PEERSWAP: A Peer-Sampler with Randomness Guarantees

Rachid Guerraoui*, Anne-Marie Kermarrec*, Anastasiia Kucherenko*, Rafael Pinot†, Martijn de Vos*
* *Ecole Polytechnique Fédérale de Lausanne (EPFL), Lausanne, Switzerland*
{rachid.guerraoui, anne-marie.kermarrec, anastasiia.kucherenko, martijn.devos}@epfl.ch
† *Sorbonne Université, Paris, France,* pinot@lpsm.paris

***Abstract*—The ability of a peer-to-peer (P2P) system to effectively host decentralized applications often relies on the availability of a peer-sampling service, which provides each participant with a random sample of other peers. Despite the practical effectiveness of existing peer samplers, their ability to produce random samples within a reasonable time frame remains poorly understood from a theoretical standpoint. This paper contributes to bridging this gap by introducing PEERSWAP, a peer-sampling protocol with provable randomness guarantees. We establish execution time bounds for PEERSWAP, demonstrating its ability to scale effectively with the network size. We prove that PEERSWAP maintains the fixed structure of the communication graph while allowing sequential peer position swaps within this graph. We do so by showing that PEERSWAP is a specific instance of an interchange process, a renowned model for particle movement analysis. Leveraging this mapping, we derive execution time bounds, expressed as a function of the network size n. Depending on the network structure, this time can be as low as a polylogarithmic function of n, highlighting the efficiency of PEERSWAP. We implement PEERSWAP and conduct numerical evaluations using regular graphs with varying connectivity and containing up to $32\,768$ (2^{15}) peers. Our evaluation demonstrates that PEERSWAP quickly provides peers with uniform random samples of other peers.**

***Index Terms*—Gossip-based Peer Sampling, Convergence of Peer Sampling, Interchange Process.**

I. INTRODUCTION

Peer-to-peer (P2P) systems are decentralized networks in which users, or *peers*, communicate only directly with each other using peer identifiers, *e.g.*, public keys or network addresses [1]. Many of the applications executed on a P2P system rely on so-called *peer sampling services* that provide each peer with a random sample of other peers [2]. Such applications include computing aggregate functions [3], information dissemination [4], [5], network size counting [6], [7], consensus [8], clock synchronization [9], maintaining overlay networks [10], [11] and decentralized learning [12].

P2P applications critically depend on the capability of peer-sampling services to offer samples randomly and uniformly selected from the overall pool of peers. This randomness is, for example, essential for facilitating rapid convergence in averaging tasks, improving model accuracy in decentralized learning, and faster propagation of information through the network. Depending on the application, even a small bias in the randomness of the peer sampling service can lead to significant performance degradation at the application layer, *e.g.*, resulting in uneven load balancing between peers, delays in information propagation, or suboptimal model convergence in decentralized learning [13].

Over the past two decades, various methods to construct peer sampling services have been developed, falling into two main categories: random-walk based and gossip-based protocols. Random-walk based peer samplers, *e.g.*, [6], [14]–[18], were used in early P2P systems like Gnutella [19] and, more recently, in Bitcoin [20]. Essentially, the idea is to execute a random walk through the network and use the final peer encountered as the random sample. However, random walk-based peer samplers have two main limitations: *(i)* they incur a high network overhead if many samples are required [21], and *(ii)* they tend to favor nodes with high in-degrees, which compromises their ability to provide a uniform sample [6].

Gossip-based peer samplers, *e.g.*, [21]–[23], have significantly evolved in recent years, successfully overcoming the aforementioned limitations. The essence of the gossip-based approach involves each peer maintaining a small subset of other peers, known as a *neighborhood*. Periodically, each peer exchanges a subset of its neighborhood with another peer. The simplicity and rapid provision of new peer samples have driven the popularity of gossip-based peer samplers. Several works extend gossip-based peer samplers to support peers joining and leaving the network (churn) [24], or to handle the presence of malicious peers [25]–[30].

In a gossip-based peer sampler, each peer's neighborhood strongly correlates with its initial neighborhood during the first few exchange rounds. As the neighborhood of each peer undergoes regular updates, the distribution of potential peers in a fixed peer's neighborhood approaches uniformity, eventually providing an almost ideally random sample from the entire network. This *convergence to uniformity* is crucial, enabling P2P applications to rely on the randomness of peer samples. Empirical evidence conveys convergence to almost uniform samples within a small, often logarithmic number of rounds in terms of the total number of peers [21], [22], [31].

Despite their excellent practical performance, existing peer-sampling services lack a comprehensive theoretical analysis of their randomness guarantees and convergence time. While some progress has been made in proving the *eventual uniformity* of samples [30]–[32], determining the time required for peer samplers to achieve uniformly random samples remains an open challenge. Without clearly defining the convergence time, *i.e.*, the time needed to provide a random sample to each

peer, the exact performance and guarantees of peer sampling services remains unknown.

Analyzing the randomness of peer samplers is a challenging endeavor, primarily due to the absence of explicit theoretical tools to analyze peers' neighborhoods in dynamic networks. One approach is to demonstrate the convergence of the entire network to a random graph, thereby asserting the randomness of each neighborhood. However, this method lacks reasonable guarantees regarding the convergence time. For instance, while eventual convergence to a random graph has been demonstrated in [31], specific time guarantees are absent. A similar approach is also explored in the flip process [33], [34], a substantial area in graph theory. The flip process is equivalent to a gossip-based peer sampler where peers exchange exactly one neighbor at each step. The best-known result for the convergence time needed for the network topology to resemble a random graph is a large polynomial, around $O(n^{16})$ where n is the number of peers [35]. This is an impractical result in large-scale networks where this convergence time will be disproportionally large. Unfortunately, a specific analysis of neighborhood characteristics instead of the network graph is difficult because of complicated network dynamics.

Our contributions. This paper is a first step towards bridging the gap between practical advancements and theoretical guarantees of gossip-based peer samplers. Concretely, we make the following contributions:

1) We introduce PEERSWAP (Section III), a gossip-based peer-sampling service that *provably and efficiently provides a random sample for each peer*. Similarly to other gossip-based peer samplers, PEERSWAP has peers randomly contacting each other and exchanging neighbors. In contrast to existing approaches, PEERSWAP *(i)* has peers exchanging their entire neighborhoods instead of subsets; *(ii)* maintains bidirectional connections for all peers; *(iii)* uses Poisson clocks on each connection to schedule regular neighborhood exchanges (every Poisson clocks periodically rings after random time intervals to trigger a neighborhood exchange).
2) We prove that the design of PEERSWAP provides an important property throughout the execution, namely that the overall *connections network maintains the same underlying pattern as the initial one* despite changes in neighborhoods and connections (Section IV). This property serves a dual purpose. Firstly, it guarantees that any network property satisfied at the start of the algorithm is preserved throughout its execution, *e.g.*, the network connectivity, expansion value, degree distributions, and diameter. This can be essential for applications whose performance critically relies on the communication patterns between peers, like decentralized learning [12]. Secondly, combined with the behavior of Poisson clocks, this property enables a comprehensive analysis of the convergence time of PEERSWAP.
3) By establishing a link between the dynamics of peers in PEERSWAP and an interchange process, an important and well-studied interacting particle system [36]–[38], we derive the convergence time of PEERSWAP (Section V). The upper bound we deduce depends only on network size (n) and its connectivity (measured by the spectral gap). We show that in sufficiently connected networks *the convergence time is as low as polylogarithmic in* n.
4) We design a variant of the PEERSWAP algorithm that temporarily locks peers involved in a neighborhood exchange. This variant can be deployed in practical settings with network delays (Section VI).
5) We conduct numerical evaluations of PEERSWAP using regular graphs with varying connectivity and with up to $32\,768$ (2^{15}) peers (Section VII). Our evaluation conveys the convergence of PEERSWAP over time and empirically demonstrates that PEERSWAP is an efficient peer sampler. We also analyze the convergence and performance of our practical variant of PEERSWAP.

The full version of the paper can be found online [39].

II. SYSTEM MODEL AND BACKGROUND

Preliminaries on graph theory. A directed graph G consists of a set of nodes V and edges E where each edge (i, j) goes from node $i \in V$ to node $j \in V$. When $(i, j) \in E$, we say that node j is adjacent to node i. We denote by A_G the adjacency matrix of the directed graph G. For each node $i \in V$, *the neighborhood of* i denoted by $N(i)$ is a set of all nodes which are adjacent to i. The degree of i, sometimes called the out-degree and denoted by $\deg(i)$, is the size of its neighborhood, or $|N(i)|$. If all nodes have the same degree d, the graph is called $d-$regular. If, for any pair of nodes $i, j \in V$, we have that $i \in N(j)$ if and only if $j \in N(i)$, then we say that *the graph G is undirected*. We use the previous notations similarly for undirected graphs.

An important measure of graph connectivity is its spectral gap. To define it, consider D, the $n \times n$ matrix with all elements equal to zero, except of diagonal $(\deg(i_1), \ldots, \deg(i_n))_{i_k \in V}$. *The spectral gap* of the graph G is:

$$\lambda(G) = \lambda(W_G) = 1 - \max(|\lambda_2(W_G)|, |\lambda_n(W_G)|),$$

where $1 = \lambda_1(W_G) \geq \lambda_2(W_G) \geq \cdots \geq \lambda_n(W_G) \geq -1$ are eigenvalues of $W_G = D^{-1/2} A_G D^{-1/2}$. *A higher spectral gap λ indicates a better-connected graph.*

A. Peer-sampling

Peers network. We consider a network with n fixed peers $U = \{u_1, \ldots, u_n\}$, where each peer is assigned a unique identifier. We refer both to a peer i and its identifier as u_i. Peers are connected over a routed network infrastructure, which enables communication between any pair of them on the condition that the sender knows the identifier of the receiver. We assume that all peers have access to a global clock and that time is *continuous*, allowing peers to send messages or take actions at any moment. Additionally, there are no failures in the system – all peers remain online during protocol execution. These assumptions align with related theoretical work on peer sampling [6], [16], [18], [31].

Network Topology. To describe the network structure at time t, we consider a time-dependent directed graph $G(t) = (V, E(t))$. Here, the set of nodes $V = [1, 2, \ldots, n]$ represents the fixed set of peers U — any node $i \in V$ corresponds to a peer $u_i \in U$, and there is a directed edge $(i, j) \in E(t)$ if and only if the peer u_i knows the identifier of the peer u_j at the time t. We denote by $N(i, t)$ the (ordered) neighborhood of i in graph $G(t)$ for any $i \in V, t \geq 0$. Although peer samplers typically operate within directed networks, we show in Section II-C that in our setting, $G(t)$ remains undirected over time.

Peer Sampling Service. A peer-sampling service with parameter b on the set of peers U is a decentralized communication protocol that, at any time $t \geq 0$, provides each peer $u_i \in U$ with a random ordered tuple of distinct peers $Sample_i(\mathrm{b}, t) = (u_{i_1}, u_{i_2}, \ldots, u_{i_\mathrm{b}}) \subseteq U \backslash \{u_i\}$ such that $\{i_1, i_2, \ldots, i_\mathrm{b}\} \in N(i, t)$.

The objective of such a service is to make the distribution of $Sample_i(\mathrm{b}, t)$ gradually approach a uniform distribution. Specifically, we want the random variable $Sample_i(\mathrm{b}, t)$ to converge in law (as $t \to \infty$) to $Uniform((U)_\mathrm{b})$, where for any finite set S and any $\mathrm{b} \leq |S|$ we denote by $(S)_\mathrm{b}$ the set of all (ordered) b-tuples of distinct elements from S:

$$(S)_\mathrm{b} = \{\mathbf{s} \in S^\mathrm{b} : \mathbf{s}[i] \neq \mathbf{s}[j],\ \forall i, j \in [\mathrm{b}], i \neq j\}.$$

Here and further the elements of $(S)_\mathrm{b}$ are denoted by boldface letters such as $\mathbf{x}$, with $\mathbf{x}[i]$ denoting the i-th coordinate of $\mathbf{x}$.

Timing Mechanism. Gossip-based peer samplers require a timing mechanism that controls when a peer will exchange its neighbors with another peer. For this purpose, we use *Poisson clocks* that ring periodically after intervals that follow an exponential distribution. We formally define this below.

A random variable Z has exponential distribution $\mathcal{E}(\alpha)$ with rate α if its cumulative distribution function is:

$$F(z, \alpha) = \mathbb{P}[Z \leq z] = \begin{cases} 1 - e^{-\alpha z}, & \text{if } z \geq 0, \\ 0, & \text{if } z < 0. \end{cases}$$

Importantly, an exponential random variable Z is *memoryless*, meaning its future behavior is unaffected by passed time.

Definition 1 (Poisson clock)**.** A Poisson clock with rate α is a sequence of random variables $(T_n)_{n \geq 0}$ such that $T_0 = 0$ and for $n \geq 1$, we have $T_n = \sum_{k=1}^{n} Z_k$, where $(Z_k)_{k \geq 1}$ is a sequence of independent exponential variables (i.e., $Z_1, \ldots, Z_n \overset{i.i.d.}{\sim} \mathcal{E}(\alpha)$). The realizations of $(T_n)_{n \geq 0}$ are called *the ring times* of the Poisson clock.

B. Continuous-time Markov Chains

In this work, we propose a peer-sampling process whose future is affected solely by its current state and is independent of its history or the exact value of current time t. This property of the process is called Markov property, that we formally outline next.

Consider a family of random variables $X = (X_t)_{t \geq 0}$ taking values in a finite state space S. X is a *continuous-time Markov chain (ctMC)* if it satisfies the Markov Property:

$$\mathbb{P}[X_{t_n} = i_n \mid X_{t_1} = i_1, \ldots, X_{t_{n-1}} = i_{n-1}] = \mathbb{P}[X_{t_n} = i_n \mid X_{t_{n-1}} = i_{n-1}].$$

for all $i_1, \ldots, i_n \in S$ and any sequence $0 \leq t_1 \leq t_2 \leq \cdots \leq t_n$ of times. The process is *time-homogeneous* if the conditional probability does not depend on the current time, but only on the time interval between observations, *i.e.*,

$$\mathbb{P}[X_{t+h} = j \mid X_h = i] = \mathbb{P}[X_t = j \mid X_0 = i], \quad \text{for any } i, j \in S, h \geq 0. \tag{1}$$

For time-homogeneous ctMC we denote the probability in (1) as *a transition probability* $P_{i,j}(t)$ and define *the matrix of transition probabilities* at time t as $P(t) = (P_{i,j}(t))_{i,j \in S}$. A ctMC is called *irreducible* if for any states $i, j \in S$ there exists $t > 0$ such that $P_{i,j}(t) > 0$. In this paper, if not stated otherwise, we consider irreducible time-homogeneous ctMCs.

The transition probability can be used to completely characterize the ctMC, but it gives a lot of information, which is not always needed. An alternative (and more succinct) way of characterizing the dynamic of a ctMC is the *infinitesimal generator* [40][Chapter 20.1].

Definition 2 (Infinitesimal generator)**.** Let $X = (X_t)_{t \geq 0}$ be a time-homogeneous ctMC with finite set of states S and transition probabilities $P(t)$. The infinitesimal generator of X is the matrix of size $|S| \times |S|$ defined as:

$$Q = \lim_{h \to 0+} \frac{P(h) - I}{h},$$

where I is the identity matrix of size $|S| \times |S|$. We denote entries of Q as $q(i, j)$ for $\forall i, j \in S$. By construction of Q we always have $\sum_{j \in S} q(i, j) = 0, \forall i \in S$.

Definition 3. The *total variation distance* d_{TV} between two distributions[1] μ, ν on the same finite set of states S is given by:

$$d_{TV}(\mu, \nu) = 1/2 \sum_{s \in S} \mid \mu_s - \nu_s \mid, \tag{2}$$

where μ_s (resp. ν_s) denotes the probability of observing s according to μ (resp. ν).

Theorem 1 (Theorem 20.1 from [40])**.** *Consider an irreducible ctMC X on a finite set of states S with matrix of transition probabilities $P(t)$. Then there exists a unique distribution $\pi = (\pi_j)_{j \in S}$ such that $\pi P(t) = \pi$ for all $t \geq 0$, and*

$$\max_{i \in S} d_{TV}(P_{i,\cdot}(t), \pi) \xrightarrow{t \to \infty} 0,$$

where $P_{i,\cdot}(t) = (P_{i,j}(t))_{j \in S}$.

[1]We use a slight abuse of notation by interchangeably referring to a discrete distribution and the vector representing its probabilities.

We call such π *a stationary distribution* of X and define the ε*-mixing time of a ctMC* X as

$$T_X(\varepsilon) = \inf\left\{t \geq 0 : \max_{i \in S} d_{TV}(P_{i,\cdot}(t), \pi) \leq \varepsilon\right\}.$$

The mixing time of ctMCs is a key component in our further derivations of the convergence time of our peer sampler.

C. Auxiliary process of peers' movement on a fixed graph

Peer-sampling services keep the set of peers fixed and modify the connections between them. In this way, graphs $G(t)$ represent the consecutive states of a given "dynamic" graph with fixed nodes and reconnecting edges. Usually, processes on dynamic graphs are complicated to analyze. Thus, in this paper, we map peer-sampling processes on dynamic graphs and an auxiliary so-called interchange process where the graph is *fixed* while peers change positions on the graph's nodes.

Interchange process. An *interchange process* $IP(k, G)$ describes dynamics of k peers on $G = (V, E)$. The set of possible states for this chain is $(V)_k$, *i.e.*, all possible positions of k peers on the nodes of G, where no two peers can be positioned on the same node. The transition between states happens by randomly choosing edges from E, and if an edge $e \in E$ is chosen, the peers (if any) positioned at the endpoints of e are switched. If there is just one peer, it still switches its position to the other endpoint of the edge. We will implicitly assume that G is connected, in which case interchane process is irreducible and thus admits a stationary distribution and mixing time [40][Example 1.12.].

Now, we formally describe an interchange process as per [41]. We define a transposition function for any edge $e = (i, j) \in E$ and node $v \in V$ as

$$f_e(v) = \begin{cases} j, & \text{if } v = i, \\ i, & \text{if } v = j, \\ v, & \text{otherwise.} \end{cases} \tag{3}$$

We also write

$$f_e(\mathbf{x}) = (f_e(\mathbf{x}[i]))_{i \in [1,k]} \text{ for } \mathbf{x} \in (V)_k. \tag{4}$$

An interchange process with parameter k on $G = (V, E)$, or $IP(k, G)$ for short, is a ctMC with state space $(V)_k$ such that for any distinct $\mathbf{x}, \mathbf{y} \in (V)_k$:

$$q(\mathbf{x}, \mathbf{y}) = \begin{cases} 1, & \text{if } \exists e \in E \text{ s.t. } f_e(\mathbf{x}) = \mathbf{y}, \\ 0, & \text{otherwise.} \end{cases} \tag{5}$$

III. Poisson-based random swap (PeerSwap)

We now introduce the PeerSwap protocol. An execution of PeerSwap starts from a set of peers U connected in a d-regular[2] undirected graph $G(0) = (V, E(0))$, where any node $i \in V$ represents a user $u_i \in U$. PeerSwap assigns an independent Poisson clock C_e with rate α to each edge $e \in E(0)$. By default, $\alpha = 1$, and system designers can adjust it to speed up or slow down swap rates based on application requirements (more details in Section VII-D). Each clock C_e is shared between the peers at the endpoints of the edge $e \in E(0)$. If no Poisson clock rings, the graph $G(t)$ remains unchanged as time passes. Whenever a Poisson clock on some edge $e = (i, j)$ rings at time t, peers u_i and u_j swap their position following the four steps visualized in Figure 1 and as described in Algorithm 1:

- **Step 1:** Peer u_i (resp. u_j) sends a Swap message initiating the swap to u_j (resp. u_i). This Swap message contains identifiers of all users u_κ such that $\kappa \in N(i, t)$ (resp. $N(j, t)$), as well as all the Poisson clocks u_i (resp. u_j) shares with its neighbors.
- **Step 2:** Peer u_i sends a Replace$_j$ message to its neighbors $(u_\kappa)_{\kappa \in N(i,t)}$, informing them about the swap and requesting them to replace its identifier, *i.e.*, u_i, with u_j. Similarly, peer u_j also sends a Replace$_i$ message to its neighbors with the identifier of peer u_i.
- **Step 3:** Upon reception of a Swap message, peer u_i (resp. u_j) overwrites its neighborhood and associated Poisson clocks it previously stored with the ones contained in the received message. This step alters the structure of the current graph $G(t)$, as shown in Step 3 of Figure 1 and

Algorithm 1 The PeerSwap algorithm from the perspective of peer u_i.

Require: Undirected $G(0) = (V, E(0))$ describing users U; parameter b; Poisson clocks with rate α $C_e = C_i(j) = C_j(i)$ for $\forall e = (i, j) \in E(0)$.
Ensure: $Sample_i(\mathrm{b}, t) \sim Uniform((U \backslash \{u_i\})_\mathrm{b})$.

upon some C_e rings between i and j at time t **do**
  Set Swap$_i \leftarrow ((u_\kappa, C_i(\kappa))_{\kappa \in N(i,t)})$
  Send Swap$_i$ to u_j ▷ Step 1
  Set Replace$_j \leftarrow u_j$
  Send Replace$_j$ request to all $(u_\kappa)_{\kappa \in N(i,t)}$ ▷ Step 2
end upon

upon receiving Swap$_j$ from u_j at time t **do** ▷ Step 3
  Erase $((u_\kappa, C_i(\kappa))_{\kappa \in N(i,t)})$
  Store $((u_l, C_j(l))_{l \in N(j,t)})$ from Swap$_j$
  ▷ $G(t)$ *changed,* $N(i, t) \leftarrow N(j, t)$
end upon

upon receiving Replace$_l$ from some u_κ **do** ▷ Step 4
  Store u_l and set $C_i(l) \leftarrow C_i(\kappa)$
  Erase $(u_\kappa, C_i(\kappa))$
  ▷ $G(t)$ *changed: in* $E(t)$ (i, κ) *is replaced with* (i, l)
end upon

function $Sample_i(\mathrm{b}, t)$
  At time t select b random peers from $(u_\kappa)_{\kappa \in N(i,t)}$
  Return a randomly ordered tuple of these peers
end function

[2]For presentation clarity, we focus on regular topologies. However, PeerSwap and its analysis are also applicable to irregular graphs.

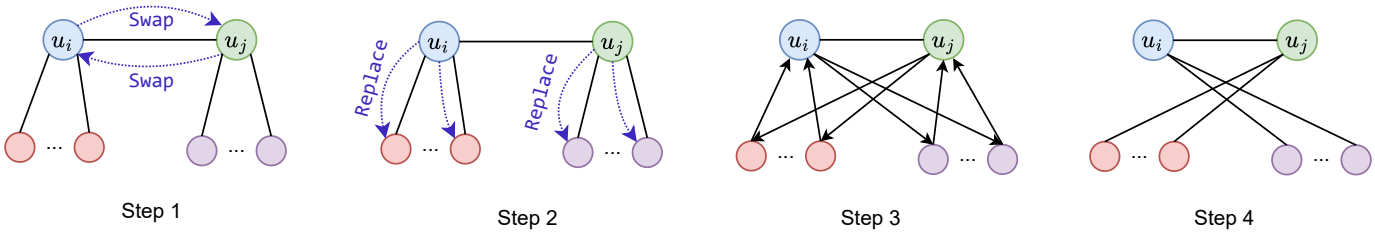


Fig. 1: A single swap during the PEERSWAP execution between a pair of adjacent peers u_i and u_j. Here, blue doted lines indicate sent messages and black solid lines represent connections between peers. We indicate a directed edge with an arrow.

may temporarily make the graph appear undirected.

- **Step 4:** Peers that received a REPLACE$_j$ message from peer u_i replace the identifier of u_i with that of u_j. Also, consider peers that were in the former neighborhood of u_i ($(u_\kappa)_{\kappa \in N(i,t)}$) that now became a neighborhood of u_j. These peers as well replace identifier u_i with u_j, but without modifying the associated Poisson clock.

PEERSWAP is driven by the Poisson clocks placed on the graph edges. Their decentralized implementation is based on shared pseudorandom generators of exponential variables with rate α and is explained in details in Appendix X of the full version of the paper [39]. The rate α indicates that, on average, two peers connected by an edge swap their positions after every time period α. Thus, peers continuously exchange their neighborhoods and get their neighborhoods refreshed. At any point in time $t \geq 0$, a peer u_i can invoke the $Sample_i(\mathrm{b}, t)$ function, yielding a random sample of b peers from the neighborhood of u_i at time t (Lines 15-18 in Algorithm 1).

Remark 1. *When we refer to $G(t)$ we always mean its final configuration. This means that if a Poisson clock rang at time t, $G(t)$ will always refer to the graph after the modification.*

For the upcoming theoretical analysis, we assume that the system has no delays and all actions during the four steps described above are executed instantly. Thus, by design, for any $t \geq 0$, the graph $G(t)$ is always undirected. In Section VI we describe how PEERSWAP can be adapted to function in more realistic settings with network delays.

IV. THE EQUIVALENCE OF PEERSWAP AND THE INTERCHANGE PROCESS

By considering the swap in Figure 1 and changing the graph representation at the end of Step 4 in Figure 2, we observe that running the swap procedure between two peers, *i.e.*, modifying the structure of the graph, is equivalent to having a fixed graph structure on which we exchange the positions of the two considered peers. In this section, we prove that this equivalence holds in general by relating PEERSWAP (which produces a dynamic sequence of graphs $G(t)$) to an interchange process on $G(0)$. This equivalence is the main building block of our analysis and allows us to obtain the convergence result for $Sample_i(b, t)$ in Section V. Complete proofs related to this section can be found in Appendix IX.B of the full version of the paper [39].

A. Isomorphism of graphs built by PEERSWAP

During PEERSWAP process, the network can be described by a sequence of graphs $(G(t))_{t\geq 0}$ where the set of nodes V represents peers (remains fixed), and edges $E(t)$ are modified.

Below, we show that for any $t \geq 0$ their exists an isomorphism between $G(t)$ and $G(0)$.

Definition 4 (isomorphism)**.** Two graphs G and G' are isomorphic according to a bijection $\sigma : V(G) \to V(G')$ if for any two vertices i and j that are adjacent in G, $\sigma(i)$ and $\sigma(j)$ are adjacent in G' (and vice versa).

Lemma 1. *For all $t \geq 0$, $G(0)$ and $G(t)$ are isomorphic according to a bijection $\sigma_t : V(G(0)) \to V(G(t))$. Furthermore, if some nodes i and j are adjacent in $G(t)$, then the Poisson clock they share is C_e with $e = (\sigma_t^{-1}(i), \sigma_t^{-1}(j)) \in E(0)$.*

Now that we know that graphs are isomorphic and that all Poisson clocks of the dynamic graphs $(G_t)_{t>0}$ can be mapped to Poisson clocks on the initial graph $G(0)$, we show that there exists an alternative bijection from $G(t)$ into $G(0)$ defined only by Poisson clocks on $G(0)$. This bijection is easier to operate with and is going to be used for the rest of the paper.

Lemma 2. *Consider PEERSWAP process starting from graph $G(0) = (V, E(0))$, resulting in sequence of graphs $(G(t))_{t\geq 0} = (V, E(t))_{t\geq 0}$. Let us fix some value $t > 0$ and let ring times of Poisson clocks up to time t be $t_1, t_2, \ldots, t_N$ with $0 = t_0 < t_1 < \cdots < t_N \leq t$. Denote $(i_m, j_m) = e_m \in E(t_{m-1})$ the edge on which the clock rang at time t_m, and let $(\hat{\mathrm{i}}_m, \hat{\mathrm{j}}_m) = \hat{\mathrm{e}}_m = (\sigma_t^{-1}(i_m), \sigma_t^{-1}(j_m)) \in E(0)$ be the corresponding edge to $e_m \in E(0)$. We define*

$$\gamma_t = f_{\hat{\mathrm{e}}_N} \circ \cdots \circ f_{\hat{\mathrm{e}}_1},$$

where for every e, f_e is as defined in (3)*.*

Then $\gamma_t = \sigma_t^{-1}$, where σ_t is as defined in Lemma1.

B. Alternative PEERSWAP process and its connection to interchange

Due to the isomorphism of $(G(t))_{t\geq 0}$ and $G(0)$, studying the dynamic of $(G(t))_{t\geq 0}$ is equivalent to studying an auxiliary process where peers move on a fixed graph $G(0)$. Formally this process is described below.

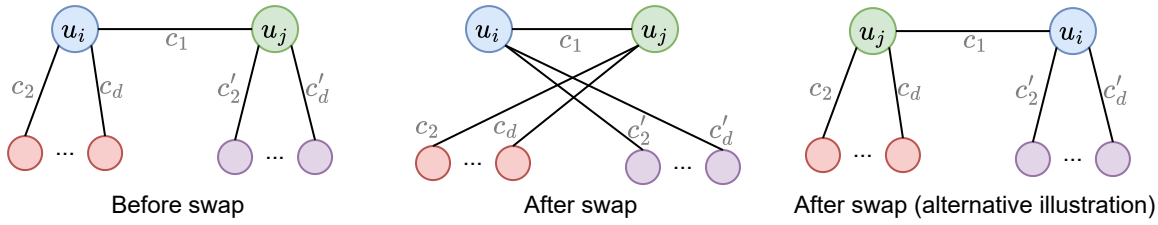


Fig. 2: The network structure before and after a swap in PEERSWAP between a pair of adjacent peers u_i and u_j (as also shown in Figure 1). $c_1, c_2, \ldots, c_d, c'_2, \ldots, c'_d$ correspond to Poisson clocks assigned to the respective edges. The alternative illustration after the swap shows that the general network pattern did not change, but peers u_i, u_j switched their positions.

Definition 5. We study connections between any k distinct peers $(u_{i_1}, u_{i_2}, \ldots u_{i_k}) \subset U$, such that $n \geq k > 0$. To do so, we consider the process $Y^{(k)} = (Y_t^{(k)})_{t\geq 0}$. It has a set of states $(V)_k$, initial state $Y_0^{(k)} = (i_1, i_2, \ldots, i_k)$ and for any $t \geq 0$ we set $Y_t^{(k)} = \gamma_t(Y_0^{(k)})$, where $\gamma_t(\mathbf{x}) = (\gamma_t(\mathbf{x}[i]))_{i\in[1,k]}$ for any $\mathbf{x} \in (V)_k$.

Studying processes $Y^{(k)}$ is an alternative way to describe dynamic of PEERSWAP. *$Y^{(k)}$ operates on a fixed graph $G(0) = (V, E(0))$, and peers U or their subset are switching positions on top of V.*

We show that for any k process $Y^{(k)}$ is a time-homogeneous ctMC and prove that it is an interchange process.

Theorem 2. *Consider* PEERSWAP *process on peers U starting from graph $G(0) = (V, E(0))$, and process $\mathrm{Y}^{(k)}$ characterizing some k peers in* PEERSWAP, *$1 \leq k \leq |V|$ as per Definition 5. Then, $\mathrm{Y}^{(k)}$ is an interchange process $IP(k, G(0))$ (as defined in Section II-C.)*

Proof sketch. We demonstrate that since isomorphisms γ_t rely on Poisson clocks that are memoryless, $\mathrm{Y}^{(k)}$ is a time-homogeneous ctMC. Then, to show that two ctMCs are equivalent, it is sufficient to prove that their infinitesimal generators are equal (as in Definition 2). We calculate the probabilities of zero, one, and more than one Poisson clock ringing in any period of length h. These probabilities allow to derive $\lim_{h\to 0+} \frac{P_{\mathbf{x},\mathbf{y}}(h)}{h} = q(\mathbf{x}, \mathbf{y})$ for any two states $\mathbf{x}, \mathbf{y} \in (V)_k$ and to show the equality with (5). □

V. CONVERGENCE TIME OF PEERSWAP

Since PEERSWAP is tightly connected to an interchange process, we can study the neighborhood of each peer in an execution of PEERSWAP on $G(0)$ by considering $IP(n, G(0))$. By utilizing this, we bound the difference between the probability of any particular peer sample that PEERSWAP can provide and the uniform probability, *i.e.*, $\frac{1}{\binom{n-1}{d}}$. Complete proofs related to this section can be found in Appendix IX.C of the full version of the paper [39].

Theorem 3. *Consider a set of peers $U = \{u_1, \ldots, u_n\}$ connected in a d-regular connected graph $G(0) = (V, E(0))$, such that $V = \{1, 2, \ldots, n\}$ and each node i corresponds to peer u_i. Also, consider* PEERSWAP *protocol starting from $G(0)$ with Poisson clocks of rate α. Then for any $\varepsilon > 0$, any peer $u_i \in U$, and any $\mathbf{u} \in (U\backslash\{u_i\})_d$ the following holds true*

$$\left|\mathbb{P}[Sample_i(d, T) = \mathbf{u}] - \frac{(n-d-1)!}{(n-1)!}\right| \leq 4\varepsilon,$$

where $T := \alpha \cdot T_{Y^{(d+1)}}(\varepsilon)$ and $Y^{(d+1)}$ is an $IP(d+1, G(0))$ i.e., ε-mixing time of an interchange process $IP(d+1, G(0))$.

Proof Sketch. We start by showing that for bounding the left side of theorem inequality it is sufficient to bound the difference between the probabilities that $\mathbf{u}$ and any other sample $\mathbf{u}' \in (U\backslash\{u_i\})_d$ are neighborhoods of u_i. Then, we use the fact from Theorem3 that connections between u_i and $\mathbf{u}$ (resp. $\mathbf{u}'$) in PEERSWAP follow the same dynamic as those of peers in Y (resp. Y'), interchange processes $IP(d+1, G(0))$. Processes Y and Y' have the same stationary distributions and mixing times $T = T_Y(\varepsilon) = T_{Y'}(\varepsilon)$. Hence, we consider ρ – the stationary probability of the event that peers are located in such a way that the first one has all others in the neighborhood. Then after time T both $\mathbb{P}[Sample_i(d, T) = \mathbf{u}]$ and $\mathbb{P}[Sample_i(d, T) = \mathbf{u}']$ differ from ρ no more than by ε. This implies that the difference between these probabilities is no more than 2ε. The amount of different samples is $\frac{(n-1)!}{(n-d-1)!}$ that concludes bounding of the theorem expression. □

The above result ensures that distribution of $Sample_i(d, t)$ converges to a uniform distribution on $(U\backslash\{u_i\})_d$ for any $u_i \in U$. We now assess the convergence time of PEERSWAP.

Theorem 4. *Consider a set of peers $U = \{u_1, \ldots, u_n\}$ connected in a d-regular connected graph $G(0) = (V, E(0))$ and* PEERSWAP *protocol executing starting from $G(0)$ with Poisson clocks of rate α. For any peer $u_i \in U$, we consider the random variable $Sample_i(d, t)$ after time t and denote by $L(Sample_i(d, t))$ its probability distribution. Then for any $\delta \in (0, 1)$ there exists $T \in O\left(\frac{\alpha \log(n) \log(n^{d+1}/\delta)}{d\lambda(G(0))}\right)$ such that:*

$$d_{TV}\left(L(Sample_i(d, T)), Uniform((U\backslash\{u_i\})_d)\right) \leq \delta.$$

Proof Sketch. To calculate the total variation distance described in the theorem statement, we write its value by Definition 3 and use the result of Theorem 3. Then, we estimate the value of T by bounding the mixing time of an interchange process $IP(d+1, G(0))$. This bound is possible due to the connection of interchange processes and random

walks. Random walks are ctMCs with mixing time that is directly related to the graph connectivity. □

Implications of Theorem 4. Hence, when δ and α are constants of n, the distribution of $Sample_i(d,t), \forall i \in V$ is δ-close to the uniform one after time $T = O\left(\frac{\log^2(n)}{\lambda(G(0))}\right)$, which is a simpler expression than the bound derived above. Moreover, if the graph is sufficiently-connected, *i.e.*, the spectral gap of $G(0)$ is $\Omega(\frac{1}{\log^c(n)})$, where c is a constant (applicable to many popular graphs such as hypercubes, Erdos-Renyi or expander graphs), then the convergence time of PEERSWAP on network $G(0)$ is polylogarithmic. Lastly, we note that our upper bound on convergence time with sample size d also applies to any other sample size b where $1 \leq b \leq d$.

VI. A PRACTICAL IMPLEMENTATION OF PEERSWAP

So far, we have presented the PEERSWAP protocol and theoretically proven its convergence. In this section, we present a variant of Algorithm 1 that can function in a setting with network delays.

A. The Challenge of PEERSWAP with Network Delays

Deploying Algorithm 1 in P2P networks with network delays may fail, as network delays affect the ordering of messages received by peers and thus break the fixed structure of the underlying topology. This is because the execution of a single swap A consists of two sequential message deliveries (SWAP and REPLACE), each message taking some time to be delivered if network delays are present. Meanwhile, the execution of another swap may interfere with A. We elaborate this with an example, involving four peers u_i, u_j, u_k and u_l, as shown in Figure 3. When the Poisson clock associated with the edge (u_i, u_j) rings, u_j will send a SWAP message to u_i (for presentation clarity, we do not show the SWAP message by u_i). Now assume that nearly at the same time, the Poisson clock associated with the edge (u_k, u_l) rings. Peer u_l thus sends a SWAP message to u_k. Assume that u_k receives this swap message before the REPLACE message from u_i, the latter being associated with the swap between u_i and u_j. Now, u_k will replace its neighbors with the neighbors of u_l. When u_k later receives the REPLACE message from u_i, u_k might now be unable to correctly replace u_i in its neighbor list, resulting in an execution error. More specifically, the REPLACE message sent by u_i to u_k should have been sent to u_l instead.

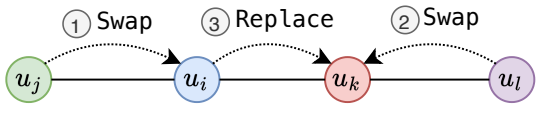


Fig. 3: A situation where the execution of Algorithm 1 in settings with network delays can violate protocol correctness and alter the fixed structure of the graph.

B. Lock-based Peer Swaps

Generally speaking, it is unsafe to execute the PEERSWAP protocol for any two swaps involving at least one common peer. We address this concern by having peers *lock other peers prior to executing a swap*. In this extended protocol, a peer u_i that is locked for some swap A will only process messages related to A. When a swap between u_i and u_j starts, both u_i and u_j attempt to lock their neighbor peers. If u_i, u_j, or any of their neighbors is already locked for another swap, the swap fails. Despite the fact that some swaps might fail, we found this simple protocol to be effective and avoiding the need for more resource-intensive forms of coordination such as consensus or other agreement protocols. We next describe the involved steps in this extended protocol that makes PEERSWAP suitable for deployment in settings with network delays. We explain the execution of a particular swap A between peers u_i and u_j, and outline the scenarios where a swap can fail.

Preparing for a swap. Each peer keeps track of its lock status, *i.e.*, if it has been locked for some swap. Whenever a Poisson clock on some edge $e = (i, j)$ rings, u_i, respectively u_j, checks if it is already locked. If u_i is not locked for another swap $A' \neq A$, it will lock itself for A. u_i then request its neighbors to lock themselves as well for A by sending a LOCKREQUEST message to all its neighbors except u_j. This message includes the specifications of swap A.

When peer u_k receives a LOCKREQUEST message from u_i, u_k will check if it is already locked for another swap $A' \neq A$. If not, it will lock itself for A and reply with a LOCKRESPONSE message. This message contains a binary flag *success* that indicates whether u_k has successfully locked itself in response to the received LOCKREQUEST message from u_i. If u_k is not locked, *success* will be set to true, whereas if u_k is already locked for another swap, it will be set to false.

Upon reception of a LOCKRESPONSE message by u_i from u_k, u_i verifies if it is still participating in swap A, *i.e.*, is locked for this swap, and ignores the message if it is unlocked. This check is necessary because swap A could have already failed (explained below) even though the LOCKRESPONSE message from u_k (or the LOCKREQUEST message to u_k) was still in transit. If all neighbors of u_i replied with a LOCKRESPONSE message containing a positive value for the SUCCESS field, u_i sends a SWAP message to u_j. However, if even a single LOCKRESPONSE message with a negative value for the SUCCESS field is received, the swap cannot proceed. In this situation, u_i sends an UNLOCK message to the neighbors it previously sent a LOCKREQUEST to, to ensure that these previously-locked neighbors can participate in other swaps. Furthermore, it sends a SWAPFAIL message to u_j, informing about the failed swap. When u_j receives a SWAPFAIL message from u_i, it also sends an UNLOCK message to its neighbors, except sending it to u_i. u_j will then unlock itself.

Executing a swap. If u_i has received positive LOCKRESPONSE messages from all its neighbors, as well as received a SWAP message from u_j, swap A is now safe to execute as all peers involved in A are aware of the swap and locked for

it. At that point, u_i sends REPLACE messages to its neighbors, similarly as described in Algorithm 1. Then, u_i erases its neighborhood and stores identifiers of the neighbors of u_j contained in the received SWAP message. Finally, u_i unlocks itself as it has completed all required actions for swap A. When u_k receives a REPLACE message, it will replace u_i with u_j in its neighborhood and unlock itself. This completes the execution of swap A.

VII. NUMERICAL EVALUATION

We present numerical evaluations that analyze the convergence speed of PEERSWAP under different parameters and network topologies. We have implemented a simulation of PEERSWAP in the Python programming language, using the NETWORKX library to generate network topologies. This simulator includes an implementation of our lock-based swap protocol described in Section VI. All our source code, experiment scripts, and documentation is publicly available.[3]

A. Setup and Convergence Metric

The main objective of our evaluation is to empirically show that PEERSWAP quickly provides peers with a neighborhood of peers that is indistinguishable from taking a uniform random sample from all peers in the topology. To do so, we run PEERSWAP with different parameters and pre-defined experiment durations T. If not stated otherwise, the default rate of Poisson clocks is 1. At the end of each experiment, we observe the neighborhood of one or more peers. We run each experiment multiple times (we specify how many for each experiment in this section), which yields the frequencies of final neighborhoods encountered for each peer. In perfect circumstances, each peer encounters each neighborhood with equal frequencies; however, these frequencies should show some variance in practice. To understand the convergence of PEERSWAP, we disable network latencies for all upcoming experiments, except for those in Section VII-D.

To determine convergence in PEERSWAP, we run a Kolmogorov–Smirnov (KS) test [42] between the observed frequencies of neighborhoods and a synthetic sequence of neighborhood frequencies that we construct by uniformly sampling all possible neighborhoods. The KS-test is a standard test that is frequently used to compare the similarity of distribution functions. In the context of PEERSWAP, it indicates how far the neighborhood frequencies of a particular peer differ from a uniform random distribution. This test yields two values: a distance and a p-value. In our experiments, the KS-test distance quantifies the maximal deviation between the cumulative distribution functions of the observed and expected neighborhood frequencies, indicating the extent of similarity. The p-value essentially assesses the statistical significance of this deviation. More precisely, it represents the probability of wrongly rejecting the null hypothesis "the two distributions are identical." A lower p-value thus suggests less likelihood that the observed distribution matches a uniform distribution.

[3]Source code available at https://github.com/sacs-epfl/peerswap.

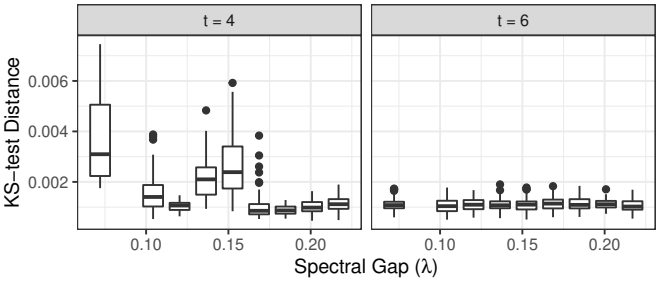


Fig. 4: The distribution of KS-test distances for neighborhoods distributions, when running PEERSWAP for topologies with different spectral gaps (λ) and experiment durations (T).

The following experiments are presented in three parts. In Section VII-B, we provide an extensive analysis of neighborhoods' frequencies (of $Sample_i(\mathrm{b}, T)$, when $\mathrm{b} = d$) when running PEERSWAP in small-scale networks, *i.e.*, $n \leq 100$ peers. In Section VII-C, we present a peer-frequency analysis (of $Sample_i(\mathrm{b}, T)$, when $\mathrm{b} = 1$) on large networks, *i.e.*, $n \geq 1000$ peers. Finally, in Section VII-D, we explore the convergence and performance of PEERSWAP with different forms of network delays.

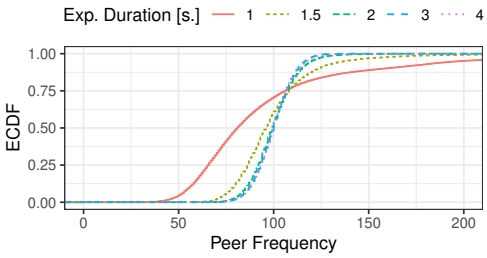
B. Neighborhood Frequencies for Small-scale Networks

We first explore how the distance between neighborhood frequencies in PEERSWAP and a uniform distribution is influenced by the network connectivity (measured by spectral gap), the initial position of peers in the graph, and the execution time of PEERSWAP.

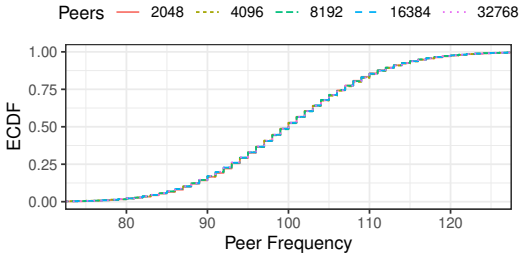
Setup. We generate d-regular topologies, run PEERSWAP on each topology for a period of T seconds, and observe the neighborhood of each peer after the experiment ends. There are $\binom{n-1}{d}$ possible neighborhoods for a particular peer. For a d-regular topology with n peers, we run each experiment $\left\lceil \binom{n-1}{d} \times \frac{100}{d} \right\rceil$ times. This ensures that, on average, each neighborhood is observed 100 times. We fix $d = 4$ and $n = 64$.

Since we aim to analyze how the spectral gap affects the convergence speed of PEERSWAP but cannot directly generate topologies with a particular spectral gap, we do the following. We first generate 10^6 4-regular topologies with 64 peers, compute for each topology its spectral gap λ, and select ten of these topologies such that their spectral gaps somewhat uniformly cover values in the range $\lambda = [0.07, 0.22]$ (these values are the lowest and highest λ we found throughout topology generation). A higher spectral gap λ indicates a better-connected topology, and we aim to understand its impacts on the convergence of PEERSWAP empirically. We run PEERSWAP on these topologies for $T = 4$ and $T = 6$ seconds while recording the neighborhood frequencies for *each* peer. We then run a KS test for each previously received neighborhood distribution.

Results. Figure 4 shows the variance in KS-test distances with a boxplot for each tested topology and for $T = 4$ (left) and $T = 6$ (right). Each boxplot consists of 64 values, each associated with the KS-test distance computed using the



(a) Peer frequencies after various durations ($n = 4096, d = 5$).



(b) Peer frequencies for different network sizes ($d = 5, T = 5$).

Peers (n)	KS-test distance	KS-test p-value
2048	0.020	0.79
4096	0.011	0.96
8192	0.012	0.65
16 384	0.006	0.88
32 768	0.003	0.97

(c) KS-test statistics associated with plot (b), for different values of n.

Fig. 5: ECDFs of peer frequencies' distributions across diverse experiment durations and larger topologies (up to 32 678 nodes). For $T \geq 3$, ECDFs are visually indistinguishable from those obtained through uniform sampling.

neighborhood distribution of a peer in the topology. On the one hand, for the topologies with a small spectral gap (*e.g.*, $\lambda = 0.07$ or $\lambda = 0.088$), we observe relatively high KS-test distances for $T = 4$ and small p-values ($p \leq 0.05$) for individual KS tests, meaning that for these topologies the neighborhood frequencies cannot be reasonably assumed to have converged to a uniform distribution yet[4]. On the other hand, for topologies with $\lambda \geq 0.18$, we find large p-values, suggesting that PEERSWAP on these topologies has converged at $T = 4$. Most of the KS-test distances are relatively low (below 0.01). The KS-test distances corresponding to $T = 6$ are lower compared to $T = 4$, and we find that 89.9% of all p-values produced by the KS test are higher than 0.05. Furthermore, we observe a low variance of KS-test distances for a particular topology and $T = 6$, indicating that the initial starting position of the peer in the graph does not affect its convergence much. This suggests that the neighborhoods of peers indeed approach a uniform sample over time.

Conclusion. These experiments demonstrate that PEERSWAP requires less time to generate uniformly random peer samples on networks with better connectivity (*i.e.*, higher spectral gaps). For most 4-regular topologies with $2^6 = 64$ peers, and irrespective of the initial starting position of peers, the neighborhood distribution closely approximates uniformity after just 6 seconds. This motivates us to focus on specific topologies and observed peers in further experiments.

C. Peer Frequencies for Large-scale Networks

Since ensuring coverage of all unique neighborhoods quickly becomes an intractable problem when the network size increases, we have limited our empirical analysis in Section VII-B to relatively small topologies ($n = 64$). To evaluate PEERSWAP with larger topologies, we count the frequencies on a per-peer instead of a per-neighborhood basis.

Setup. Similar to the previous experiments, we repeat each experiment until, on average, each peer is observed 100 times. Also, based on the outcome of our previous experiment in Section VII-B, we established that results do not vary significantly, either for different observed peers or for topologies with different but relatively high connectivity. Therefore, in this set of experiments, we average the results for each pair of (n, d) over five different random topologies while examining one random peer. We focus on how KS-test distances change with different network sizes n and durations of the PEERSWAP execution T.

Results. Figure 5a shows the empirical cumulative distribution function (ECDF) of peer frequencies for topologies with $n = 4096$, $d = 5$ and with five different experiment durations T. We visually notice that the distribution of peer frequencies quickly approaches uniform sampling, *e.g.*, peer frequencies become more concentrated around 100. Figure 5b shows the distribution of peer frequencies for increasing values of n up to 32 768 (2^{15}) peers while fixing $d = 5$ and $T = 5$. We observe that all network sizes yield peer frequencies visually indistinguishable from one another and from those obtained through uniform sampling. Finally, we summarize the KS-test distances for each value of n and topology in Figure 5c. We can see that the KS-test distance ranges from 0.003 to 0.02 and provides large p-values for all tests.

Conclusion. For the evaluated large-scale topologies, the distribution of peer frequencies approaches that of uniform sampling in a short time. This demonstrates the convergence of PEERSWAP in large networks.

D. Convergence and Throughput of PEERSWAP with Network Delays

In this part, we explore the convergence and performance of PEERSWAP in the presence of network delays. We use the adapted protocol as described in Section VI. Our goal is to see if the distribution of peer samples provided by PEERSWAP converges to the uniform one even if messages are delayed. Additionally, we believe that adjusting the main parameter of PEERSWAP, the Poisson clocks rate α, can result in more swaps per second and thus help to improve the convergence speed. To explore this, we specifically focus on the average throughput of PEERSWAP, *i.e.*, the average number of (successful) swaps per second, for varying network delays and edge activation frequencies.

Setup. We consider a network of $n = 1024$ peers, connected in a d-regular topology with $d = 5$. For each pair of peers, we generate pair-wise network delays through uniformly

[4]For the topology with $\lambda = 0.088$, we find a disproportionate median KS-test distance of 0.15 with $T = 4$. For presentation clarity, we omitted the corresponding boxplot from Figure 4.

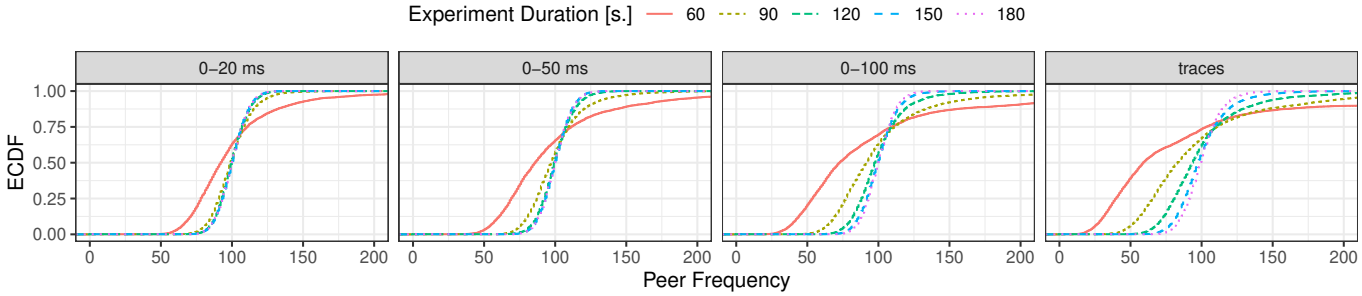


Fig. 6: Distribution of peer frequencies for different $\delta_{max} \in \{20, 50, 100\}$ values (in milliseconds) and with realistic traces applied (right-most plot), for different experiment durations.

drawing a delay between the interval $[0, \delta_{max}]$. This set of experiments considers three synthetic delay values – those of $\delta_{max} \in \{20, 50, 100\}$ ms. We also run a fourth configuration with realistic network traces that are sourced from WonderNetwork [43]. For the sake of presentation, we present results with respect to the average number of edge activations in the system, denoted by r. This value depends on the total number of edges in the graph and Poisson clock rate α.

First, we run a similar experiment to Section VII-C in the presence of delays. We consider various durations of $T \in \{60, 90, 120, 150, 180\}$ seconds and derive the distribution of peer frequencies provided by samples of PEERSWAP. Here, we adjust the Poisson clock rate α in such a way that on average 50 swaps per second are initiated.

Second, we control the interconnected values of α and r, and estimate the influence of Poisson clocks' rates on the PEERSWAP throughput. We vary r from 10 to 100 edge activations per second. Our goal is to estimate PEERSWAP throughput, defined as the average number of successful swaps per second. Higher is the throughput, quicker is the convergence of PEERSWAP. We run each experiment for two minutes and repeat for five seeds.

Convergence results. Figure 6 shows the empirical cumulative distribution function (ECDF) of peer frequencies with varying delays δ_{max} and when using traces. Comparing the plots in Figure 6, we can see that it takes longer for PEERSWAP to converge when the larger delays or realistic traces are applied. This is due to the lower throughput of PEERSWAP with realistic traces and large delays than with smaller delays.

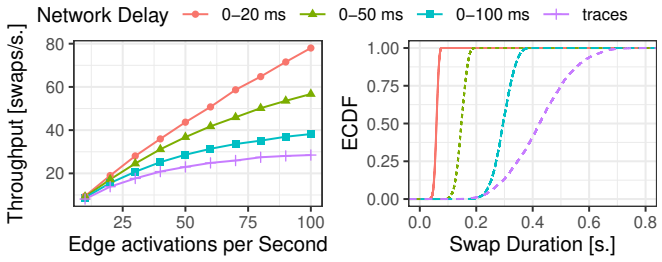


Fig. 7: The throughput of PEERSWAP (in swaps/s.) for increasing frequencies of edge activation (left) and the distribution of individual swap durations (right), for different network delays.

Throughput results. Figure 7 (left) shows the throughput of PEERSWAP, for different values of δ_{max} and with our realistic traces. Increasing the swap frequency generally increases the achievable throughput. We find this increase to flatten out, which is particularly noticeable for the experiments with larger network delays. This can be explained by two reasons. First, the probability of two "conflicting" swaps being initiated at the same time increases as more swaps are initiated within the same time period. Second, when network delays increase, each swap takes longer to finish, resulting in prolonged periods of nodes being locked. Figure 7 (right) shows the distribution of swap durations for each network delay profile, indeed confirming our observation. While throughput optimization is not the main focus of our paper, modifying the Poisson clock rate is a tangible way of improving in performance, especially if message delays are mild (*i.e.*, up to 20 ms). This comes, however, at the cost of additional network traffic.

Conclusion. Our empirical results hint that PEERSWAP can reach quick convergence even in challenging network conditions, *e.g.*, when deployed on the Internet. Also, adjustment of the Poisson clocks rate can significantly improve the throughput and protocol convergence.

VIII. CONCLUDING REMARKS

Our work makes the first step towards establishing gossip-based peer samplers with provable randomness guarantees. We introduced PEERSWAP, a protocol which provides each peer with a random sample within a time frame determined by the network's size and connectivity. The idea underlying PEERSWAP is simple: two adjacent peers periodically swap their entire neighborhood. The theoretical analysis, which is more challenging, has been made possible by establishing a link between the dynamics of peers in PEERSWAP and interchange processes. We also introduced a variant of PEERSWAP that operates in the presence of network delays. Our numerical evaluation using a simulated version of PEERSWAP conveys the convergence of PEERSWAP over time, thus confirming that it is an efficient and scalable peer sampler.

ACKNOWLEDGMENTS

The authors thank Roberto Oliveira and Laurent Massoulie for their insights on interchange processes, and Nirupam Gupta, Rishi Sharma, and Akash Dhasade for their fruitful discussions and proofreading of the paper.

References

[1] R. Steinmetz and K. Wehrle, *Peer-to-peer systems and applications*, 2005, vol. 3485.

[2] G. P. Jesi, D. Hales, and M. van Steen, "Identifying malicious peers before it's too late: a decentralized secure peer sampling service," in *Proceedings of the 1st International Conference on Self-Adaptive and Self-Organizing Systems (SASO)*, 2007, pp. 237–246.

[3] D. Kempe, A. Dobra, and J. Gehrke, "Gossip-based computation of aggregate information," in *Proceedings of the 44th Annual IEEE Symposium on Foundations of Computer Science(FOCS)*, 2003, pp. 482–491.

[4] R. Karp, C. Schindelhauer, S. Shenker, and B. Vocking, "Randomized rumor spreading," in *Proceedings of the 41st Annual Symposium on Foundations of Computer Science (FOCS)*, 2000, pp. 565–574.

[5] R. Baldoni, R. Beraldi, V. Quema, L. Querzoni, and S. Tucci-Piergiovanni, "Tera: topic-based event routing for peer-to-peer architectures," in *Proceedings of the inaugural international conference on Distributed event-based systems*, 2007, pp. 2–13.

[6] L. Massoulié, E. Le Merrer, A.-M. Kermarrec, and A. Ganesh, "Peer counting and sampling in overlay networks: random walk methods," in *Proceedings of the 35th annual ACM symposium on Principles of distributed computing (PODC)*, 2006, pp. 123–132.

[7] S. Chatterjee, G. Pandurangan, and P. Robinson, "Byzantine-resilient counting in networks," in *Proceedings of the 42nd International Conference on Distributed Computing Systems (ICDCS)*, 2022, pp. 12–22.

[8] S. Bano, A. Sonnino, M. Al-Bassam, S. Azouvi, P. McCorry, S. Meiklejohn, and G. Danezis, "Sok: Consensus in the age of blockchains," in *Proceedings of the 1st ACM Conference on Advances in Financial Technologies*, 2019, pp. 183–198.

[9] R. Baldoni, A. Corsaro, L. Querzoni, S. Scipioni, and S. T. Piergiovanni, "Coupling-based internal clock synchronization for large-scale dynamic distributed systems," *IEEE Transactions on Parallel and Distributed Systems*, vol. 21, pp. 607–619, 2009.

[10] S. Voulgaris and M. Van Steen, "Epidemic-style management of semantic overlays for content-based searching," in *Proceedings of the 11th International Euro-Par Conference*, 2005, pp. 1143–1152.

[11] M. Jelasity, A. Montresor, and O. Babaoglu, "T-man: Gossip-based fast overlay topology construction," vol. 53, pp. 2321–2339, 2009.

[12] M. de Vos, S. Farhadkhani, R. Guerraoui, A.-M. Kermarrec, R. Pires, and R. Sharma, "Epidemic learning: Boosting decentralized learning with randomized communication," in *Proceedings of the 37th Annual Conference on Neural Information Processing Systems (NeurIPS)*, 2023.

[13] D. Stutzbach, R. Rejaie, N. Duffield, S. Sen, and W. Willinger, "On unbiased sampling for unstructured peer-to-peer networks," in *Proceedings of the 6th ACM conference on Internet measurement(SIGCOMM)*, 2006, pp. 27–40.

[14] M. Zhong, K. Shen, and J. Seiferas, "The convergence-guaranteed random walk and its applications in peer-to-peer networks," *IEEE Transactions on Computers*, vol. 57, no. 5, pp. 619–633, 2008.

[15] Z. Bar-Yossef, R. Friedman, and G. Kliot, "Rawms-random walk based lightweight membership service for wireless ad hoc networks," *ACM Transactions on Computer Systems (TOCS)*, vol. 26, no. 2, pp. 1–66, 2008.

[16] C. Gkantsidis, M. Mihail, and A. Saberi, "Random walks in peer-to-peer networks," in *Proceedings of the IEEE International Conference on Computer Communications (INFOCOM)*, vol. 1, 2004.

[17] C. Law and K.-Y. Siu, "Distributed construction of random expander networks," in *Proceedings of the 22nd Annual Joint Conference of the Computer and Communications Societies (INFOCOM)*, vol. 3, 2003, pp. 2133–2143.

[18] A. Sevilla, A. Mozo, and A. F. Anta, "Node sampling using random centrifugal walks," *Journal of Computational Science*, vol. 11, pp. 34–45, 2015.

[19] M. Ripeanu, "Peer-to-peer architecture case study: Gnutella network," in *Proceedings of the 1st international conference on peer-to-peer computing*, 2001, pp. 99–100.

[20] S. Nakamoto, "Bitcoin: A peer-to-peer electronic cash system," *Decentralized business review*, 2008.

[21] M. Jelasity, S. Voulgaris, R. Guerraoui, A.-M. Kermarrec, and M. Van Steen, "Gossip-based peer sampling," *ACM Transactions on Computer Systems (TOCS)*, vol. 25, no. 3, pp. 8–es, 2007.

[22] S. Voulgaris, D. Gavidia, and M. Van Steen, "Cyclon: Inexpensive membership management for unstructured P2P overlays," *Journal of Network and systems Management*, vol. 13, pp. 197–217, 2005.

[23] N. Tölgyesi and M. Jelasity, "Adaptive peer sampling with newscast," in *European Conference on Parallel Processing*, 2009, pp. 523–534.

[24] A. Allavena, A. Demers, and J. E. Hopcroft, "Correctness of a gossip based membership protocol," in *Proceedings of the 24th symposium on Principles of distributed computing (PODC)*, 2005, pp. 292–301.

[25] A. Auvolat, Y.-D. Bromberg, D. Frey, D. Mvondo, and F. Taïani, "Basalt: A rock-solid byzantine-tolerant peer sampling for very large decentralized networks," in *Proceedings of the 24th International Middleware Conference*, 2023, pp. 111–123.

[26] A. Antonov and S. Voulgaris, "Securecyclon: Dependable peer sampling," in *Proceedings of the 43rd International Conference on Distributed Computing Systems (ICDCS)*, 2023, pp. 1–12.

[27] M. Pigaglio, J. Bruneau-Queyreix, Y.-D. Bromberg, D. Frey, E. Rivière, and L. Réveillère, "Raptee: Leveraging trusted execution environments for byzantine-tolerant peer sampling services," in *Proceedings of the 42nd International Conference on Distributed Computing Systems (ICDCS)*, 2022, pp. 603–613.

[28] G. Badishi, I. Keidar, and A. Sasson, "Exposing and eliminating vulnerabilities to denial of service attacks in secure gossip-based multicast," *IEEE Transactions on Dependable and Secure Computing*, vol. 3, no. 1, pp. 45–61, 2006.

[29] H. Johansen, A. Allavena, and R. Van Renesse, "Fireflies: scalable support for intrusion-tolerant network overlays," *The ACM Special Interest Group in Operating Systems (SIGOPS)*, vol. 40, no. 4, pp. 3–13, 2006.

[30] E. Bortnikov, M. Gurevich, I. Keidar, G. Kliot, and A. Shraer, "Brahms: Byzantine resilient random membership sampling," in *Proceedings of the 27th ACM symposium on Principles of distributed computing (PODC)*, 2008, pp. 145–154.

[31] A.-M. Kermarrec, V. Leroy, and C. Thraves, "Converging quickly to independent uniform random topologies," in *Proceedings of the 19th International Euromicro Conference on Parallel, Distributed and Network-Based Processing*, 2011, pp. 159–166.

[32] Y. Busnel, R. Beraldi, and R. Baldoni, "On the uniformity of peer sampling based on view shuffling," *Journal of Parallel and Distributed Computing*, vol. 71, no. 8, pp. 1165–1176, 2011.

[33] P. Mahlmann and C. Schindelhauer, "Peer-to-peer networks based on random transformations of connected regular undirected graphs," in *Proceedings of the 17th annual ACM symposium on Parallelism in algorithms and architectures (SPAA)*, 2005, pp. 155–164.

[34] T. Feder, A. Guetz, M. Mihail, and A. Saberi, "A local switch markov chain on given degree graphs with application in connectivity of peer-to-peer networks," in *Proceedings of the 47th Annual IEEE Symposium on Foundations of Computer Science (FOCS)*, 2006, pp. 69–76.

[35] C. Cooper, M. Dyer, and 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)*, 2009, p. 141–150.

[36] A. Lazarescu, "The physicist's companion to current fluctuations: one-dimensional bulk-driven lattice gases," *Journal of Physics A: Mathematical and Theoretical*, vol. 48, no. 50, p. 503001, 2015.

[37] T. M. Liggett, *Stochastic interacting systems: contact, voter and exclusion processes*, 1999, vol. 324.

[38] A. J. Wood, R. A. Blythe, and M. R. Evans, "Combinatorial mappings of exclusion processes," *Journal of Physics A: Mathematical and Theoretical*, vol. 53, no. 12, p. 123001, 2020.

[39] R. Guerraoui, A.-M. Kermarrec, A. Kucherenko, R. Pinot, and M. de Vos, "Peerswap: A peer-sampler with randomness guarantees," 2024. [Online]. Available: https://arxiv.org/abs/2408.03829

[40] D. A. Levin and Y. Peres, *Markov chains and mixing times*. American Mathematical Soc., 2017, vol. 107.

[41] R. I. Oliveira, "Mixing of the symmetric exclusion processes in terms of the corresponding single-particle random walk," 2013.

[42] V. W. Berger and Y. Zhou, "Kolmogorov–smirnov test: Overview," *Wiley statsref: Statistics reference online*, 2014.

[43] WonderNetwork, "Global ping statistics," https://wondernetwork.com/pings, accessed: 2022-05-12.