On the Uniformity of Peer Sampling based on View Shuffling

Yann Busnel[a], Roberto Beraldi[b], Roberto Baldoni[b]

[a] *Computer Sciences Department, LINA, University of Nantes – France.*
[b] *Department of Computer and Systems Sciences, MIDLAB, Sapienza University of Rome – Italy.*

Abstract

Consider a group of peers, an ideal random peer sampling service should return a peer, which is a uniform independent random sample of the group. This paper focuses on the implementation and analysis of a peer sampling service based on symmetric view shuffling, where each peer is equipped with a local view of size c, representing a uniform random sample of size c of the whole system. To this end, pairs of peers regularly and continuously swap a part of their local views (*shuffle operation*). The paper provides the following formal proofs: *(i)* starting from any non-uniform distribution of peers in the peers' local views, after a sequence of pairwise shuffle operations, each local view eventually represents a uniform sample of size c; *(ii)* once previous property holds, any successive sequence of shuffle operations does not modify this uniformity property and *(iii)* a *lower bound* for convergence speed. This paper also presents some numerical results concerning the speed of convergence to uniform samples of the local views.

Keywords: Peer sampling, Gossip-based protocol, Theoretical analysis, Stochastic process, Numerical evaluation

1. Introduction

Uniform peer sampling service has been shown recently to be a basic building block for several applications in large-scale distributed systems [1] as information dissemination [2], counting [3], clock synchronization [4], *etc*. A peer service is called *uniform* if it returns a peer ID of the whole system with the same probability. Working on the top of a not uniform peer sampling can affect the performance of the application using the service and, most importantly its correctness.

There are two main approaches to implement uniform random sampling, *random walk* and *gossip-based* protocols.

A random walk on a given graph is a sequential process that consists in visiting the nodes of the graph according to a random order induced by the way the walker is allowed to move.

The key property of a random walk is that, after a suitable number of steps, called the mixing-time, the visited node is the same as drawn from a uniform distribution [5],[3].

Unfortunately, the mixing-time depends on the topological property of the graph, which is generally unknown. Thus, for the reached node to be uniformly sampled, the length of the walk has to be properly tuned. Moreover, this technique may incur in a long delay to return a sample.

This paper focuses on uniform peer sampling based on gossip protocols. We consider a system formed by n peers (*i.e.*, nodes), each provided with a local view of size $c \leq n$. Each node runs a simple *shuffling protocol* where pairs of nodes regularly and continuously swap part of their local views (*shuffle operation*). This protocol is similar to the ones used in [6, 1, 7, 8]. The shuffling protocol aims that local views eventually represent a uniform random sample of the system. The main results presented in this paper show formally that:

1. starting from any non-uniform distribution of nodes in the local views, after a sufficiently long sequence of pairwise shuffle operations executed by the shuffling protocol, each local view represents a uniform random sample of size c among the whole system (Theorem 5.4);
2. once the previous property has been established, any sequence of successive shuffle operations does not modify the previous property (Corollary 5.3);
3. using this protocol, optimal setting can be identified in term of convergence speed (Theorem 5.6).

To the best of our knowledge, these results have never been formally proved before, despite the fact that there is empirical evidence shown in many papers [1, 8], that protocols based on view shuffling can indeed provide uniform sampling.

Let us remark that this result complements two previous outcomes respectively presented in [9] and [10].

Authors of [9] propose a protocol based on view shuffling and formally prove that this protocol converges to a uniform peer sampling also in the presence of byzantine peers. Each run of their protocol leads, after a sufficiently long sequence of shuffle operations, to verify the property: "each local view is a uniform random sample of the system". However, each time a user requires to get a new uniform sample, another instance of this protocol has to be started and it has to converge to a new uniform random sample. Conversely, the shuffling protocol presented in this paper shows that once the local view converges

Email addresses: Yann.Busnel@univ-nantes.fr (Yann Busnel), Roberto.Beraldi@dis.uniroma1.it (Roberto Beraldi), Roberto.Baldoni@dis.uniroma1.it (Roberto Baldoni)
URL: http://www.univ-nantes.fr/~busnel-y/ (Yann Busnel), http://www.dis.uniroma1.it/~beraldi/ (Roberto Beraldi), http://www.dis.uniroma1.it/~baldoni/ (Roberto Baldoni)

to represent a uniform sample of the system, then successive shuffle operations do not modify the property (Corollary 5.3). Therefore, there is a continuous availability of a uniform random sample without the need to start other instances of the base protocol[1], also known as Ergodicity in the literature [11].

Recently, a set of correctness properties for gossip-based membership protocol and a protocol designed for dealing with message loss, meeting these requirements, have been defined in [10]. The analytical framework used in the paper is based on the idea of graph transformation. A protocol defines a Markov Chain with states representing the distribution of the membership graphs and state transitions triggered by protocol's actions. The properties of the protocol are then deduced by the properties of an expected membership graph. Although this modeling approach is very general, it can provide some difficulty when applied to real protocols. For example, the proposed protocol is characterized by a dynamic view size and, in order to study the degree distribution, iterative numerical methods have to be used.

Our work uses a quite different and simpler modeling strategy, which is more suitable for studying symmetric shuffle operations among nodes equipped with fixed view size. We guess that such modeling approach is an interesting contribution *per se*, which can be generalized for studying membership protocols or used in conjunction with the graph transformation approach. For example, our model allows to formally proof the intuitive result that the maximum convergence speed to the uniform distribution is reached when nodes exchange a half of their views. An extended comparison between this paper and ours is presented in Section 7.

This paper is organized as follows: Section 2 presents the system model. Two shuffling protocols are presented in Section 3 while Section 4 provides an analytical model of these shuffling protocols. Section 5 proves that the local views shaped by these shuffling protocols converge to uniform random samples of the system, and that the convergence speed can be optimized using correct settings. Section 6 provide some stochastic evaluations in order to illustrate these latter formal outcomes according to the system parameters. Finally, related works and conclusion are given respectively in Section 7 and in Section 8.

2. System model

We consider a finite set of n nodes (with $n \geq 2$), which are uniquely identified through a system-wide identifier (ID). Each node i manages a local partial view of the system, denoted V_i of size $c \leq n$, about all the other nodes in the system, including itself[2]. In other word, each node owns a small set of connections (*aka* neighbors) corresponding to its vision of the overall system.

[1] A simple safe condition for keeping the sample returned by a node uniform is to "refresh" the returned ID. This requirement ties the sample with the shuffle rate, as explained in Section 5.5.

[2] Note that c is the same for all nodes in the system. This fixed value can be viewed as a parameter of the system.

The view of node i is modeled as a fixed-size set of binary random variables indicating whenever the identifier k appears in V_i or not

$$X_i = (X_{1i}, X_{2i}, \ldots, X_{ni})$$

where

$$X_{ki} = \begin{cases} 1 & \text{if } k \in V_i; \\ 0 & \text{otherwise} \end{cases}$$

The vector X_i is referred as the *characteristic vector* of the view. A vector X_i is associated with a *probability vector*

$$P_i = (p_{1i}, p_{2i}, \ldots, p_{ni})$$

where p_{ki} is the probability that k belongs to V_i, *i.e.*,

$$p_{ki} = \mathbb{P}[X_{ki} = 1].$$

The whole system is then modeled as the collection

$$S = (X_1, X_2, \ldots, X_n)$$

of the characteristic vectors corresponding to the nodes' view. The corresponding set of probability vectors is called a *configuration* of the system,

$$C = (P_1, P_2, \ldots, P_n).$$

The random process that determines X_i according to P_i is defined in the following Section 3, which introduces the shuffling operation.

Definition 2.1. *A view is* uniform *if at a given instant of time all IDs appear in this view with the same probability.*

Definition 2.2. *A system is called* uniform *if at a given instant of time, all views are uniform.*

In this paper we use the notion of potential function to deal with arbitrary configurations, according to the uniform state.

Definition 2.3. *The* local potential function *of given probability vector P is*

$$h(P) = \max_{p_k \in P} \left\{ p_k - \frac{c}{n} \right\}.$$

Definition 2.4. *The* potential function *of configuration C is*

$$h(C) = \max_{P_i \in C} \{h(P_i)\} = \max_{P_i \in C} \max_{p_{ki} \in P_i} \left\{ p_{ki} - \frac{c}{n} \right\}.$$

The potential function is a sort of distance measure between a generic configuration and the uniform configuration, namely the configuration with all probabilities equal to $\frac{c}{n}$. For such configuration, let us introduce the following Lemma 2.5.

Lemma 2.5. *Let C be a configuration of local views.*

$$h(C) \text{ is zero } \Leftrightarrow C \text{ is uniform.}$$

Proof. First of all, consider the following property:

Property 2.6. *The expected size of a view V_i is*

$$\mathbb{E}\left[\sum_k X_{ki}\right] = \sum_{k=1}^{n} \mathbb{E}[X_{ki}] = c$$

($\Leftarrow$) Let consider the configuration C as uniform, as presented in Definition 2.2. As all the views are uniform, all the probability for a node to appear in any view is the same, so called $\overline{p}$. Thus, from Property 2.6, we have:

$$\forall i,\ c = \sum_{k=1}^{n} \mathbb{E}[X_{ki}] = n \cdot \overline{p} \quad \Longrightarrow \quad \overline{p} = \frac{c}{n}.$$

and then, for all nodes, the local potential is zero. So, $h(C) = 0$.

($\Rightarrow$) On the other hand, if the potential function is zero ($h(C) = 0$), then, by definition, the maximum for any probability vector is $\frac{c}{n}$. Thus, from Property 2.6, this also implies that *all* the probabilities are equals to $\frac{c}{n}$, which is the definition of uniformity (*cf.* Definition 2.2). □

For convenience, we finally introduce the notation M_{ij} that denotes the expected number of shared elements between i and j. More formally,

$$\forall i, j, \quad M_{ij} = \sum_{k} \mathbb{P}[X_{ki} = 1] \cdot \mathbb{P}[X_{kj} = 1].$$

3. The shuffling protocol

We now consider a distributed protocol in which nodes manage their views by performing elementary pairwise shuffle or *shuffling operation*, denoted as $\diamond$. The notation $i \diamond j$ is used to denote that i performs a shuffling operation with j. The effect of an operation is to update the nodes' view, as detailed later in this section. We then show that the protocol makes the system to converge towards a uniform configuration, namely a configuration with zero potential value.

We assume that two shuffles involving a common node may not take place concurrently. Once a node initiates a shuffle, it will be locked until the operation is terminated.

The shuffling operation

The shuffling operation is the core aspect of the whole protocol. The shuffling protocol consists of applying the shuffling operation repeatedly to pairs of nodes i, j taken at random. How nodes are chosen to do the shuffling operation is discussed in more details in Section 4.3

This shuffling operation has one parameter, the shuffle length l, and involves two views, say V_i and V_j. For the sake of simplicity, we will also use the shuffle ratio, $\gamma = \frac{l}{c}$. The operation $\diamond$ acts as follows.

The view V_i (resp. V_j) is split into two random parts. The first part, denoted as ℓ_i (resp. ℓ_j), is the *sent view*, which is a subset of V_i (resp. V_j) of size l. The elements in ℓ_j are added to $V_i - \ell_i$, and inversely. If the size of this new set, $V_i' = (V_i - \ell_i) \cup \ell_j$ is lower than c (this could happen if ℓ_j and $V_i - \ell_i$ have common elements), then $l' = c - |V_i'|$ elements are taken from $\ell_i - \ell_j$ at random and added to V_i'. More formally, the shuffling operation consists of the steps presented in Algorithm 1. In the latter, UniRand(h, V) returns a subset of h elements taken uniformly at random from a set V. The shuffle operation is symmetric in the sense that node j acts exactly as node i. Moreover, the two nodes make their decisions about which elements to keep from the sent view, if any, independently from each other. Thus, the probability of a node k to appear in a view is only determined by the elements in the interacting nodes before the shuffle[3].

Consider the following example. Assume that $c = 7$ and $l = 3$. Consider then a shuffle between the views

$$\begin{cases} V_i &= \{0, 12, 1, 5, 3, 7, 8\} \\ V_j &= \{3, 11, 4, 5, 8, 2, 1\} \end{cases}$$

with sent subset $\ell_i = \{3, 7, 8\}$, $\ell_j = \{8, 2, 1\}$. We then have that the first manipulation:

$$V_i' = (V_i - \ell_i) \cup \ell_j$$

produces

$$V_i' = \{0, 12, 1, 5\} \cup \{8, 2, 1\} = \{0, 12, 1, 5, 8, 2\}$$

As $|V_i'| = 6$ while $c = 7$, we need to add some random elements of the set

$$\ell_i - \ell_j = \{3, 7, 8\} - \{8, 2, 1\} = \{3, 7\}$$

A remark on system partitioning

Unfortunately, the aforementioned operation can lead to a partition of the system. Indeed, consider a system composed by two distinct clusters which are linked by two edges ($i \to j$ and $j \to k$, where i, k belongs to the first cluster). Assume that $l = 1$. If i makes a shuffle with j, then it may happen that i sends j to j and j sends k back to i. If node j integrate j in its view and i replaces j with k, then the system remain partitioned in two clusters forever, namely none of the node in the first cluster is aware of nodes in the second cluster and *vice versa*.

This specific case can be avoided using a tricky mechanism, as in [8]. To guarantee that the system could not become partitioned after a shuffling operation, we drive the choice of the shuffling partner to be in ℓ_i and force the initiator node of the shuffle to send its own ID by replacing the one of the partner. This ensures any shuffling operation results into at least to a link reversal (*cf.* [8] for more details of the proof).

Our biased shuffling operation uses link swap as a technique for preserving system connectivity. The new operation is presented in Algorithm 2. The initiator node chooses l nodes in its current view to fill ℓ_i'. Then, as explained above, it chooses the partner j at random and replaces it by its own ID to obtain ℓ_i, unless i already appears in ℓ_i' (this is required to keep the view size constant).

A remark on message loss

One could argue that our simple protocol does not take into account message loss. In fact, since a shuffling operation involves information exchange between two nodes, it takes some time to proceed. If the amount of information exchange is large or the nodes are topologically separated by many IP hops, then

[3] A correlation would arise if, for example, node i decides to add the identifier k received from j only if j promises something else back.

Algorithm 1: Basic shuffling operation

`node` i	`node` j
$\ell_i \leftarrow \text{UniRand}(l, V_i)$	$\ell_j \leftarrow \text{UniRand}(l, V_j)$
$V'_i \leftarrow (V_i - \ell_i) \cup \ell_j$	$V'_j \leftarrow (V_j - \ell_j) \cup \ell_i$
$V_i \leftarrow V'_i \cup \text{UniRand}(c - \lvert V'_i \rvert, \ell_i - \ell_j)$	$V_j \leftarrow V'_j \cup \text{UniRand}(c - \lvert V'_j \rvert, \ell_j - \ell_i)$

Algorithm 2: Biased shuffling operation

`node` i	`node` j
$\ell'_i \leftarrow \text{UniRand}(l, V_i)$	$\ell_j \leftarrow \text{UniRand}(l, V_j)$
$j \leftarrow \text{UniRand}(1, \ell'_i)$	⋮
if $i \notin \ell'_i$ **then**	⋮
$\quad \ell_i \leftarrow (\ell'_i - \{j\}) \cup \{i\}$	⋮
else	⋮
$\quad \ell_i \leftarrow \ell'_i$	⋮
$V'_i \leftarrow (V_i - \ell_i) \cup \ell_j$	$V'_j \leftarrow (V_j - \ell_j) \cup \ell_i$
$V_i \leftarrow V'_i \cup \text{UniRand}(c - \lvert V'_i \rvert, \ell_i - \ell_j)$	$V_j \leftarrow V'_j \cup \text{UniRand}(c - \lvert V'_j \rvert, \ell_j - \ell_i)$

some shuffling operations might not be able to "finish on time". In reality, as c is usually chosen very small, each sent vector is quite small. Moreover, we can consider a shuffling operation as a semaphore [12]. An initiated shuffling operation will delay any following request until the former one is not finished. As we do not consider node failure, our shuffling operation is able to perform under message loss using timeout and reemission.

In principle for high message loss probability, the buffer used for delaying requests might overflow. Moreover, as nodes will delay new request, a shuffling requester could be blocked for a while. Recursively, a chain of blocked node can occurs, implying classical asynchronous issues. How that could affect our theoretical result remains an open question for future works.

n	Total number of nodes in the system
c	Size of each local view
l	Size of the sent vector
γ	Shuffle ratio ($\gamma = \frac{l}{c}$)
V_i	Local view of node i
ℓ_i	Sent view of node i
X_{ki}	Indication function
X_i	Characteristic vector of view V_i
P_i	Probability vector of view V_i
$\overline{p}$	Expected uniform probability ($\overline{p} = \frac{c}{n}$)
M_{ij}	Expected number of shared elements between i and j
$i \diamond j$	Shuffling operation (i shuffles with j)

Table 1: List of main symbols

4. Protocol analysis

In this section, we derive an analytical model of the shuffling protocol, which captures the variation of the system configuration over time. The main symbols used in this paper are reported in Table 1.

4.1. View evolution under the basic shuffling operation

Let consider how the presence of element k in the view of i varies after a shuffling operation among nodes i and j. A shuffling operation between i and j, denoted $i \diamond j$, generates two new characteristic vectors, X'_i and X'_j, starting from the original vectors X_i and X_j. In other words, after the operation, the view of node i (resp. j) is described by X'_i (resp. X'_j).

The evolution of the view over time is then described by a relationship among X and X'. Before describing this relationship, it is important to understand that X'_{ki} is independent from the others random variables X'_{kj}. In fact, the elements that are inserted or removed due to shuffling into the view of node i, are not influenced by elements inserted/removed into the view of node j. In other words, as explained above, i and j do not coordinate somehow about their decisions on the way to change the views. In fact, nodes i and j act locally and then, make any decision by them-self, independently from each other. Let $P_{1\to0}$ be the probability that, after the shuffle, node k is removed from V_i and $P_{0\to1}$ the probability that k is inserted (for the sake of simplicity indexes are omitted), namely

$$\begin{cases} P_{1\to0} = \mathbb{P}[X'_{ki} = 0 | X_{ki} = 1] \\ P_{0\to1} = \mathbb{P}[X'_{ki} = 1 | X_{ki} = 0] \end{cases}$$

The probability that k appears in V_i, given that $i \diamond j$, is then

$$\begin{aligned} \mathbb{P}[X'_{ki} = 1 | i \diamond j] = \quad & (1 - \mathbb{P}[X_{ki} = 1]) \cdot P_{0\to1} \\ + \quad & \mathbb{P}[X_{ki} = 1] \cdot (1 - P_{1\to0}) \end{aligned} \tag{1}$$

This expression has the following meaning. The probability that node k appears in i's view, after a shuffle between i and j, is given by the probability that k was not in the view and it has been added or the probability that k was already in the view and

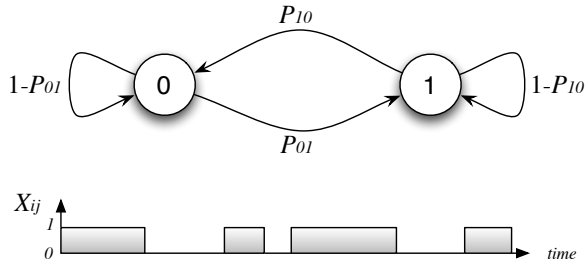


Figure 1: Markov chain representing the evolution of X_{ij}, the presence of element i into the view of node j. When the view is uniform, the fraction of time that the element appears in the view, (*i.e.*, $X_{ij} = 1$) is the same for any element. This condition corresponds to the steady state of the chain.

it has not been deleted. The evolution of a view is best described as a two states Markov chain, see Figure 1, where state 1 (resp. state 0) means that k is (resp. not) in the node i's view.

Node i receives l elements from j. As an element is sent with probability γ, the expected number of elements that ℓ_j and V_i have in common is:

$$\begin{aligned}&\textstyle\sum_k \mathbb{P}[X_{ki} = 1] \cdot \gamma \cdot \mathbb{P}[X_{kj} = 1]\\ &= \gamma \cdot \textstyle\sum_k \mathbb{P}[X_{ki} = 1]\mathbb{P}[X_{kj} = 1]\\ &= \gamma \cdot M_{ij}\end{aligned}$$

Now, the view size must remain constant. The expected number of elements removed from i must then be equal to the number of *new* elements added into V_i, which is equal to $l - \gamma M_{ij}$. As all elements have to equally likely be removed, the probability to remove the element k is $\frac{l-\gamma M_{ij}}{c}$. From which:

$$P_{1\to 0} = \gamma \cdot \left(1 - \frac{\sum_k \mathbb{P}[X_{kj} = 1]\mathbb{P}[X_{ki} = 1]}{c}\right)$$

On the other hand, as element k can be added only if it belongs to ℓ_j, we have:

$$P_{0\to 1} = \gamma \cdot \mathbb{P}[X_{kj} = 1].$$

4.2. View evolution under the biased shuffling operation

In this case, we have to distinguish between the initiator and the partner of the shuffle.

4.2.1. Initiator view change

As the partner of an exchange is not forced to put any ID into the sent view, the view evolution of the initiator, which receives the sent view, remains as described above.

4.2.2. Partner point of view

In this case, we have to model the biased part of the shuffle. We have:

$$\mathbb{P}[X'_{ki} = 1 | j \diamond i] =$$

$$\begin{cases} 1 & \text{if } k = j \\ (1 - \mathbb{P}[X_{ii} = 1]) \cdot \gamma \cdot \mathbb{P}[X_{jj} = 1] & \\ \quad + \mathbb{P}[X_{ii} = 1] \cdot (1 - P_{1\to 0}) & \text{if } k = i \\ (1 - \mathbb{P}[X_{ki} = 1])P^*_{0\to 1} & \\ \quad + \mathbb{P}[X_{ki} = 1](1 - P^*_{1\to 0}) & \text{otherwise.} \end{cases} \quad (2)$$

First, as j sends his own identifier in each exchange that it initiates, we have: $\mathbb{P}[X'_{ji} = 1 | j \diamond i] = 1$.

If the identifier j is already in ℓ'_j, then the sent vector contains i. Otherwise, i is replaced by j and so i is not sent. Thus, j sends i with probability $\gamma \cdot \mathbb{P}[X_{jj} = 1]$.

Thus, the probability that after the exchange i contains its own identifier is the sum of two contributions corresponding to the following events: (i) i was not present but it was received from j and (ii) i was present before the exchange and it was not replaced.

Finally, in all other cases, the value of $\mathbb{P}[X'_{ki} = 1 | j \diamond i \wedge k \neq j \wedge k \neq i]$ is given by the expression given in Equation 1 by adapting $P_{0\to 1}$ and $P_{1\to 0}$, respectively with $P^*_{0\to 1}$ and $P^*_{1\to 0}$. In fact, as one of the slot of V_i and ℓ_j contains the ID j for sure, the effective size of i's view and the sent vector has to be reduced by one unit.

Thus, we have:

$$\begin{cases} P^*_{0\to 1} = \dfrac{l-1}{c-1} \cdot \mathbb{P}[X_{kj} = 1] \\ P^*_{1\to 0} = \dfrac{l-1}{c-1} \cdot \left(1 - \dfrac{\sum_{k\neq j} \mathbb{P}[X_{kj} = 1]\mathbb{P}[X_{ki} = 1]}{c-1}\right) \end{cases}$$

4.3. Evolution of the system

Let now consider how the system evolves. As explained above, we assume that concurrent operations cannot occur. Thus, we can serialize parallel shuffles in an arbitrary order and assume that only one shuffling operation may take place at a time. Let $P_{ex}(i, j)$ be the probability that i and j make the shuffle, *i.e.*, $P_{ex}(i, j)$ is the probability that the operation $i \diamond j$ takes place.

As all the nodes ideally initiates an exchange at the same rate, we can consider that the initiator node is selected at random among all the n nodes. The target node j is taken at random from ℓ_i. Hence

$$P_{ex}(i, j) = \frac{1}{n} \cdot \mathbb{P}[X_{ji} = 1] \cdot \gamma \cdot \frac{1}{l} = \frac{1}{n} \cdot \mathbb{P}[X_{ji} = 1] \cdot \frac{1}{c}.$$

We can describe the global evolution of the system with the following expression:

$$\mathbb{P}[X'_{ki} = 1] =$$

$$\sum_j P_{ex}(i,j) \cdot \mathbb{P}[X'_{ki} = 1 | i \diamond j] \tag{3a}$$

$$+ \sum_j P_{ex}(j,i) \cdot \mathbb{P}[X'_{ki} = 1 | j \diamond i] \tag{3b}$$

$$+ \left(1 - \sum_j (P_{ex}(i,j) + P_{ex}(j,i))\right) \cdot \mathbb{P}[X_{ki} = 1] \tag{3c}$$

This last equation means that the probability vector of a node follows the view evolution presented in Equation 1 if it is involved in a view shuffle (Equation 3a and 3b) and remains the same if it is not involved in the last shuffle (Equation 3c).

5. Convergence property of the protocol

Let now consider the converge property of both shuffling protocols. In particular, we show that if any of the two shuffling protocol is executed by a system with arbitrary view distribution, then eventually the system converges towards a uniform configuration, *i.e.*, a system in which all the local views represent uniform random samples of the system. In order to show this result, we exploit the notion of potential function, introduced in Section 2. We will show that if the potential function of a configuration is greater than zero, then after a shuffling operation the potential function of the configuration is reduced. Roughly speaking, this means that a shuffling operation moves the system towards a "more" uniform system, or, in other words, makes the system closer to the uniform configuration.

5.1. Basic shuffling operation

Formally, using the first protocol described in Algorithm 1, we have:

Lemma 5.1 (Operator $\diamond$ reduces the potential)**.** *Let C and C' respectively the configuration of the system before and after a* basic *shuffling operation. If C is not the uniform configuration (*i.e.*, $h(C) \neq 0$), then $h(C') < h(C)$.*

Proof. Let P and Q be two probability vectors of nodes i and j and let P', Q' be these vectors after a shuffling operation $i \diamond j$. For the sake of simplicity, we denote the maximum probability before the shuffle as $p^{\max} = h(C) + \frac{c}{n}$. We prove below that $\forall k \in [1..n]$, (1) $\Delta p_k = p'_k - p^{\max} < 0$, and that (2) $\Delta q_k = q'_k - p^{\max} < 0$ where p_k, q_k, p'_k and q'_k denote respectively $P[k]$, $Q[k]$, $P'[k]$ and $Q'[k]$. This means that the highest probability decreases and no other probabilities can become greater than the previous maximum.

(1) We want to prove that $\Delta p_k < 0$. From equation 1, we have:

$$p'_k = (1 - p_k) \cdot \gamma \cdot q_k + p_k \cdot \left(1 - \gamma + \gamma \frac{M_{ij}}{c}\right)$$

$$\begin{aligned} \implies \Delta p_k &= p'_k - p^{\max} \\ &= p_k + \gamma \left(q_k - q_k \cdot p_k - p_k + p_k \cdot \frac{M_{ij}}{c}\right) - p^{\max} \end{aligned}$$

We distinguish two cases:

1. If $\left(q_k - q_k \cdot p_k - p_k + p_k \cdot \frac{M_{ij}}{c}\right) < 0$, as $p_k \leq p^{\max}$ and $\gamma \geq 0$, we obtain $\Delta p_k < 0$.
2. If $\left(q_k - q_k \cdot p_k - p_k + p_k \cdot \frac{M_{ij}}{c}\right) \geq 0$:

As $M_{ij} = \sum_k p_k \cdot q_k < \sum_k p^{\max} \cdot q_k = p^{\max} \cdot c$ and $\gamma \leq 1$, we have:

$$\begin{aligned} \Delta p_k &\leq q_k - q_k \cdot p_k + p_k \cdot \frac{M_{ij}}{c} - p^{\max} \\ &< q_k - q_k \cdot p_k + p_k \cdot p^{\max} - p^{\max} \\ &= q_k(1 - p_k) - p^{\max}(1 - p_k) = (q_k - p^{\max}) \cdot (1 - p_k) \\ &\leq 0 \qquad\qquad \text{as } \forall i, \quad p_k \leq p^{\max} \leq 1 \text{ and } q_k \leq p^{\max}. \end{aligned}$$

(2) It remains to prove the same upper bound for Q', *i.e.*

$$\forall i, \Delta q_k = q'_k - p^{\max} < 0.$$

According to the Equation 1, by symmetry, we have:

$$\mathbb{P}[X'_{ki} = 1 | j \diamond i] = q_k + \gamma \left(p_k - p_k \cdot q_k - q_k + q_k \cdot \frac{M_{ij}}{c}\right).$$

Thus, following the same reasoning, we obtain that

$$\forall i, \Delta q_k < 0.$$

Therefore, we can conclude that

$$\max\{h(P'), h(Q')\} < \max\{h(P), h(Q)\} \leq p^{\max}.$$

Let us denote $\mathcal{S}_i = \sum_j P_{ex}(i,j)$ and $\mathcal{S}'_i = \sum_j P_{ex}(j,i)$. Then, according to the Equation 3, we obtain:

$$\forall i, k, \quad \mathbb{P}[X'_{ki} = 1] < \mathcal{S}_i \cdot p^{\max} + \mathcal{S}'_i \cdot p^{\max} + (1 - \mathcal{S}_i - \mathcal{S}'_i) \cdot p^{\max} = p^{\max}.$$

Then, $h(C') = \max_{P_i \in C'} h(P_i) < p^{\max} - \frac{c}{n} = h(C)$ as claimed. □

5.2. Biased shuffling operation

We are now proving the same lemma for the second operation described in Algorithm 2:

Lemma 5.2 (Operator $\diamond$ reduces the potential)**.** *Let C and C' respectively the configuration of the system before and after a* biased *shuffling operation. If C is not the uniform configuration (*i.e.*, $h(C) \neq 0$), then $h(C') < h(C)$.*

Proof. Let P and Q be two probability vectors of nodes i and j and let P', Q' be these vectors after a shuffling operation. Moreover, p_k, q_k, p'_k and q'_k denote respectively $P[k]$, $Q[k]$, $P'[k]$ and $Q'[k]$.

For the sake of simplicity, we also denote in this proof the maximum probability before the shuffle as $p^{\max} = h(C) + \frac{c}{n}$.

Let split the study according to the three members of Equation 3. We prove below that the highest probability decreases and no other probabilities can become greater than the previous maximum.

(1) Consider the Equation 3a. Section 4.2.1 presents the view evolution from the initiator point of view. As the view evolution

in this case is exactly the same as in the basic shuffle, lemma 5.1 gives us, for all k:

$$\mathbb{P}[X'_{ki} = 1 | i \diamond j] < p^{\max}$$

(2) Consider now the Equation 3b. As explicitly denoted in Section 4.2.2, we have to split the analysis in 2 cases: *(2a)* $k \neq i$ and *(2b)* $k = i$.

2a First consider the case $k \neq i$. In this case, we have:

$$\begin{aligned}
&\sum_j P_{ex}(j,i) \cdot \left(\mathbb{P}[X'_{ki} = 1 | j \diamond i] - p^{\max}\right) \\
=\ &\sum_{j \neq k} P_{ex}(j,i) \cdot \left(\mathbb{P}[X'_{ki} = 1 | j \diamond i] - p^{\max}\right) \\
&+ P_{ex}(k,i) \cdot \left(\mathbb{P}[X'_{ki} = 1 | k \diamond i] - p^{\max}\right) \\
=\ &\frac{1}{n} \cdot \frac{1}{c} \cdot \sum_{j \neq k} p_j \cdot \left((1-p_k) \cdot P^*_{0\to 1} + p_k \cdot (1 - P^*_{1\to 0}) - p^{\max}\right) \\
&+ \frac{1}{n} \cdot p_k \cdot \frac{1}{c} \cdot (1 - p^{\max}) \\
<\ &\frac{1}{n} \cdot p_k \cdot \frac{1}{c} \cdot (1 - p^{\max}) \\
&+ \frac{n-1}{n} \cdot \frac{1}{c} \cdot p^{\max} \cdot \left((1-p_k) \cdot \frac{l-1}{c-1} \cdot q_k \right. \\
&\left. + p_k \cdot \left(1 + \frac{l-1}{c-1} \cdot \left(\frac{M_{ij}}{c-1} - 1\right)\right) - p^{\max}\right)
\end{aligned}$$

As in the previous proof, with here $M_{ij} < p^{\max} \cdot (c-1)$ and $\frac{l-1}{c-1} \leq 1$, we have:

$$\begin{aligned}
<\ &\frac{1}{n} \cdot p_k \cdot \frac{1}{c} \cdot (1 - p^{\max}) \\
&+ \left(\frac{1}{n}\right) \cdot \frac{1}{c} \cdot p^{\max} \cdot (q_k - p_k \cdot q_k + p_k \cdot p^{\max} - p^{\max}) \\
<\ &\frac{1}{n} \cdot (1 - p^{\max}) + \left(\frac{1}{n}\right) \cdot (q_k - p^{\max}) \cdot (1 - p_k)
\end{aligned}$$

Thus, the latter expression is lower than 0 if $\frac{1}{n} + \left(1 - \frac{1}{n}\right) \cdot (p_i - p^{\max}) \leq 0$. This inequality is equivalent to $\frac{1}{n} \cdot (1 + p^{\max} - p_i) \leq p^{\max} - p_i \Leftrightarrow \frac{1}{n} \leq 1 - \frac{1}{1+p^{\max}-p_i} \leq 1 - \frac{1}{2} = \frac{1}{2}$. Hence, as $n \geq 2$ by definition, we obtain that:

$$k \neq i \Rightarrow \sum_j P_{ex}(j,i) \cdot \left(\mathbb{P}[X'_{ki} = 1 | j \diamond i] - p^{\max}\right) < 0.$$

2b Let consider the case $k = i$. Here, we have:

$$\begin{aligned}
&\mathbb{P}[X'_{ii} = 1 | j \diamond i] - p^{\max} \\
&= (1-p_i) \cdot \gamma \cdot q_j + p_i \cdot \left(1 - \gamma \cdot \left(1 - \frac{M_{ij}}{c}\right)\right) - p^{\max} \\
&< q_j + p_i \cdot q_j + p_i - p_i - p_i \cdot p^{\max} - p^{\max} \quad \text{as } \gamma \leq 1 \\
&= (q_j - p^{\max}) \cdot (1 - p_i) < 0
\end{aligned}$$

Then, we obtain that, for all k:

$$\sum_j P_{ex}(j,i) \cdot \mathbb{P}[X'_{ki} = 1 | j \diamond i] < \sum_j P_{ex}(j,i) \cdot p^{\max}$$

(3) Finally, consider now the Equation 3c. In this part, we simply have to apply the assumption:

$$\mathbb{P}[X_{ki} = 1] < p^{\max}$$

Let us denote $\mathcal{S}_i = \sum_j P_{ex}(i,j)$ and $\mathcal{S}'_i = \sum_j P_{ex}(j,i)$. Then, according to the Equation 3, we obtain:

$$\forall i,k, \quad \mathbb{P}[X'_{ki} = 1] < \mathcal{S}_i \cdot p^{\max} + \mathcal{S}'_i \cdot p^{\max} + (1 - \mathcal{S}_i - \mathcal{S}'_i) \cdot p^{\max} = p^{\max}.$$

Then, $h(C') = \max_{P_i \in C'} h(P_i) < p^{\max} - \frac{c}{n} = h(C)$ as claimed. □

5.3. Convergence of both protocols

Let us now show a corollary stating that once local views represent uniform samples of the system, the shuffling protocol keeps this property true forever.

Lemma 5.3 (Operator $\diamond$ preserves uniformity)**.** *Let C be a uniform unpartitioned configuration of local views. A shuffling operation executed by the shuffling protocol presented in Section 3 between any pair of two local views X_i and X_j belonging to C produces a configuration C' that is uniform.*

Proof. Lemmata 5.1 and 5.2 give us that the potential of two views involved in a shuffling operation can only decrease. Given the fact that C corresponds to the uniform configuration, X_i and X_j are uniform and $P_i = P_j$ are vectors with all elements equal to $\overline{p} = \frac{c}{n}$. Thus, due to Lemma 2.5 and Definition 2.4, the potential of X_i and X_j are $h(P_i) = h(P_j) = 0$. From Lemma 5.1 or 5.2, after the shuffle, $h(P_i)$ and $h(P_j)$ cannot increase and thus, remain to 0. Then, C' is the uniform configuration.

More formally, it is possible to compute analytically the evolution of each probability vector. From the hypothesis on C, we have $\forall k, \mathbb{P}[X_{ki} = 1] = \frac{c}{n}$. Hence:

$$\begin{cases}
\mathbb{P}_{0\to 1} = \dfrac{l}{c} \times \dfrac{c}{n} = \dfrac{l}{n} \\
\mathbb{P}_{1\to 0} = \dfrac{l}{c} \cdot \left(1 - \dfrac{\sum_j \frac{c}{n} \frac{c}{n}}{c}\right) = \dfrac{l}{c} \cdot \left(1 - \dfrac{c}{n}\right)
\end{cases}$$

With these values, from Equation 1, we have:

$$\mathbb{P}[X'_{ki} = 1 | i \diamond j] = \left(1 - \frac{c}{n}\right)\frac{l}{n} + \frac{c}{n}\left(1 - \frac{l}{c} \cdot \left(1 - \frac{c}{n}\right)\right) = \frac{c}{n}. \quad \square$$

We are now in the position to state the following theorem, independently of the shuffling operation considered:

Theorem 5.4 (Convergence to uniformity)**.**
Let t be the number of shuffling operations executed on a system of n nodes, C_0 be any initial unpartitioned configuration of local views and C_t be the configuration of the system after those t shuffling operations. Local views built by the shuffling protocol presented in Section 3 will converge to uniform random samples of the system, i.e.*,*

$$\forall C_0, \lim_{t\to\infty} h(C_t) = 0.$$

Proof. First, the claim comes from Lemmata 5.1 and 5.2, as a shuffling operation *strictly* reduce the global potential, independently of the pair involved in the shuffle, while $h(C_t) \neq 0$. Thus, the distance of the current distribution of sample with the uniformity could only monotonically reduce. So, we have:

$$\exists m \in [0, h(C_0)], \ \lim_{t\to\infty} h(C_t) = m.$$

On the other hand , it remains to prove that $m = 0$. Consider a given $t \in \mathbb{N}$ in the following. Let split the rest of the proof in two parts:

1. If $h(C_t) = 0$: given Lemma 5.3, $\forall t' > t, h(C_{t'}) = 0$ and then, $\lim_{t\to\infty} h(C_t) = 0$.
2. If $h(C_t) > 0$: we proof in the last part of the proof that

$$h(C_{t+1}) < h(C_t) \cdot \left(1 - \frac{l}{n^2}\right) \tag{4}$$

$$\implies h(C_t) \leq f(t) = \left(1 - \frac{l}{n^2}\right)^t.$$

As $0 \leq h(C_t) \leq f(t)$ and $\lim_{t\to\infty} f(t) = 0$, we obtain that:

$$\forall C_0, \lim_{t\to\infty} h(C_t) = 0.$$

Finally, let us prove Equation 4. For the sake of simplicity, we also denote in this proof the maximum probability before the shuffle as $p^{\max} = h(C_t) + \frac{c}{n}$. Given the global evolution of the system, Equation 3 for any node i and k gives us:

$\mathbb{P}[X_{ki}^{t+1} = 1] =$

$$\sum_j P_{ex}(i,j) \cdot \mathbb{P}[X_{ki}^{t+1} = 1 | i \diamond j] \tag{5a}$$

$$+ \sum_j P_{ex}(j,i) \cdot \mathbb{P}[X_{ki}^{t+1} = 1 | j \diamond i] \tag{5b}$$

$$+ \left(1 - \sum_j (P_{ex}(i,j) + P_{ex}(j,i))\right) \cdot \mathbb{P}[X_{ki}^{t} = 1] \tag{5c}$$

Due to Lemmata 5.1 and 5.2, we have:

$$(5b) \leq \sum_j P_{ex}(j,i) \cdot p^{max}$$

By definition of p^{max}, we also have:

$$(5c) \leq \left(1 - \sum_j (P_{ex}(i,j) + P_{ex}(j,i))\right) \cdot p^{max}$$

Then:

$$\begin{aligned}(5b) + (5c) &\leq \left(1 - \sum_j (P_{ex}(i,j))\right) \cdot p^{max} \\ &\leq \left(1 - \sum_j \left(\frac{1}{c} \cdot \frac{1}{n} \cdot p^{max}\right)\right) \cdot p^{max} \\ &\leq \left(1 - \frac{p^{max}}{c}\right) \cdot p^{max}\end{aligned}$$

On the other hand, by definition:

$$\mathbb{P}[X_{ki}^{t+1} = 1 | i \diamond j] = (1 - \mathbb{P}[X_{ki} = 1]) \cdot P_{0\to1} + \mathbb{P}[X_{ki} = 1] \cdot (1 - P_{1\to0}).$$

As $(1 - \mathbb{P}[X_{ki} = 1]) \leq 1$ and $P_{0\to1} \leq \gamma \cdot p^{max}$, we have:

$$(5a) \leq \sum_j P_{ex}(j,i) \cdot \left(\gamma \cdot p^{max} + p^{max} \cdot \left(1 - \gamma\left(1 - \frac{M_{ij}}{c}\right)\right)\right)$$

Moreover, $M_{ij} < c \cdot p^{max}$ (*cf.* Proof of Lemma 5.1) implies:

$$(5a) < \sum_j P_{ex}(j,i) \cdot (\gamma \cdot p^{max} + p^{max} \cdot (1 - \gamma + \gamma \cdot p^{max}))$$

As $\sum_j P_{ex}(j,i) = \sum_j (\frac{1}{c} \cdot \frac{1}{n} \cdot \mathbb{P}[X_{ji} = 1]) = \frac{1}{n}$, we obtain:

$$(5a) < \frac{p^{max}}{n} \cdot (1 - \gamma \cdot p^{max})$$

Finally, we have:

$$\mathbb{P}[X_{ki}^{t+1} = 1] < p^{max} \cdot \left(1 - \frac{p^{max}}{c} + \frac{1}{n} - \gamma \cdot \frac{p^{max}}{n}\right)$$

As $p^{max} \geq \frac{c}{n}$ and $\frac{c}{n} \geq (1 - \frac{l}{n^2}) \cdot \frac{c}{n}$, we obtain:

$$\begin{aligned}h(C_{t+1}) &= \max_{P_i \in C_{t+1}} h(P_i) \\ &< p^{max} \cdot \left(1 - \frac{1}{c} \cdot \left(\frac{c}{n} + \frac{c \cdot l}{n^2}\right) + \frac{1}{n}\right) - \left(1 - \frac{l}{n^2}\right) \cdot \frac{c}{n} \\ &= h(C_t) \cdot \left(1 - \frac{l}{n^2}\right).\end{aligned}$$

□

5.4. Convergence speed lower bound

Experimental approaches [13, 1, 6, 7, 8] point out that, in the design of a gossip-based protocol, l has to be set to the half of c in order to obtain the highest efficiency in term of convergence speed. This conjecture can be intuitively shown as sketched below.

A shuffle operation with a sent vector ℓ between two nodes is equivalent to a shuffle with the complementary of ℓ (*i.e.* $V - \ell$), followed by swapping the ID of these two nodes (*cf.* Figure 2). Indeed, in this figure, the content of V_i after the shuffle on the left side is equivalent to the content of V_i on the right side after *(1)* a shuffle with the sent vector $\ell'_i = V_i - \ell_i$ and *(2)* swapping the node's ID (i becomes k and *vice versa*).

Now, consider $l \leq \frac{c}{2}$. It is obvious that the higher the size of the sent vector, the greater the effectiveness of a shuffle. Moreover, according to the above equivalence, a shuffle with l is equivalent to a shuffle with $c - l$. Thus, for $l \geq \frac{c}{2}$, the lesser the size of the sent vector, the greater the effectiveness of a shuffle. So, the greatest effectiveness is reached for $l = \left\lfloor \frac{c}{2} \right\rfloor$, as confirmed numerically in Figure 8.

A formal proof is now given in Theorem 5.6 below. First, we have to express in a measure of the effectiveness:

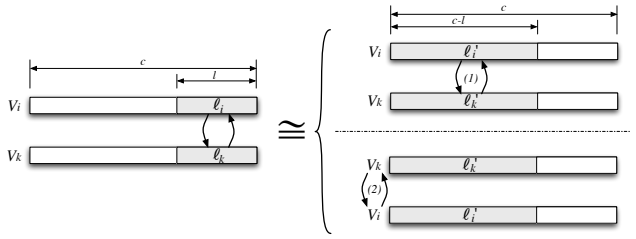


Figure 2: Intuitive equivalence between a small ℓ value and the opposite $c - \ell$ ones.

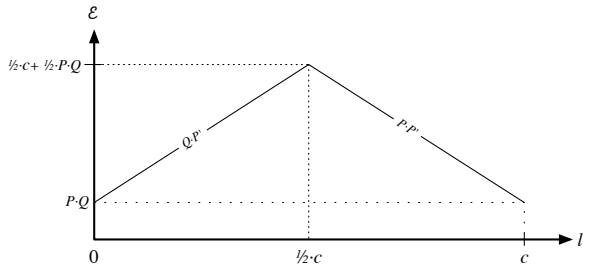


Figure 3: Effectiveness $\mathcal{E}(P')$ according to l for a fixed view size c.

Definition 5.5 (Shuffling Effectiveness)**.** *The effectiveness of a shuffle operation correspond to the magnitude of difference between an update view and both of the views before the shuffle,* i.e.*:*

$$\mathcal{E}(P') = \min\{P \cdot P', Q \cdot P'\}$$

where $P \cdot P'$ represent the scalar product between the vectors P and P'.

Indeed, according to the reasoning above, the core idea of a shuffling operation is to mix as much as possible both views involved in the shuffle. Thus, having a high difference between P' and its initial state P (nor partner state Q) is a good metric.

Starting from this definition, we should *maximize the effectiveness* of each operation. We prove below that this maximization is achieved for $\gamma = 0.5$.

Theorem 5.6 (Greatest Effectiveness of a Shuffle)**.** *Given two probability vectors P and Q. The maximum value of the expected effectiveness is reached for $\gamma = \frac{1}{2}$.*

Proof. Consider that P (resp. Q) is the probability vector of node i (resp. j). Let us remark that $P \cdot Q = \sum_k p_k \cdot q_k = M_{ij}$. Then, $P \cdot Q$ represents the expected number of elements in common in V_i and V_j.

Moreover, the number of *new* elements added in V_i after a shuffle corresponds to the size of of the partner's sent view ℓ_j, minus the number of elements in this sent view which was yet in V_i. As we show in Section 4 that $|V_i \cap \ell_j| = \gamma \cdot M_{ij}$, this number of new elements inserted is given by:

$$l - \frac{l}{c} P \cdot Q$$

Thus, we can infer an expression of $P \cdot P'$ and $Q \cdot P'$. In first hand, $P \cdot P'$ corresponds to the number of elements in common between P and P', which is the size of the view minus the number of new element inserted. On the other hand, $Q \cdot P'$ corresponds to the number of common elements between Q and P', which is the size of the sent view (inserted for sure in V_i after the shuffle) plus the unsent elements that were in common with P and Q. Then, we obtain:

$$\begin{cases} P \cdot P' &= c - l + \dfrac{l}{c} \times P \cdot Q \\ &= l\left(\dfrac{P \cdot Q}{c} - 1\right) + c \\ \\ Q \cdot P' &= l + \left(1 - \dfrac{l}{c}\right) P \cdot Q \\ &= l\left(1 - \dfrac{P \cdot Q}{c}\right) + P \cdot Q \end{cases}$$

In the proof of Lemma 5.1, we have proved that $M_{ij} \leq c$. So, we have $\frac{P \cdot Q}{c} \leq 1$. Given a fixed view size c, $P \cdot P'$ (resp. $Q \cdot P'$) is then a linear function of l with a negative (resp. positive) slope, and a range from 0 to $P \cdot Q$ (resp. from $P \cdot Q$ to 0) *cf.* Figure 3. Then, the effectiveness is equal to $P \cdot P'$ for small value of l and to $Q \cdot P'$ for large value of l. Thus, the greatest value of $\mathcal{E}$ is obtained for:

$$\begin{aligned} & P \cdot P' = Q \cdot P' \\ \iff & l\left(\frac{P \cdot Q}{c} - 1\right) + c = l\left(1 - \frac{P \cdot Q}{c}\right) + P \cdot Q \\ \iff & l\left(2 \cdot \frac{P \cdot Q}{c} - 2\right) = P \cdot Q - c \\ \iff & 2 \cdot l \cdot \frac{P \cdot Q - c}{c} = P \cdot Q - c \\ \iff & l = \frac{c}{2}. \end{aligned}$$

The greatest effectiveness is then reached for $\gamma = \dfrac{1}{2}$. □

5.5. Providing a sample stream

Until now, we have proved that the shuffle protocol makes the local view of all nodes to converge to the uniform distribution, that is all IDs appear in any view with the same probability. We want now elaborate about the ability of the protocol to provide a continuous stream of independent samples[4], contrary to [9] that ever provide the same one after convergence.

Let S and S' be two consecutive samples returned by the same node. By definition, for the new sample to be independent from the previous one, for any pair s, s', it must be

$$\mathbb{P}[S' = s' | S = s] = \mathbb{P}[S' = s].$$

[4] *aka* the ergodic property defined in [11]

As the returned sample is taken from the node view, it is clear there that is a high probability that the previous sample is returned back again, *i.e.*, $s' = s$. One possible solution for the independency property to hold is to force the sampler to provide a new sample, only after the previously returned one has been selected for being sent out during a shuffle operation (s in our example). In this way, s is replaced by another uniformly chosen ID. Note that this doesn't mean that s cannot be returned two consecutive times. In fact, the partner node for one of these shuffle operations can send s back again.

In order to estimate the independence of two consecutive samples, we consider the notion of ε-independent view: the view obtained after a sequence of shuffle operations is ε-independent from the initial one if the probability that the former returned sample s has not been sent out once, is at most ε.

We now seek for the minimum number κ of shuffling operations required to obtain ε-independence. The following lemma provides a lower-bound of κ, given a ε:

Lemma 5.7. *Given ε, the minimum number of shuffling operations κ required to obtain a ε-independent view is*

$$\kappa \geq \frac{\log \varepsilon}{\log(1-\gamma)}$$

Proof. As shuffling operations are pairwise independent in term of sent vector selection, the probability that after κ shuffle operations s has been refreshed is given by

$$\sum_{k=1}^{\kappa} \gamma\,(1-\gamma)^{k-1}\,.$$

In fact, the probability to send s exactly at the k-th operation is $\gamma\,(1-\gamma)^{k-1}$, as γ corresponds to the probability that s is selected in the sent view.

Then, the minimum number κ of shuffle operations such that $\sum_{k=1}^{\kappa} \gamma\,(1-\gamma)^{k-1} \geq 1-\varepsilon$ (*i.e.*, s has been sent out at least once during κ operations with a probability greater than $1-\varepsilon$) can be obtained exploiting a geometric series. Recall that $\gamma < 1$ and $\varepsilon < 1$. We thus obtain:

$$\gamma\frac{1-(1-\gamma)^{\kappa}}{1-(1-\gamma)} \geq 1-\varepsilon \Leftrightarrow (1-\gamma)^{\kappa} \leq \varepsilon \Leftrightarrow \kappa \geq \frac{\log \varepsilon}{\log(1-\gamma)}$$

that concluded the proof. □

For instance, given $\gamma = 1/4$ and $\varepsilon = 0.0005$, we obtain that sample independence again after $5,5$ operations. In practical terms, after 6 operations the returned sample is refreshed with probability at least 0.9995.

That speaks about temporal independence of samples under a local point of view. In order to provide a global analysis, based on correlated spatial and temporal independence, we should also consider the global evolution of the system configuration (*cf.* [10] for more details). How the shuffling operations affect the spatial dependency remains an open question for future works.

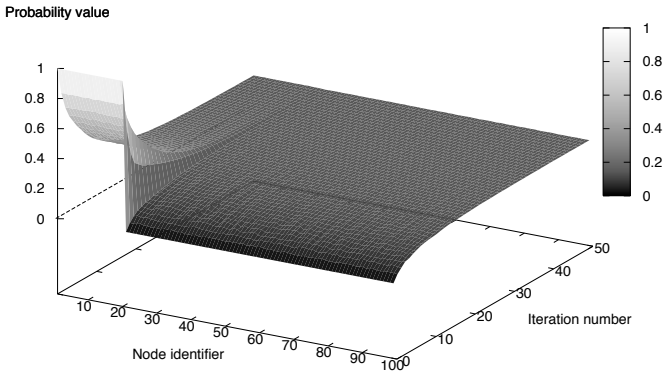


Figure 4: Evolution of $\mathbb{P}[X_{ji} = 1]$ for i fixed according to the gossip cycle iteration.

Settings: $n = 100$, $c = 20$ and $l = 4$ for 50 iterations.

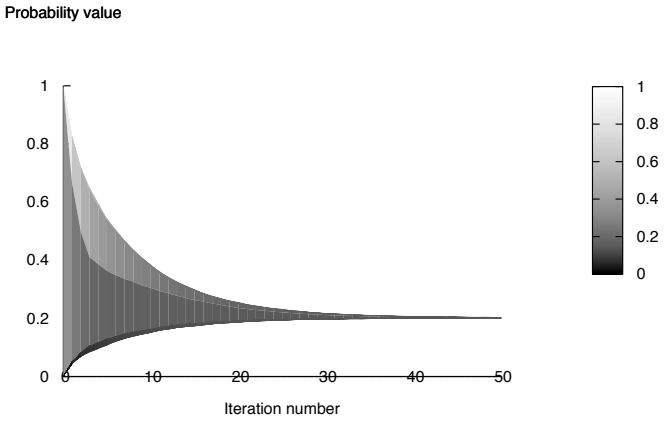


Figure 5: Evolution of $\mathbb{P}[X_{ji} = 1]$ for i fixed according to the gossip cycle iteration (Planar view).

Settings: $n = 100$, $c = 20$ and $l = 4$ for 50 iterations.

6. Numerical results

In this section, we apply the analytical model (*cf.* Equation 3) in order to numerically derive some representative evolutions of a system, in which shuffles are organized into cycles. One cycle corresponds to all nodes initiate exactly one shuffle with a random partner chosen from its own view[5].

Consider a system with view size $c = 20$. Initially, the views of nodes are set to [1..20]. This corresponds to one of the worst cases of starting state. Indeed, among a population of 100 nodes, the identifiers [21..100] do not appeared in any view at starting point. They will be introduced progressively by the initiator using the biased shuffling operation, as explained in Section 3.

Figure 4 shows the view evolution of one node. The z-axis shows the probability that an ID appears in the view of this node. At the beginning of an execution, the overlay is then

[5]This cycle-based behavior is well-known in gossip-based protocols [1, 6, 7, 8, 13].

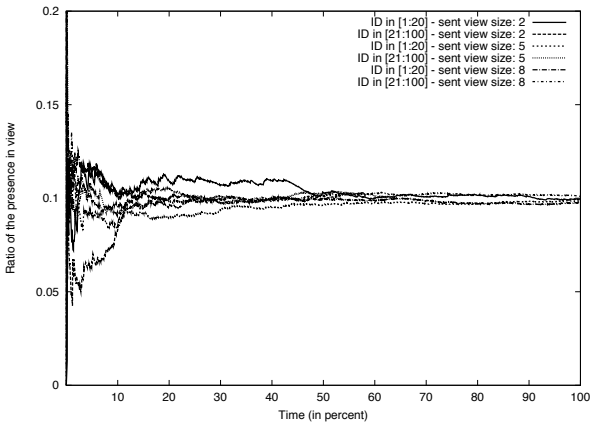


Figure 6: Duration of presence in a view V of a node, initially located or not in all views, for different ℓ size.

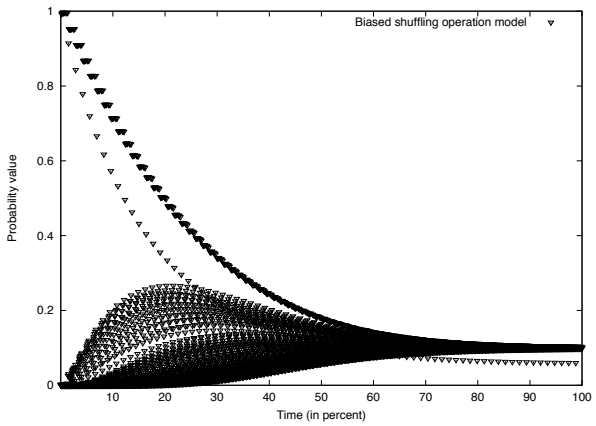


Figure 7: Duration of presence in a view V of a node, initially located or not in all views, for different ℓ size.

fixed: c nodes have a probability equals to 1 to appear in a view and the other nodes a probability equals to 0. When the protocol runs, all nodes are proceeding to theirs shuffles during each cycle. Figures 4 shows the evolution of each probability $\mathbb{P}[X_{ji} = 1]$ according to the node identifier j and the iteration of the algorithm, where one iteration corresponds to one gossip cycle. Figures 5 represents the same data but in a planar view (all the probabilities of each time are mapped vertically). Thus, the latter figure shows the evolution of the maximum (and *a fortiori* the minimum) probability value at a given time. It is possible to observe that, as expected, all the probabilities converge to the average value ($\frac{c}{n} = 0.2$) in less than 40 gossip cycles.

On the other hand , we want to focus on another metric which must reach also uniformity. In fact, by simulate the biased algorithm from the same initial system state, we observed the duration of each *inter-inview time* (IIT); this correspond to the length of the period between two times a given node belongs to the view of a specific node. In a uniformly random behavior, this IIT must be equal to $\frac{c}{n}$. Figure 6 presents, for a view equals to 10 and different sent view sizes, the evolution of the IIT according to time. One can observe on this figure that the IIT oscillate around very different values just after the starting time (due to the unbalanced representation of each node in all view, in the aforementioned initial state). However, we could observe that the protocol makes these IIT converge to 0.1, which is exactly $\frac{c}{n}$ in our settings.

For the comparison purpose, we model the evolution of the system according to Equation 3, for both *basic* and *biased* shuffling operation. As in Figure 5, but according to time and not to cycle iteration, we present the probability vector evolution of one view, in Figure 7. The latter speaks of the equivalence of both approaches in term of convergence speed. Moreover, we can observe that the biased operation could lead to decrease a bit faster the very high probability. This is due to the fact that each shuffle initiator node input its own ID in the sent view, and then, refresh the knowledge of its ID faster than the basic approach.

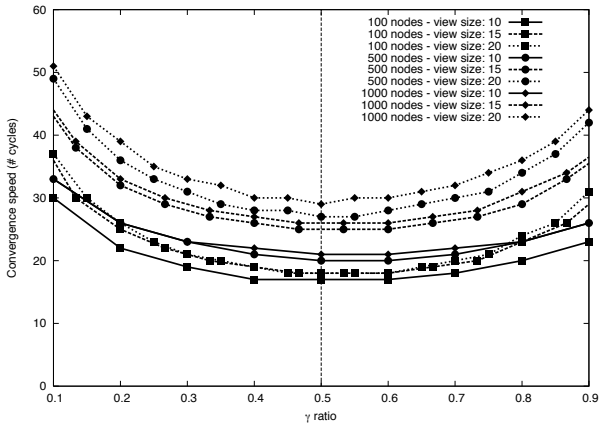


Figure 8: Number of gossip cycles required to reach uniformity for several settings, according to the ratio γ.

Last but not least, in order to evaluate the impact of the system parameters, Figure 8 presents the average convergence time required to reach the uniform sampling, according to the ratio $\gamma = \frac{l}{c}$, for different settings of the system (from 100 to 1,000 nodes with a view size varying from 10 to 20), starting from the same aforementioned worst case. This figure speaks about how to obtain the best convergence time according to γ. Independently of the size of the network, the size of the view c and the initial state, the fastest convergence is obtained with a ratio $\gamma = 0.5$ (represented by a vertical line on Figure 8). Thus, in the design of a gossip-based protocol, l has to be set to the half of c in order to obtain the highest efficiency in term of convergence speed, as formally shown in section 5.4 above.

Note that this is not an end in itself. Actually, one could suffice with a convergence speed that is marginally slower than the optimal one but with $\gamma = 0.25$. In fact, the advantage is

that a smaller lambda means smaller messages and bandwidth consumed.

7. Related works

The most relevant papers related to our work are [9, 10]. Our work differs from the one described in [9] in that while in that work a random sample request triggers a new shuffling protocol, our protocol allows for obtaining a continuous flow of random samples (more precisely, a safe condition for keeping uniform the sample returned by a node is to engage a new shuffle operation for this node that includes the ID returned by the last sample, so that the ID is "refreshed". As the optimal number of IDs exchanged in a shuffle operation is proved to be an half of the local view, the returned ID is refreshed after just two shuffle operations, in expectation. Hence, continuous uniformity is sustained by simply setting the shuffle rate to at least twice the sample rate). This *ergodicity* of the returned samples has recently been characterized in presence of Byzantine nodes [11]. This paper shows the intrinsic limitation of any sampling method in presence of malicious node, and defines some required assumptions on the system to make it possible. The work described in [10] uses ergodic view sample with duplicate nodes allowed to appear in a view. Moreover, the system model is different from our as it is based on asymmetric one-way communication among peers. We extend the comparison with this paper below.

Apart of the paper presented in [9, 10], several contributions have been proposed in the context of gossip-based peer sampling service [1, 8]. In these works, authors have studied the same framework used in this paper (as known as gossip-based protocol). However, they only provide empirical results. Using the same experimental approach, PuppetCast [14] is a protocol for ergodic and uniform peer sampling in large-scale distributed systems that tolerates up to 50 % of the nodes are acting maliciously. To the best of our knowledge, no fully theoretical analysis of the shuffling protocol with respect to sampling uniformity has been proposed so far.

It is worth remarking the key differences with the work described in [10]. The analysis provided in that paper is based on a simple push operation that sends one link endpoint and reverses the another one. On the contrary, we consider two-way symmetric exchange operations. The view model of [10] is based on a multiset where empty slots are allowed, whereas we consider fully filled views with no duplications.

Furthermore, in order to obtain the spatial independence, study of [10] restricts the range of possible initial state, whereas we proof that we converge to the uniformity from any initial state.

Finally, although the one-way communication paradigm inherently faces message loss, we could deal with losses as outlined in Section 3.

Several contributions provided some fully theoretical analysis of gossip-based protocols as [13, 1, 15, 16, 17]. However, those analysis aims to provide some theoretical outcomes on a specific characteristic of these protocols as convergence speed of dissemination protocols, by defining precise lower and upper bound of the mixing time, degree balancing, *etc.* Nevertheless, in these works, except for [13], authors do not consider the local view as the information to analyze. In their works, the network is modeled as a probabilistic matrix, which represents the meeting probability of any pair of peers, and this matrix is used as a building block of their analyses. Our study can then be used to provide this specific matrix and/or to confirm that the matrix used in these related works are consistent with the real behavior of gossip-based protocols.

On the other hand , Bakhshi *et al.* propose in [13] an analytical model of shuffling protocols. However, as they are interested in the characteristics of the dissemination of data items, they investigate two other properties than ours that characterized these protocols : *(i)* the number of replicas of an item in the network, and *(ii)* the coverage achieved by an item over time. Nevertheless, we can parallel our work by considering node *IDs* as the disseminated data. Despite the fact that their analysis relies on a close approximation scale down by a correction factor, it is interesting to note that authors of [13] found the same optimum value of $\gamma \sim 0.5$.

Another set of theoretical approaches has to be raised [18, 19, 20]. These works provides some decentralized method to generates random graphs, based on edge-flipping. Initially, Mahlmann and Schindelhauer have proposed *Random k-Flipper* [18], a graph transformation algorithm that creates random regular connected undirected graphs, asymptotically almost surely uniform. They have analyzed the cost of maintenance operation for a P2P network, based on random regular connected graphs. Following the same approach, Cooper *et al.* show in [19] that performing random flip operations on a regular graph samples almost uniformly at random, in time polynomial in graph size and error value. Both contributions present interesting theoretical results according the convergence time of such decentralized sampling method, that could draw a parallel between our contribution. Additionally, in the context of weakly connected multi-digraphs with regular out-degree, authors of [18] has proposed a local graph transformation, namely Pointer-Push&Pull operation [20] that produces any of these graphs uniformly at random after a series of random operations. The latter's definition is attractive because, as in [10], it consists in exchanging only two neighboring edges. Their analysis fits with our work, in the specific ($l = 1$)-case.

Finally, as remarked in Section 1, random walks have been also used to provide uniform peer sampling [3, 21]. These contributions proposed how to bias the simple random walks model in the way to extract uniform sampling. Both of them provide a theoretical analysis of their protocols. Finally, a solution of the peer sampling service, based on a structured P2P system, has been proposed in [22]. Authors propose an algorithm based on Chord [23] and proved that it provides nodes with uniform random samples of the system.

8. Concluding Remarks

The paper has provided a theoretical ground to the fact that a shuffling protocol provides eventually nodes with uniform ran-

dom samples of a system. Before this was only an empirical evidence. Differently from [9], our analysis shows that the same instance of the shuffling protocol can provide permanently a node with uniform sample of the system. Corollary 1 formally grasps this difference.

On the other hand , future works consist in a deeper analysis of our shuffling operation, as the impact of the topological distribution, or thorough proofs of properties as spatial and temporal independence (as done in [10] for a different model).

The paper also presented a numerical evaluation of the shuffling algorithm on its convergence speed of the local views to uniform random samples. We also formally proved what is the best fraction of the local views to swap in a shuffling operation to get best convergence speed.

Acknowledgments

The authors would like to warmly thank Leonardo Querzoni for his help with the simulations. The authors are indebted with the reviewers for their comments and suggestions that greatly improved presentation and content of the paper. A short and preliminary version of this paper [24] appeared at the 10th International Conference on Parallel and Distributed Computing Applications and Technologies.

References

[1] M. Jelasity, S. Voulgaris, R. Guerraoui, A.-M. Kermarrec, M. van Steen, Gossip-based peer sampling, ACM Transaction on Computer System 25 (3) (2007) 8.

[2] R. Baldoni, R. Beraldi, V. Quema, L. Querzoni, S. Tucci-Piergiovanni, TERA: topic-based event routing for peer-to-peer architectures, in: the 2007 inaugural international conference on Distributed Event-Based Systems (DEBS '07), ACM, Toronto, Ontario, Canada, 2007, pp. 2–13. doi:http://doi.acm.org/10.1145/1266894.1266898.

[3] L. Massoulié, E. L. Merrer, A.-M. Kermarrec, A. Ganesh, Peer counting and sampling in overlay networks: Random Walk Methods, in: the 25th annual ACM symposium on Principles of distributed computing (PODC '06), ACM Press, Denver, CO, USA, 2006, pp. 123–132. doi:http://doi.acm.org/10.1145/1146381.1146402.

[4] R. Baldoni, A. Corsaro, L. Querzoni, S. Scipioni, S. T. Piergiovanni, Coupling-based internal clock synchronization for large-scale dynamic distributed systems, IEEE Transactions on Parallel and Distributed Systems 99 (RapidPosts) (2009) 607–619. doi:http://doi.ieeecomputersociety.org/10.1109/TPDS.2009.111.

[5] B. Bollobás, Random Graphs – 2nd Edition, Cambridge University Press, Cambridge, UK, 2001.

[6] P. T. Eugster, S. Handurukande, R. Guerraoui, A.-M. Kermarrec, P. Kouznetsov, Lightweight probabilistic broadcast, ACM Transactions on Computer Systems 21 (4) (2003) 341–374. doi:http://doi.acm.org/10.1145/945506.945507.

[7] M. Jelasity, A. Montresor, O. Babaoglu, T-Man: Gossip-based Fast Overlay Topology Construction, Computer Networks 53 (13) (2009) 2321–2339.

[8] S. Voulgaris, D. Gavidia, M. van Steen, CYCLON: Inexpensive Membership Management for Unstructured P2P Overlays, Journal of Network System Management 13 (2) (2005) 197–217.

[9] E. Bortnikov, M. Gurevich, I. Keidar, G. Kliot, A. Shraer, Brahms: byzantine resilient random membership sampling, in: the 27th ACM symposium on Principles of Distributed Computing (PODC '08), ACM, Toronto, Canada, 2008, pp. 145–154. doi:http://doi.acm.org/10.1145/1400751.1400772.

[10] M. Gurevich, I. Keidar, Correcness of Gossip-Based Membership under Message Loss, in: the 28th annual ACM Symposium on Principles of distributed computing (PODC '09), ACM Press, Calgary, AL, Canada, 2009, pp. 151–160.

[11] E. Anceaume, Y. Busnel, S. Gambs, Uniform and ergodic sampling in unstructured peer-to-peer systems with malicious nodes, in: the 14th International Conference On Principles Of Distributed Systems (OPODIS 2010), Tozeur, Tunisia, 2010, pp. 1–15.

[12] E. W. Dijkstra, Cooperating sequential processes, in: F. Genuys (Ed.), Programming Languages: NATO Advanced Study Institute, Academic Press, 1968, pp. 43–112.

[13] R. Bakhshi, D. Gavidia, W. Fokkink, M. van Steen, An analytical model of information dissemination for a gossip-based protocol, Computer Networks 53 (13) (2009) 2288–2303.

[14] A. Bakker, M. van Steen, Puppetcast: A secure peer sampling protocol, Computer Network Defense, European Conference on 0 (2008) 3–10. doi:http://doi.ieeecomputersociety.org/10.1109/EC2ND.2008.7.

[15] R. Karp, C. Schindelhauer, S. Shenker, B. Vocking, Randomized rumor spreading, in: the 41st Annual Symposium on Foundations of Computer Science (FOCS '00), IEEE Computer Society, Washington, DC, USA, 2000, p. 565.

[16] D. Kempe, A. Dobra, J. Gehrke, Gossip-Based Computation of Aggregate Information, in: the 44th Annual IEEE Symposium on Foundations of Computer Science (FOCS '03), IEEE Press, Cambridge, MA, USA, 2003, pp. 482–491.

[17] S. Boyd, A. Ghosh, B. Prabhakar, D. Shah, Randomized gossip algorithms, IEEE/ACM Transaction on Networks 14 (SI) (2006) 2508–2530. doi:http://dx.doi.org/10.1109/TIT.2006.874516.

[18] P. Mahlmann, C. Schindelhauer, Peer-to-peer networks based on random transformations of connected regular undirected graphs, in: Proceedings of the seventeenth annual ACM symposium on Parallelism in algorithms and architectures (SPAA '05), ACM, New York, NY, USA, 2005, pp. 155–164. doi:http://doi.acm.org/10.1145/1073970.1073992.

[19] C. Cooper, M. Dyer, A. J. Handley, The flip markov chain and a randomising p2p protocol, in: Proceedings of the 28th ACM symposium on Principles of distributed computing (PODC '09), ACM, New York, NY, USA, 2009, pp. 141–150. doi:http://doi.acm.org/10.1145/1582716.1582742.

[20] P. Mahlmann, C. Schindelhauer, Distributed random digraph transformations for peer-to-peer networks, in: Proceedings of the eighteenth annual ACM symposium on Parallelism in algorithms and architectures (SPAA '06), ACM, New York, NY, USA, 2006, pp. 308–317. doi:http://doi.acm.org/10.1145/1148109.1148162.

[21] M. Zhong, K. Shen, J. Seiferas, Non-uniform random membership management in peer-to-peer networks, in: in 24th Annual Joint Conference of the IEEE Computer and Communications Societies (INFOCOM '05), Vol. 2, IEEE Press, Piscataway, NJ, USA, 2005, pp. 1151–1161 vol. 2. doi:10.1109/INFCOM.2005.1498342.

[22] V. King, J. Saia, Choosing a random peer, in: the 23rd annual ACM symposium on Principles of Distributed Computing (PODC '04), ACM, New York, NY, USA, 2004, pp. 125–130. doi:http://doi.acm.org/10.1145/1011767.1011786.

[23] I. Stoica, R. Morris, D. Liben-Nowell, D. R. Karger, M. F. Kaashoek, F. Dabek, H. Balakrishnan, Chord: a scalable peer-to-peer lookup protocol for internet applications, IEEE/ACM Transaction on Networks 11 (1) (2003) 17–32. doi:http://dx.doi.org/10.1109/TNET.2002.808407.

[24] Y. Busnel, R. Beraldi, R. Baldoni, A formal characterization of uniform peer sampling based on view shuffling, in: the 10th International Conference on Parallel and Distributed Computing Applications and Technologies, IEEE Computer Society, Los Alamitos, CA, USA, 2009, pp. 360–365. doi:http://doi.ieeecomputersociety.org/10.1109/PDCAT.2009.61.