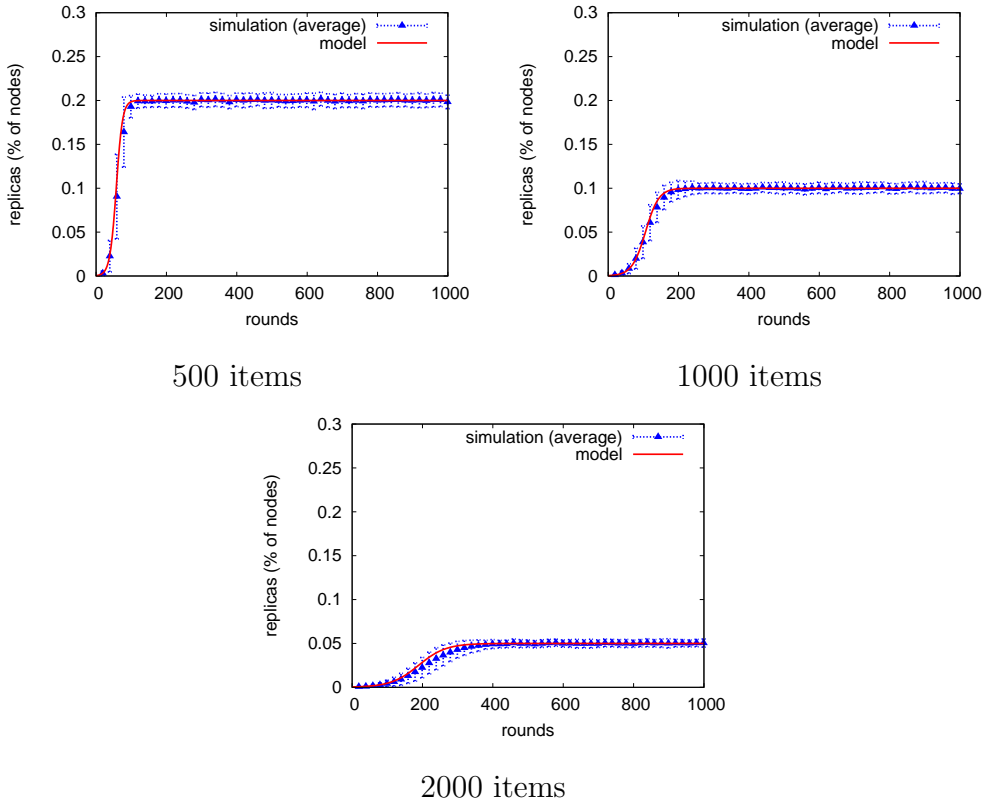


Fig. 8. Percentage of nodes in the network with a replica of item d in their cache, for $N = 2500$, $c = 100$, $s = 50$, and $n = 500$, $n = 1000$ or $n = 2000$.

We evaluate the accuracy of $x(t)$ as a representation of the fraction of nodes carrying a replica of d , by running a series of experiments where $N = 2500$ nodes execute the shuffle protocol, and their caches are monitored for the

presence of d . Unlike the experiments in Sec. 4, we assume full connectivity; that is, for each node, all other nodes are within reach. After 1000 rounds, where items are disseminated and replicated, a new item d is inserted at a random node, at time $t = 0$. We track the number of replicas of d for the next 1000 rounds. The experiment is repeated 100 times and the results are averaged. These simulation results (average and standard deviation) for the protocol, together with $x(t)$, are presented in Fig. 8. This figure shows the same initial increase in replicas after d has been inserted, and in all cases the steady state reaches precisely the expected value $\frac{c}{n}$ predicted from the stationary solution.

We repeat the calculation from Sec. 3.5, but now against $x(t)$, to determine which size of the exchange buffer yields the fastest convergence to the steady-state for both replication and coverage. That is, we search for the s that maximises the value of $x(t)$. We first compute the derivative of $x(t)$ with respect to s ($z(t, s)$), and then derive the value of s that maximises $x(t)$, by taking $z(\cdot, m) = \frac{\partial x}{\partial s}|_m = 0$: $z(t, s) = \frac{\partial x}{\partial s} = \frac{2e^{kt}(cN-n)(cn+s(-2n+s))t}{(cN+(-1+e^{kt})n)^2(n-s)^2}$, where $k = 2\frac{s}{c}\frac{c-s}{n-s}$. Let $z(t, s) = 0$. For $t > 0$, $cn = s(2n - s)$. Solving this equation we get $s = n \pm \sqrt{n(n - c)}$. Taking into the account that $s \leq n$, the only solution is $s = n - \sqrt{n(n - c)}$. This also coincides with the optimal exchange buffer size found in Sec. 3.5.

5.2 Coverage

We use the term coverage to denote the percentage of nodes in the network that have seen item d from the moment it was introduced into the network. Let $y(t)$ represent the coverage of d at time t . The variation in coverage per time unit, $\frac{dy}{dt}$, is determined by the fraction of nodes that have not seen d , $1 - y$, that interacts with nodes that have d in their cache, x . Let $* \in \{0, 1\}$, then:

$$\frac{dy}{dt} = \frac{1}{2} \cdot (P(1*|01) \cdot P(*1|*1) \cdot (1 - y) \cdot x + P(*1|10) \cdot P(1*|1*) \cdot x \cdot (1 - y)) \quad (4)$$

The first term represents increased coverage due to nodes discovering d after interacting with nodes that have d in their cache. This can occur when a node initiates the exchange ($P(1*|01)$), or when the node is contacted ($P(*1|10)$). Each of these scenarios has a 50% chance of occurring. The second part of these terms represents the case when a node discovers and does not give away its copy of d within the same round to another node. This is because coverage is measured only at the end of a gossiping round, meaning that a node that sees item d for the first time, and drops it in the same round, is considered not to

have seen item d yet.⁶ Since nodes shuffle, on average, twice per round (once when they initiate the shuffle and again if they are contacted by a neighbour), this could occur under two scenarios: i) the node acquired d by initiating an exchange with a node that had d ($P(1*|01)$) and next lost its copy of d when shuffling with a node that contacted it, or ii) the node was first contacted by a node that sent a copy of d ($P(*1|10)$) and next initiated a shuffle and gave away its copy of d .

The value of the probability $P(*1|*1)$ can be calculated from the following cases: a) the gossip partner of the node does not have d , and: i) the state of two nodes does not change after the gossip ($P(01|01) \cdot (1 - x)$), and ii) the gossip partner obtains a copy of d after the gossip ($P(11|01) \cdot (1 - x)$); and b) the gossip partner of the node has d , and: i) two nodes have the same state after the exchange ($P(11|11) \cdot x$), and ii) the gossip partner loses its copy of d after the gossip ($P(01|11) \cdot x$). Hence,

$$\begin{aligned} P(*1|*1) &= (P(01|01) + P(11|01)) \cdot (1 - x) + (P(11|11) + P(01|11)) \cdot x \\ &= P_{\neg select} + P_{select} \cdot (P_{\neg drop} \cdot (1 - x) + P_{select} \cdot x + P_{\neg drop} \cdot P_{\neg select}) \end{aligned}$$

Due to the symmetry of both gossiping nodes, $P(*1|*1) = P(1*|1*)$.

Substituting these probabilities into (4), we obtain

$$\frac{dy}{dt} = \frac{s}{c} \cdot \left(1 - \frac{s}{c} + \frac{s}{c} \cdot \frac{c-s}{n-s} \left(2 - \frac{s}{c} \right) + \left(\frac{s}{c} - \frac{c-s}{n-s} \right) \cdot x \right) \cdot (1 - y) \cdot x$$

The solution of this equation, taking into account that $y(0) = \frac{1}{N}$, is

$$y(t) = 1 - (N - 1) \cdot N^{\beta-1} \left(\left(N - \frac{n}{c} \right) + \frac{n}{c} \cdot e^{\alpha t} \right)^{-\beta} \cdot e^{-\lambda}$$

where λ denotes $\frac{\frac{s}{c} \cdot \left(\frac{s}{c} - \frac{c-s}{n-s} \right)}{\alpha \cdot \frac{n}{c}} \left(\frac{1}{N} - \frac{e^{\alpha t}}{\left(N - \frac{n}{c} \right) + \frac{n}{c} e^{\alpha t}} \right)$, and β denotes $\frac{\frac{n}{c} \cdot \kappa + \frac{s}{c} \cdot \left(\frac{s}{c} - \frac{c-s}{n-s} \right)}{\alpha \cdot \left(\frac{n}{c} \right)^2}$, wherein κ is $\frac{s}{c} \cdot \left(1 - \frac{s}{c} + \frac{s}{c} \cdot \frac{c-s}{n-s} \left(2 - \frac{s}{c} \right) \right)$. By imposing stationarity $\frac{dy}{dt} = 0$, we find the stationary solution 1, meaning that eventually all nodes will see d . In order to evaluate how closely $y(t)$ models coverage, we use the traces from the simulations executed for Sec. 5.1. At every round, the nodes that carry a replica of d are identified, and a record of the nodes that have seen d since it was published is kept. Fig. 9 presents the coverage measured for three sets of experiments, each set with a different value for n . As n increases, a newly inserted item requires more time to cover the whole network. This is due to having more competition from other items to create replicas in the limited

⁶ The reason for this is that the application has an opportunity to read from the lower-level cache only once every round.

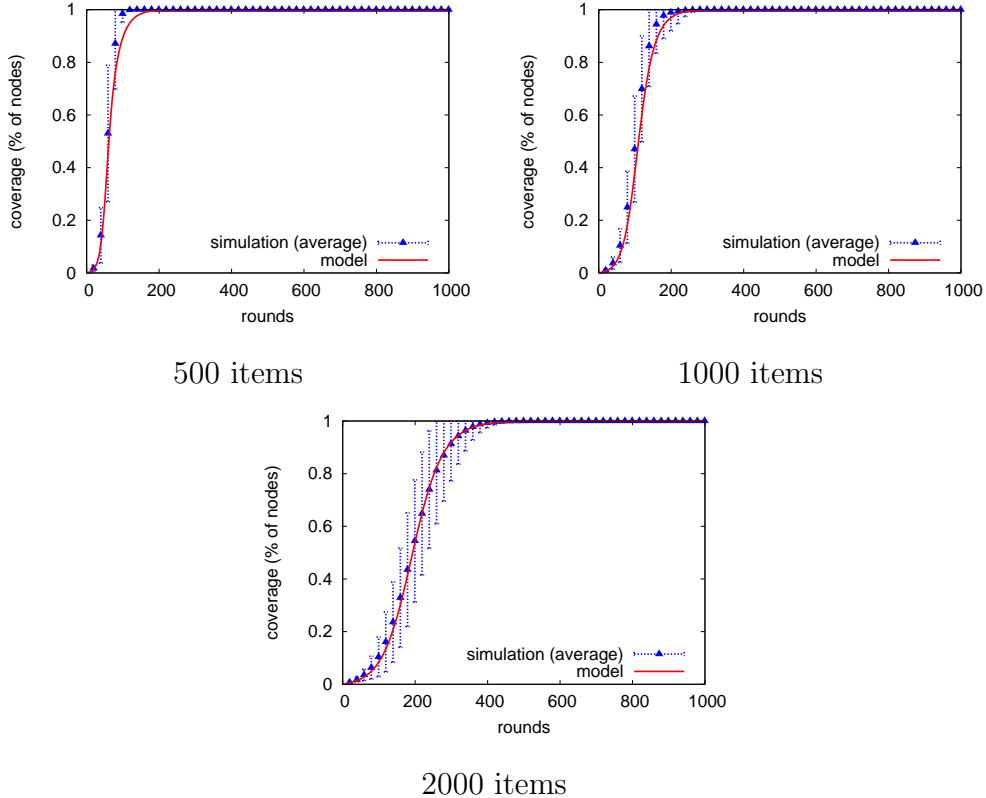


Fig. 9. Percentage of nodes in the network that have already seen a replica of item d , for $N = 2500$, $c = 100$, $s = 50$, and $n = 500$, $n = 1000$ or $n = 2000$.

space available, as was previously shown in Fig. 8. However, as predicted by the stationary solution, in all cases the coverage eventually reaches 1. As shown in Fig. 9, the solution $y(t)$ models the behaviour observed in simulations in case of $n = 2000$ and $n = 1000$, falling nicely within the standard deviation of the simulation results. However, for smaller n (i.e. $n = 500$), the solution $y(t)$ becomes less accurate, converging slower than the simulation results. In Appendix A we describe more accurate version of the coverage model for arbitrary number of contacts between nodes, which is rather complicated for understanding of the emergent behaviour of system.

6 Time Complexity of Experiments

In this section we discuss some topics/issues that arose during the development and testing of the model for the shuffle protocol.

During the experimental phase of this work, we observed a remarkable disparity between the time required to run an experiment with the shuffle protocol or with the model. In both cases, the experiment was the same in terms of

properties being measured and parameters used (cache size c , exchange buffer size s , number of different items n and network size N).

The difference in execution times can be traced back to the two different algorithms executed at each simulated node. From a node's point of view, the shuffle protocol requires the selection of items to send to the gossiping partner and the selection of items to keep for the next round. The first operation can be done in linear time, $O(c)$. The second operation requires checking if the incoming items are already in the cache and removing entries from the cache if space is needed. These steps can be done in $O(c \cdot \log c)$ time. With the second operation dominating the execution time, we can estimate the time complexity for one round of the shuffle to be $O(c \cdot \log c)$.

Now, let us look at the time complexity of the model. Unlike the protocol implementation, the model requires very little state, namely the value of the parameters and a variable to indicate the presence or absence of item d . At each round, each simulated node and its gossiping partner only have to determine their current state (a_1, b_1) and transition to a new state (a_2, b_2) according to the appropriate transition probabilities. The transition probabilities themselves do not change during the simulation (since they depend solely on the parameters c , s and n), so they are precomputed at the initialisation phase. As a result, the execution of the model for each node has a constant time complexity, $O(1)$.

The defining factor in the execution time of the simulations with the shuffle protocol is the size of the cache. With a cache size of 100 for all of our experiments, the execution times for our simulations with the shuffle protocol and the model differed by approximately two orders of magnitude. Considering that large networks are needed to clearly observe emergent behaviour and that this behaviour evolves over many rounds, the value of having a model becomes evident. Being simply parameters in the model, the cache size and exchange buffer size have no effect on the memory requirements and execution time of the simulation, thus freeing computational resources to experiment with larger networks and different topologies.

7 Conclusions

In this paper, we have demonstrated that it is possible to model a gossip protocol through a rigorous probabilistic analysis of the state transitions of a pair of nodes engaged in the gossip. We have shown, through an extensive simulation study, that the dissemination of a data item can be faithfully reproduced by the model. Having an accurate model of node interactions, we have been able to carry out the following:

- After finding precise expressions for the probabilities involved in the model, we provide a simplified version of the transition probabilities. These simplified, yet accurate, expressions can be easily computed, allowing us to simulate the dissemination of an item without the complexity of executing the actual shuffle protocol. These simulations use very little state (only some parameters and variables, as opposed to maintaining a cache) and can be executed in a fraction of the time required to run the protocol.
- The model reveals relationships between the parameters of the system. Armed with this knowledge, we successfully optimised one of the parameters (the size of the exchange buffer) to obtain the fastest convergence of the observed properties.
- Under the assumption of full connectivity, we are able to use the transition probabilities to model the properties of the dissemination of a generic item. Each property is ultimately expressed as a formula which is shown to display the same behaviour as the average behaviour of the protocol, verifying the validity of the model. We are working to extend our results with equational models, assuming various network topologies. We plan to publish these results in a future work.

While gossip protocols are easy to understand, even for a simple push/pull protocol, the interactions between nodes are unexpectedly complex. Understanding these interactions provides insight into the mechanics behind the emergent behaviour observed in gossip protocols. We believe that understanding the mechanics of gossiping is the key to optimising (and even shaping) the emergent properties that make gossiping appealing as communication paradigm for distributed systems.

Future work

In this paper, we consider a simple push-pull gossip protocol that was originally proposed for wireless environments. After successfully analysing this protocol at an abstract level (pairwise node interactions) independent of network topology or the characteristics of the environment itself, we are interested in the following possibilities for future study:

- We would like to investigate how our transition probabilities can be applied to model the properties of the protocol for other network topologies. As a first step, we have analysed the modelling of properties for a fully-connected network. We are interested in exploring the effect of more complex topologies in the modelling of the properties.
- In order to have a more realistic representation of a wireless setting, we would like to incorporate the presence of lossy links in our network. The characteristics of the lossy links would model an underlying MAC layer. In

other words, we would like to relax the assumption of atomic transactions for the shuffle protocol. This would result in new possible transitions in the state transition diagram (Fig. 2) due to loss of messages.

Acknowledgements

We thank Sandjai Bhulai and the anonymous reviewers for their advice and insightful comments.

References

- [1] M. Abramowitz, I. A. Stegun, Handbook of Mathematical Functions with Formulas, Graphs, and Mathematical Tables, 9th ed., Dover, New York, 1972.
- [2] A. Allavena, A. Demers, J. E. Hopcroft, Correctness of a gossip based membership protocol, in: Proc. PODC'05, ACM, 2005, pp. 292–301.
- [3] S. Assaf, P. Diaconis, K. Soundararajan., A rule of thumb for riffle shuffling., Tech. rep., Stanford University (2008).
- [4] N. T. Bailey, Mathematical Theory of Infectious Diseases and Its Applications, 2nd ed., Griffin, London, 1975.
- [5] R. Bakhshi, F. Bonnet, W. Fokkink, B. Haverkort, Formal analysis techniques for gossiping protocols, ACM SIGOPS Oper. Syst. Rev. 41 (5) (2007) 28–36.
- [6] R. Bakhshi, D. Gavidia, W. Fokkink, M. van Steen, An analytical model of information dissemination for a gossip-based protocol, in: Proc. Conf. on Distributed Computing and Networking (ICDCN 2009), vol. 5408 of LNCS, Springer, 2009, pp. 230–242.
- [7] D. Bayer, P. Diaconis, Trailing the dovetail shuffle to its lair., Annals of Applied Probability 2 (2) (1992) 294–313.
- [8] F. Bonnet, Performance analysis of Cyclon, an inexpensive membership management for unstructured P2P overlays, Master thesis, ENS Cachan Bretagne, University of Rennes, IRISA (2006).
- [9] S. Boyd, A. Ghosh, B. Prabhakar, D. Shah, Gossip algorithms: Design, analysis and applications, in: Proc. INFOCOM'05, vol. 3, IEEE, 2005, pp. 1653–1664.
- [10] D. J. Daley, J. Gani, Epidemic Modelling: An Introduction., Cambridge University Press, Cambridge, UK, 1999.
- [11] S. Deb, M. Médard, C. Choute, Algebraic gossip: a network coding approach to optimal multiple rumor mongering, IEEE/ACM TON 14 (SI) (2006) 2486–2507.

- [12] A. Demers, D. Greene, C. Hauser, W. Irish, J. Larson, S. Shenker, H. Sturgis, D. Swinehart, D. Terry, Epidemic algorithms for replicated database maintenance, in: Proc. 6th Annu. ACM Symp. on Principles of Distributed Computing (PODC'87), ACM Press, 1987, pp. 1–12.
- [13] A. P. Dempster, N. M. Laird, D. B. Rubin, Maximum likelihood from incomplete data via the EM algorithm, *J. Roy. Statist. Soc. B* 39 (1977) 1–38.
- [14] N. Drost, E. Ogston, R. van Nieuwpoort, H. Bal, ARRG: Real-World Gossiping, in: Proc. 15th Symp. on High Performance Distributed Computing (HPDC-15'06), ACM, 2007, pp. 147–158.
- [15] P. Eugster, R. Guerraoui, S. Handurukande, A.-M. Kermarrec, P. Kouznetsov, Lightweight Probabilistic Broadcast, *ACM TOCS* 21 (4) (2003) 341–374.
- [16] P. T. Eugster, R. Guerraoui, A.-M. Kermarrec, L. Massoulié, From epidemics to distributed computing, *IEEE Computer* 37 (5) (2004) 60–67.
- [17] D. Gavidia, S. Voulgaris, M. van Steen, A gossip-based distributed news service for wireless mesh networks, in: Proc. WONS'06, IEEE, 2006, pp. 59–67.
- [18] M. Gilleland, Working with Fractions in Java (2002).
URL <http://www.merriampark.com/fractions.htm>
- [19] I. Gupta, M. Nagda, C. F. Devaraj, The design of novel distributed protocols from differential equations, *Distributed Computing* 20 (2) (2007) 95–114.
- [20] P. Hilton, D. Holton, J. Pedersen, *Mathematical Reflections*, Springer-Verlag, New York, 1997.
- [21] M. Jelasity, A. Montresor, O. Babaoglu, Gossip-based aggregation in large dynamic networks, *ACM Trans. Comput. Syst.* 23 (3) (2005) 219–252.
- [22] M. Jelasity, S. Voulgaris, R. Guerraoui, A.-M. Kermarrec, M. van Steen, Gossip-based peer sampling, *ACM TOCS* 25 (3) (2007) 8.
- [23] R. Karp, C. Schindelhauer, S. Shenker, B. Vocking, Randomized rumor spreading, in: Proc. FOCS'00, IEEE, 2000, pp. 565–574.
- [24] D. Kempe, A. Dobra, J. Gehrke, Gossip-based computation of aggregate information, in: Proc. 44th Annu. IEEE Symp. on Found. of Comput. Sci., IEEE, 2003, pp. 482–491.
- [25] S. Y. Ko, I. Gupta, Y. Jo, A new class of nature-inspired algorithms for self-adaptive peer-to-peer computing, *ACM Trans. Auton. Adapt. Syst.* 3 (3) (2008) 1–34.
- [26] G. J. McLachlan, D. Peel, *Finite Mixture Models*, John Wiley & Sons, 2000.
- [27] M. Mitzenmacher, The power of two choices in randomized load balancing, *IEEE Trans. Parallel Distrib. Syst.* 12 (10) (2001) 1094–1104.
- [28] R. Graham, D. Knuth, O. Potashnik, *Concrete Mathematics*, 2nd ed., Addison-Wesley, 1994.

- [29] S. Voulgaris, D. Gavidia, M. van Steen, Cyclon: Inexpensive membership management for unstructured P2P overlays., *Netw. Syst. Manag.* 13 (2) (2005) 197–217.

A Coverage model (revisited)

The coverage model in Sec. 5.2 has an implicit assumption that a node shuffles two times per round (once when it initiates the gossip and once when it is contacted by another node). Although this is true on average, what actually happens is that a node initiates a shuffle once per round (with some random neighbour), but can be contacted an arbitrary number of times ($\leq N - 1$). In Fig. A.1), it is depicted for $k \geq 1$ which percentage of nodes, on average, shuffle k times in a given round. The number of times a node shuffles in a round has an impact on coverage. In this subsection, we revisit our coverage model to take into account that there is a probability distribution for the number of times a node is contacted.

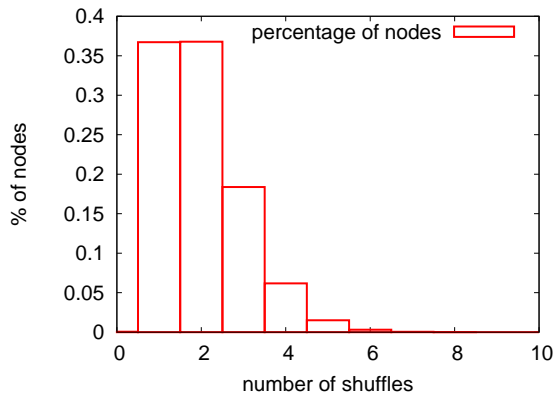


Fig. A.1. Distribution of nodes according to the number of shuffles they execute per round, for a fully connected network of $N = 2500$ nodes.

We compute the probability that a node is contacted i times by other nodes in the network:

$$C(i) = \binom{N-1}{i} \left(\frac{1}{N-1}\right)^i \left(\frac{N-2}{N-1}\right)^{(N-1)-i}, \quad i \leq N-1$$

Namely, there are $\binom{N-1}{i}$ ways to choose i nodes in a network of $N-1$ nodes that contact the given node. Those i nodes contact the given node with probability $\frac{1}{N-1}$ each, and the remaining $(N-1)-i$ nodes do not contact the given node with probability $\frac{N-2}{N-1}$.

A node may discover d during any of its shuffles. However, coverage is only

measured at the end of a gossip round, meaning that a node that sees item d for the first time and drops it in the same round is considered not to have seen item d at all.⁷ Consequently, coverage increases only under the following conditions: i) a node that has not seen item d obtains a copy of item d during one of the contacts, and ii) by the end of the round (after i exchanges), it still holds on to a copy of d .

In order to model coverage, we need to express the probabilities P_{get} that a node that does not have a copy of d gets this copy in a shuffle, and P_{lose} that a node that has a copy of d loses this copy in a shuffle.

$$P_{get} = P(1*|0*) = x \cdot P(1*|01) = x \cdot (P(10|01) + P(11|01))$$

$$P_{\neg get} = 1 - P_{get}$$

$$\begin{aligned} P_{\neg lose} &= P(1*|1*) = x \cdot P(1*|11) + (1-x) \cdot P(1*|10) \\ &= x \cdot (P(10|11) + P(11|11)) + (1-x) \cdot (P(10|10) + P(11|10)) \end{aligned}$$

$$P_{lose} = 1 - P_{\neg lose}$$

The increase in coverage from one round to the next can be modelled by identifying all the possible cases where a node that has previously not seen item d discovers it by the end of the round. Let Φ_i express the probability that a node that does not hold d , does hold d after performing i shuffles. We have

$$\Phi_i = \sum_{m=0}^{i-1} (1 - \Phi_m) \cdot P_{get} \cdot (P_{\neg lose})^{(i-m)-1}, \quad i \geq 0$$

where $\Phi_0 = 0$. Namely, the expression $(1 - \Phi_m) \cdot P_{get} \cdot (P_{\neg lose})^{(i-m)-1}$ captures the case where a node does not have the item at the end of $m < i$ shuffles (with probability $1 - \Phi_m$), then discovers it in the m th shuffle (with probability P_{get}), and does not lose it in the remaining shuffles (with probability $(P_{\neg lose})^{(i-m)-1}$).

In a given round, only the fraction $1 - y$ of nodes in the network that did not yet see item d can contribute to an increase in coverage. Such a node is contacted $i \geq 0$ times in the round with probability $C(i)$, and then performs $i + 1$ shuffles in total. With probability Φ_{i+1} the node will hold item d at the end of the round. Thus, coverage can be modelled by the equation

$$\frac{dy}{dt} = (1 - y) \cdot \sum_{i=0}^k C(i) \cdot \Phi_{i+1}$$

where k is the maximum number of times that a node is contacted in a round, i.e., $k = N - 1$.

⁷ The reason for this is that in general the application has an opportunity to read from the lower-level cache only once every round.

As before, $y(0) = \frac{1}{N}$, as initially only one of the N nodes holds d .

For a network of 2500 nodes with full connectivity, the probability of a node being contacted more than four times in one round is negligible (less than 1%). We therefore use the aforementioned coverage model with a limit of $k = 4$ to estimate the coverage and compare with our simulation traces. The results can be seen in Fig. A.2.

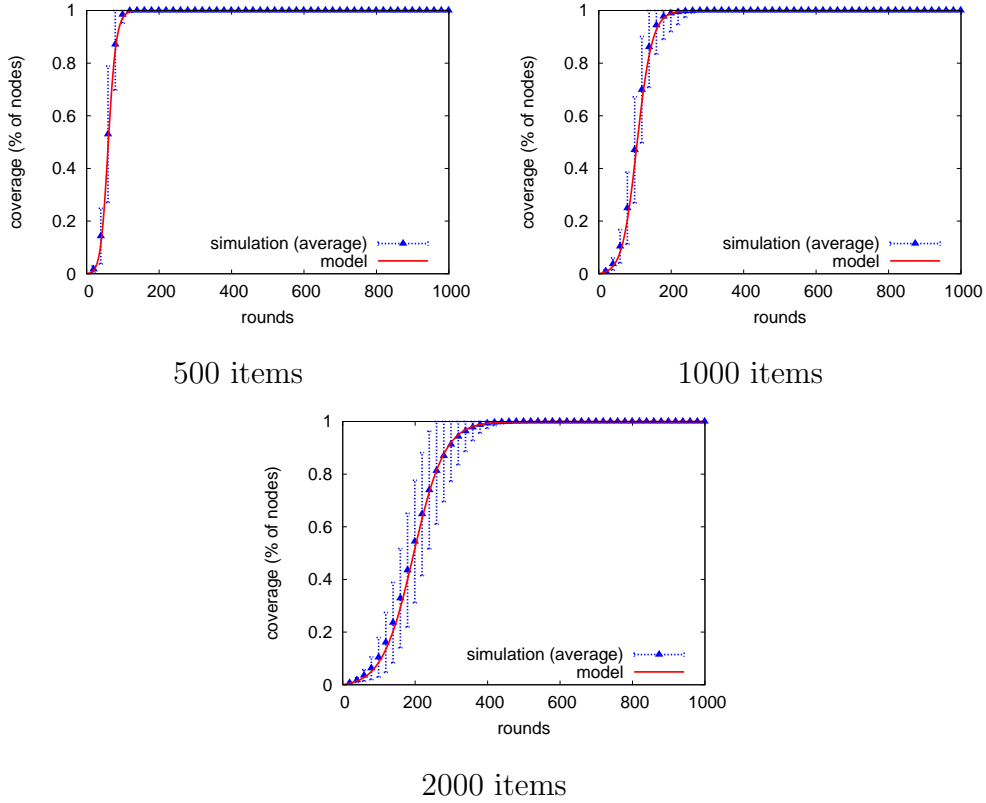


Fig. A.2. Percentage of nodes in the network that have already seen a replica of item d , for $N = 2500$, $c = 100$, $s = 50$, and $n = 500$, $n = 1000$ or $n = 2000$.

Unlike the results from Fig. 9, the current model not only falls into the standard deviation of the shuffle simulation results, but also closely reproduces the curve of average values in all three cases ($n = 500$, $n = 1000$ and $n = 2000$).