Uniform Node Sampling Service Robust against Collusions of Malicious Nodes

Emmanuelle Anceaume
IRISA / CNRS
Rennes, France
Emmanuelle.Anceaume@irisa.fr

Yann Busnel
LINA / Université de Nantes
Nantes, France
Yann.Busnel@univ-nantes.fr

Bruno Sericola
Inria Rennes – Bretagne
Atlantique, France
Bruno.Sericola@inria.fr

***Abstract*—We consider the problem of achieving uniform node sampling in large scale systems in presence of a strong adversary. We first propose an omniscient strategy that processes on the fly an unbounded and arbitrarily biased input stream made of node identifiers exchanged within the system, and outputs a stream that preserves Uniformity and Freshness properties. We show through Markov chains analysis that both properties hold despite any arbitrary bias introduced by the adversary. We then propose a knowledge-free strategy and show through extensive simulations that this strategy accurately approximates the omniscient one. We also evaluate its resilience against a strong adversary by studying two representative attacks (flooding and targeted attacks). We quantify the minimum number of identifiers that the adversary must insert in the input stream to prevent uniformity. To our knowledge, such an analysis has never been proposed before.**

Keywords*—*Data stream; strong adversary; uniform sampling; Markov chains; randomized approximation algorithm.

I. Introduction

The uniform node sampling service offers to applications using it a single simple primitive that returns the identifier of a random node that belongs to the system. Providing at any time randomly chosen nodes in the system has deserved a lot of attention to construct large scale distributed applications. A typical example is load balancing in cluster-based applications: choosing a host at random among those that are available is often a choice that provides performance close to that offered by more complex selection criteria, without imposing any burden [1]. Another important example is epidemic-based applications: by periodically selecting few random nodes as neighbors, large-scale environments preserve their connectivity despite nodes dynamicity [2]–[5].

Node sampling is a cooperative service in the sense that all the nodes of the system contribute to this service by continuously sending and forwarding information about their presence. Unfortunately, the unavoidable presence of malicious nodes in large scale and open systems seriously impedes the construction of uniform node sampling [6]–[8]. The objective of malicious nodes mainly consists in continuously and largely biasing the input data stream out of which samples are obtained, to prevent (correct) nodes from being selected as samples. Consequences of these collective attacks (typically called Sybil attacks [9]) are, among others, the overwhelming load of some specific nodes when it is used to provide random locations for data caching or storage, or the eventual partitioning of the system when the node sampling service is used to build nodes local views in epidemic-based protocols. Solutions that basically consist in storing the identifier of all the nodes of the system so that each of these node identifiers can be randomly selected when needed are impracticable and even infeasible due to the size and the dynamicity of such networks. Rather providing a solution that requires as little space as possible (*e.g.* sublinear in the population size of the system) is definitely desirable. Bortnikov *et al.* [10] have recently proposed a uniform node sampling algorithm that tolerates malicious nodes by exploiting the properties offered by min-wise permutations. Specifically, the sampling component, which is fed with the stream of node identifiers periodically gossiped by nodes, outputs the node identifier whose image value under the randomly chosen permutation is the smallest value ever encountered. Thus eventually, by the property of min-wise permutation, the sampler converges towards a random sample. However by the very same properties of min-wise permutation functions, once the convergence has been reached, it is stuck to this convergence value independently from any subsequent input values. Thus the sample does not evolve according to the current composition of the system, which makes it static. Actually it has been shown in [11] that imposing strict restrictions on the number of messages sent by malicious nodes during a given period of time and providing each correct node with a very large memory (proportional to the size of the system) is a necessary and sufficient condition to output an unbiased and non static stream. Thus, lack of adaptivity or full-space algorithms seem to be the only defenses against adversarial behaviors when considering deterministic algorithms.

In this paper, we solve this problem by adopting a probabilistic approach. We first propose an omniscient algorithm, called in the following the *omniscient strategy*, that is capable of tolerating any bias introduced by the adversary in the input stream. By omniscient we mean that the algorithm knows the number of occurrences of each received element in the full input stream. We analyze the stationary behavior of this algorithm through a Markov chains analysis. We then propose a randomized approximation algorithm, called in the following the *knowledge-free strategy*, that is capable of outputting an unbiased and non static sample of the population whatever the strategy of the adversary is. This sample may deviate from an exact uniform sample, however the deviation is bounded with any tunable probability. This algorithm is a one-pass algorithm, *i.e.*, each piece of data of the input stream is scanned sequentially on the fly, and only compact synopses or sketches that contain the most important information about data items are locally stored. This algorithm does not require any a priori knowledge neither on the size of the input stream, nor on the number of distinct elements that compose it, nor on the frequency distribution of these elements. We then evaluate the minimum effort that needs to be exerted by a strong adversary to bias the output stream when two representative attacks are launched, *i.e.*, the *targeted attacks* in which the adversary focuses on biasing the frequency of a single node identifier, and the *flooding attack* which aims at biasing all the node identifiers frequencies. Both evaluations are conducted by modeling them as an urn problem. One of the main results of this analysis is the fact that the effort that needs to be exerted by the adversary to subvert the sampling service can be made arbitrarily large by any correct node by just increasing the memory space of the sampler. Finally, extensive simulations (both on real data and synthetic traces) confirm the robustness of our sampler service. To the best of our knowledge, no previous work has proposed such an analysis.

The outline of this paper is the following. In Section II, we present related works. Section III describes the model of the system and the one of the adversary. Section IV details the properties characterizing the node sampling service, and presents the omniscient and knowledge-free strategies to implement such a service. Stationary behavior of the node sampling service is studied. Section V presents an analysis of the minimum effort that needs to exerted by the adversary to subvert the node sampling service. Finally, extended simulations have been conducted in different adversarial contexts and the main lessons drawn from these simulations are presented in Section VI. Section VII concludes.

II. Related Work

Different approaches have been proposed to solve the node sampling problem in presence of malicious behaviors in large scale systems. Jesi *et al.* [6] propose a random sampling algorithm taking explicitly into account malicious nodes. Their solution assumes that the ultimate goal of the malicious nodes is to mutate the random graph into a hub-based graph, hub for which malicious nodes gain the lead. This approach, also adopted in several structured based systems [8] through auditing mechanisms, or in sensor networks [7], is effective only if the number of malicious nodes is very small with respect to the size n of the system (*i.e.*, typically of $O(\log n)$). As said in the Introduction, Bortnikov *et al.* [10] have proposed a uniform node sampling algorithm that tolerates up to a linear number of malicious nodes. Their sampling mechanism exploits the properties offered by min-wise permutations. The sampling component is fed with the stream of node identifiers periodically gossiped by nodes, and outputs the node identifier whose image value under the randomly chosen permutation is the smallest value ever encountered. Thus eventually, by the property of min-wise permutation, the sampler converges towards a random sample. By limiting the number of node identifiers malicious nodes can periodically issue (no more than 20% of the total number of requests can be sent by malicious nodes), their solution requires a single node identifier to be stored in the local memory. However, once convergence has been reached, it is stuck to this convergence value independently from the input values. Thus the sample does not evolve according to the current composition of the system.

Streaming algorithms have shown their highly desirable properties in data intensive monitoring applications. These algorithms process the input stream in a single pass and sequentially. All these algorithms rely on pseudo-random functions that map elements of the stream to uniformly distributed image values. The interested reader is invited to read the nice survey by Muthukrishnan [12]. Most of the research done so far with this approach has manly focused on computing functions or statistic measures with error ε using poly$(1/\varepsilon, \log N)$ space where N is the domain size of the data items. These include the computation of the number of different data items in a given stream [13]–[15], the frequency moments [16], the most frequent data items [16], [17], or the entropy of the stream [18], [19].

In this work, we go a step further by continuously computing, in a strong adversarial context, a uniform sample of the nodes of the system so that for any node identifier present in the stream, the probability that this node identifier is selected as a sample is equal to $1/n$, where n is the size of the population, and at any time, any node identifier in the stream has a non null probability to be selected as a sample.

III. System Model and Assumptions

A. Model of the Network

We consider a large scale and dynamic open system $\mathcal{N}$ in which each node $i \in \mathcal{N}$ receives a very large stream σ_i (or σ when it is clear from context) made of node identifiers (also denoted ids in the following). Node identifiers arrive quickly and sequentially. Each node identifier j of σ is drawn from a set $\Omega = \{1, \ldots, 2^r\}$, where r is chosen to be large enough to make the probability of identifiers collision negligible ($r = 160$ for the standard SHA-1 hash function). Node identifiers may recur in the stream with an unknown bias. The number of times a node identifier i recurs in the stream is commonly called the *frequency* of i. Data streams are potentially unbounded in size. For memory constraints, nodes can locally store only a small amount of information with respect to the number of nodes in the system. Thus the stream needs to be processed in an online manner, that is, any item of the stream that has not been locally stored for any further processing cannot be read anymore. In addition the amount of computation per data element of the stream must be low to keep pace with the data stream.

B. Adversary

We assume the presence of malicious (*i.e.*, Byzantine) nodes that collectively try to subvert the system by manipulating the prescribed protocol. We model these adversarial behaviors through an adversary that fully controls and manipulates these malicious nodes. We suppose that the adversary is strong in the sense that it may actively tamper with the data stream of any node i by observing, and inserting any number of malicious nodes identifiers. Indeed, the goal of the adversary is to judiciously increase the frequency of ℓ chosen node identifiers to bias the sample built by non malicious nodes. The number ℓ is chosen by the adversary and depends on the sampling protocol parameters. Note that each malicious node identifier does not need to correspond to a single real node. Indeed, the adversary will augment its power by generating numerous node identifiers, such that only a limited number of real malicious nodes are linked to these identifiers. However, affecting multiple identifiers to a single node is costly as one needs to interact with a central authority to receive a certificate assessing the validity and integrity of the identifier. This model refers to the Sybil model attack presented by Douceur and Donath [9].

A node present in the system that is not malicious is said to be *correct*. Note that correct nodes cannot *a priori* distinguish correct node identifiers from malicious ones. Classically, we assume that the adversary can neither drop a message exchanged between two correct nodes nor tamper with its content without being detected. This is achieved by assuming the existence of a signature scheme (and the corresponding public-key infrastructure) ensuring the authenticity and integrity of messages. This refers to the *authenticated Byzantine failure model* [21].We finally suppose that any algorithm run by any correct node to build a uniform node sampling service is public knowledge to avoid some kind of security by obscurity. However the adversary has not access to the local random coins used in the algorithms.

C. Sampling Assumptions

Similarly to Bortnikov *et al.* [10], we first assume that there exists a time T_0 such that after that time, the churn of the system ceases (churn is classically defined as the rate of turnover of nodes in large scale systems [22]). This assumption is necessary to make the notion of uniform sample meaningful. Thus from T_0 onwards, the population of the system $\mathcal{N}$ is composed of $n \leq 2^r$ nodes such that ℓ of them are malicious, with $\ell < n$. We also suppose that at any discrete time $t \geq T_0$ all the correct nodes in $\mathcal{N}$ are weakly connected, which means that there exists a path between any pair of correct nodes in $\mathcal{N}$. In the following, without loss of generality, we consider that $T_0 = 0$.

D. 2-universal Hash Functions

In the following, we intensively use hash functions randomly picked from a 2-universal hash family. Given an integer Y, let us denote $[Y] = \{1, \dots, Y\}$ in the remaining of this paper. A collection H of hash functions $h : [M] \to [M']$ is said to be *2-universal* if for every two different items $x, y \in [M]$,

$$\forall h \in H, \mathbb{P}\{h(x) = h(y)\} \leq \frac{1}{M'},$$

which is exactly the probability of collision obtained if the hash function assigned truly random values to any $x \in [M]$.

IV. Node Sampling Service tolerant to Malicious Nodes

A node sampling service tolerant to malicious nodes is a functionality local to each correct node i of the system[1]. This service continuously reads the input stream σ_i received by node i. Data streams are made of the nodes identifiers exchanged within the system. Note that the analysis presented in this paper is independent from the way data streams are built. That is, they may result from the continuous propagation of node ids through gossip-based algorithms, or from the node ids received during random walks initiated at each node of the system.

[1]Although malicious nodes have also access to a sampling service, we cannot impose any assumptions on how they use it as their behavior can be totally arbitrary.

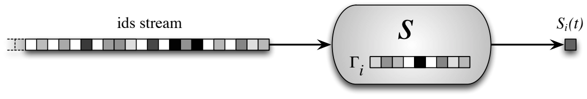


Fig. 1. Sampling component of node $i \in \mathcal{N}$.

In addition, the input stream of any correct node can be arbitrarily biased by an adversary, which is achieved by infinitely often augmenting it with the ℓ ids it manipulates. The objective of the sampling service *strategy* is to process on the fly the input stream and to output a stream guaranteeing both Uniformity and Freshness. Specifically, if $S_i(t)$ denotes the output of the sampling service at any correct node i at any discrete time t, then a sampling service tolerant to malicious behaviors should meet the following two properties.

Property 1 (Uniformity): For any discrete time $t \geq 0$, for any node $j \in \mathcal{N}$,

$$\mathbb{P}\{S_i(t) = j\} = \frac{1}{n}$$

Property 2 (Freshness): For any discrete time $t \geq 0$, for any node $j \in \mathcal{N}$,

$$\{t' > t \mid S_i(t') = j\} \neq \emptyset \text{ with probability } 1.$$

Uniformity states that any node in the system should have the same probability to appear in the sample of correct nodes in the overlay, while Freshness says that any node that recurs infinitely often in the stream, should have a non-null probability to appear infinitely often in the sample of any correct nodes in the system. It is important to note that even if the adversary can insert infinitely often the identifiers of the ℓ nodes it manipulates, by the weak connectivity assumption, at any discrete time $t \geq 0$ there is a non null probability that the node identifier of any correct node in $\mathcal{N}$ appears in the input stream of any correct node.

A. Omniscient One-pass Strategy

This section presents an *omniscient one-pass strategy* that guarantees both the Uniformity and Freshness properties. We recall that, by one-pass, we mean that the strategy reads sequentially the input stream and if some node id has not been locally stored for further processing, once it has been read it cannot be read anymore. By omniscient, we mean that the *strategy* knows exactly the population size n of the system $\mathcal{N}$ and each time a node id j is received in σ_i, the strategy knows exactly the occurrence probability p_j of j in the full stream σ_i. Note however that the omniscient strategy does not know ahead of time the identifiers that will appear in σ_i. This knowledge is built on the fly when reading σ_i.

The omniscient strategy has uniquely access to a data structure Γ_i, referred to as the *sampling memory* as illustrated in Figure 1. The cardinality of Γ_i is constant and is denoted by c with $c \ll n$. The sampling memory will contain the node ids that will be selected by the strategy when reading σ_i and that will be output by the sampler. Algorithm 1 describes the pseudo-code of the omniscient strategy.

Algorithm 1: Omniscient Node Sampling Strategy run at any correct node $i \in \mathcal{N}$

```
Input: An arbitrary input stream $\sigma_i$;
Output: A modified output stream $\sigma'_i$;
Data: $\Gamma_i$ a set of maximum size $c$;
1 for $j \in \sigma_i$ do
2     if $|\Gamma_i| < c$ then
3         $\Gamma_i \leftarrow \Gamma_i \cup \{j\}$;
4     else
5         with probability $a_j$ do
6             choose $k$ from $\Gamma_i$ with probability $\frac{r_k}{\sum_{\ell \in \Gamma_i} r_\ell}$;
7             $\Gamma_i \leftarrow (\Gamma_i \setminus \{k\}) \cup \{j\}$;
8     choose $k'$ from $\Gamma_i$ with probability $\frac{1}{c}$;
9     write $k'$ in the output stream;
```

Specifically, the omniscient strategy reads on the fly and sequentially the input stream and for each read element j, decides whether j is a good candidate for being stored into the constant size memory Γ_i or not. If p_j is very small, then j must definitively be stored into Γ_i so that j might have a chance to be part of the output stream. On the other hand, with larger p_j, there will be other opportunities for the sampler to receive j in the future. The probability to insert j in Γ_i is denoted by a_j in the algorithm. Although inserting j into Γ_i with probability a_j is a necessary condition to prevent very frequent ids from continuously eclipsing the ids already stored in Γ_i, this is not sufficient to guarantee that a rare id k already stored in Γ_i will not be evicted each time a new id j is stored (assuming that Γ_i is full upon receipt of j). Recall that the goal of the adversary is to prevent identifiers of correct nodes to uniformly appear in the output stream. A sufficient condition is achieved by removing k from Γ_i with probability $r_k / \sum_{\ell \in \Gamma_i} r_\ell$, where $r_1, \ldots, r_n$ are positive real numbers. Finally, a random node id k' is chosen from Γ_i and written in the output stream (note that k' is not removed from Γ_i). Clearly, for any node identifier $j \in \mathcal{N}, a_j$ depends on $(p_l)_{l \in \mathcal{N}}$ and r_j depends on $(p_l)_{l \in \mathcal{N}}$ and $(a_l)_{l \in \mathcal{N}}$.

In the remaining of this section, we prove that there exist both $(a_j)_{j \in \mathcal{N}}$ and $(r_j)_{j \in \mathcal{N}}$ such that the output stream provided by Algorithm 1 meets the Uniformity and Freshness properties. Specifically, we show that this is verified for the following two families

$$\forall j \in \mathcal{N}, a_j = \frac{\min_{i \in \mathcal{N}}(p_i)}{p_j} \text{ and } r_j = \frac{1}{n}.$$

To prove this assertion, we model the receipt of node ids from σ_i by using a homogeneous discrete-time Markov chain denoted by $X_i = \{X_{i,t}, t \geq 0\}$. Markov chain X_i represents the evolution of the node identifiers in Γ_i. Note that for clarity reason we shall omit the subscript i when it is clear from context. The state space S of X is defined by $S = \{A \subseteq \mathcal{N}$ such that $|A| = c\}$. For any $t \geq 0$, the event $X_t = A$ means that after the t-th transition (*i.e.*, the t-th received node identifier), $\Gamma = A$. The transition probability matrix, denoted by P, is given for every $A, B \in S$ with $A \neq B$, by

$$P_{A,B} = \begin{cases} \dfrac{r_i}{\sum\limits_{\ell \in A} r_\ell} p_j a_j & \text{if } A \setminus B = \{i\} \text{ and } B \setminus A = \{j\} \\ 0 & \text{otherwise} \end{cases}$$

Matrix P being stochastic, we have for every $A \in S$,

$$\begin{aligned} P_{A,A} &= 1 - \sum_{B \in S, B \neq A} P_{A,B} \\ &= 1 - \sum_{i \in A} \sum_{j \notin A} \left(\sum_{B \in S, A \setminus B = \{i\}, B \setminus A = \{j\}} P_{A,B} \right) \\ &= 1 - \sum_{i \in A} \sum_{j \notin A} \frac{r_i}{\sum\limits_{\ell \in A} r_\ell} p_j a_j \\ &= 1 - \sum_{j \notin A} p_j a_j \\ &= 1 - \sum_{j \in \mathcal{N}} p_j a_j + \sum_{j \in A} p_j a_j. \end{aligned}$$

It is easily checked that $|S| = \binom{n}{c}$.

The Markov chain X is clearly irreducible and aperiodic. It thus have a stationary distribution that we denote by $\pi = (\pi_A,\ A \in S)$. The row vector π is thus the unique solution to the linear system $\pi = \pi P$ with $\pi \mathbb{1} = 1$, where $\mathbb{1}$ is the column vector with all entries equal to 1. The symmetries observed in the transition probability matrix P gives us the intuition that X is reversible, *i.e.*, that, for every $A, B \in S$, we have $\pi_A P_{A,B} = \pi_B P_{B,A}$. This intuition is verified by the following theorem.

Theorem *3:* The Markov chain X is reversible and for every $A \in S$, we have

$$\pi_A = \frac{1}{K} \left(\sum_{\ell \in A} r_\ell \right) \left(\prod_{h \in A} \frac{p_h a_h}{r_h} \right) \tag{1}$$

where

$$K = \sum_{A \in S} \left(\sum_{\ell \in A} r_\ell \right) \left(\prod_{h \in A} \frac{p_h a_h}{r_h} \right).$$

Proof: Consider the vector $\pi = (\pi_A,\ A \in S)$ defined by Relation (1) and let $A, B \in S$ such that $A \neq B$. We then have, by definition of matrix P,

$$\begin{aligned}
&\pi_A P_{A,B}\\
&= \frac{1}{K}\left(\sum_{\ell\in A} r_\ell\right)\left(\prod_{h\in A}\frac{p_h a_h}{r_h}\right)\frac{r_i p_j a_j}{\sum_{\ell\in A} r_\ell}\mathbf{1}_{\{A\setminus B=\{i\},B\setminus A=\{j\}\}}\\
&= \frac{1}{K}\left(\prod_{h\in A}\frac{p_h a_h}{r_h}\right) r_i p_j a_j \mathbf{1}_{\{A\setminus B=\{i\},B\setminus A=\{j\}\}}\\
&= \frac{1}{K}\left(\prod_{h\in A\cap B}\frac{p_h a_h}{r_h}\right)\frac{p_i a_i}{r_i} r_i p_j a_j \mathbf{1}_{\{A\setminus B=\{i\},B\setminus A=\{j\}\}}\\
&= \frac{1}{K}\left(\prod_{h\in A\cap B}\frac{p_h a_h}{r_h}\right) p_i a_i p_j a_j \mathbf{1}_{\{A\setminus B=\{i\},B\setminus A=\{j\}\}}
\end{aligned}$$

In the same way, exchanging the roles of integers i and j and of sets A and B, we get

$$\begin{aligned}
&\pi_B P_{B,A}\\
&= \frac{1}{K}\left(\sum_{\ell\in B} r_\ell\right)\left(\prod_{h\in B}\frac{p_h a_h}{r_h}\right)\frac{r_j p_i a_i}{\sum_{\ell\in B} r_\ell}\mathbf{1}_{\{B\setminus A=\{j\},A\setminus B=\{i\}\}}\\
&= \frac{1}{K}\left(\prod_{h\in B}\frac{p_h a_h}{r_h}\right) r_j p_i a_i \mathbf{1}_{\{A\setminus B=\{i\},B\setminus A=\{j\}\}}\\
&= \frac{1}{K}\left(\prod_{h\in B\cap A}\frac{p_h a_h}{r_h}\right)\frac{p_j a_j}{r_j} r_j p_i a_i \mathbf{1}_{\{A\setminus B=\{i\},B\setminus A=\{j\}\}}\\
&= \frac{1}{K}\left(\prod_{h\in B\cap A}\frac{p_h a_h}{r_h}\right) p_j a_j p_i a_i \mathbf{1}_{\{A\setminus B=\{i\},B\setminus A=\{j\}\}}.
\end{aligned}$$

We thus have shown that, for every $A, B \in S$ such that $A \neq B$, we have $\pi_A P_{A,B} = \pi_B P_{B,A}$. This implies that X is reversible and that the probability vector $\pi = (\pi_A,\ A \in S)$, given by Relation (1), is the stationary distribution of X. ∎

Let us introduce now, for every $\ell \in \mathcal{N}$, the subset of states S_ℓ defined by

$$S_\ell = \{A \in S \mid \ell \in A\}$$

and consider the probability for X to be in subset S_ℓ in stationary regime. If we denote by γ_ℓ this probability, we have

$$\gamma_\ell = \sum_{A\in S_\ell} \pi_A.$$

Theorem *4:* For every $\ell \in \mathcal{N}$, the probability γ_ℓ is

$$\gamma_\ell = \frac{c}{n}.$$

Proof: It is easily checked, as expected, that we have

$$|S_\ell| = \binom{n-1}{c-1} \text{ and } \sum_{\ell=1}^{n} \gamma_\ell = c.$$

By taking, for every $h \in \mathcal{N}$, $r_h = 1/n$ and $a_h = 1/p_h \times \min_{i\in\mathcal{N}/(p_i)}$, we easily get,

$$K = \frac{c}{n}\binom{n}{c} n^c \left(\min_{i\in\mathcal{N}}(p_i)\right)^c$$

and, for every $A \in S$,

$$\pi_A = \frac{1}{\binom{n}{c}},$$

which leads, for every $\ell \in \mathcal{N}$, to

$$\gamma_\ell = \sum_{A\in S_\ell} \pi_A = \frac{|S_\ell|}{\binom{n}{c}} = \frac{c}{n}.$$

∎

The following corollary summarizes the analysis.

Corollary *5:* Algorithm 1 implements a Byzantine-tolerant node sampling service if

$$\forall j \in \mathcal{N}, a_j = \frac{\min_{i\in\mathcal{N}}(p_i)}{p_j} \text{ and } r_j = \frac{1}{n}.$$

Proof: Let any correct node i run Algorithm 1. By assumption, at any discrete time t, every node j in $\mathcal{N}$ has probability $p_j > 0$ to feed Algorithm 1. Thus, $a_j > 0$ for every $j \in \mathcal{N}$. From Theorem 4, when t tends to infinity, any node j has probability $\gamma_j = c/n$ to be in the sampler memory. From Algorithm 1, the output of the sampler is any node from Γ_i chosen with probability $1/c$. Thus j appears as output with probability $1/n$, which ensures the Uniformity property. Similarly, as $p_j a_j > 0$, j is guaranteed to appear in Γ_i with non null probability. Thus the Freshness property holds. ∎

B. Knowledge-free One-pass Strategy

We have proposed in Section IV an omniscient sampling strategy capable of locally unbiasing on the fly any unbounded and continuous stream σ that may have been arbitrarily ordered and manipulated by a strong adversary. This strategy uses a constant amount of memory, and does not need to know ahead of time which node identifiers will appear in σ. However the strategy needs to know the size n of $\mathcal{N}$, and upon receipt of a data item j, its occurrence probability p_j in σ. Clearly both assumptions are unrealistic since the adversary may modify on the fly the occurrence probability of any node identifier in the stream by increasing the occurrence frequency of the ℓ node identifiers it manipulates.

In this section, we propose a strategy, called hereafter *knowledge-free strategy*, that makes no assumption with respect to the input stream σ. For each received j from σ, the proposed strategy selects the id that will be part of the output stream by solely relying on an estimation of both n and p_j. Both estimations are computed on the fly by using very few space and a small number of operations.

Specifically, the knowledge-free strategy uses one additional data structure with respect to the omniscient one, as illustrated in Figure 2. This data structure is the *Count-Min Sketch matrix* $\hat{F}$ proposed by Cormode and Muthukrishnan [23]. Matrix $\hat{F}$ is built on the fly and provides at any time, and for each j read from σ, an approximation of the number of times j has appeared in σ from the inception of the stream. For self-containment reasons, we briefly describe how $\hat{F}$ is built.

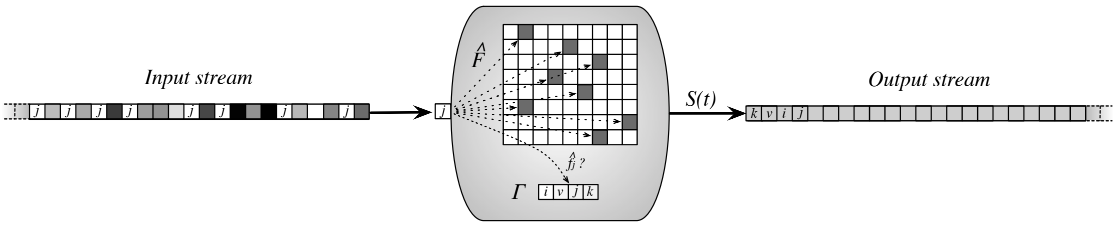


Fig. 2. Sampling component of node $i \in \mathcal{N}$.

Algorithm 2: Estimating the Frequency of Items in the Input Stream (Count-Min Sketch algorithm [23])

```
   Input: An input stream $\sigma$; $\delta$ and $\varepsilon$ settings;
   Output: The estimate $\hat{f}_j$ for the frequency of any item $j$ read
           from the input stream
1  $s \leftarrow \lceil \log(1/\delta) \rceil$;
2  $k \leftarrow \lceil e/\varepsilon \rceil$;
3  $\hat{F}[1..s][1..k] \leftarrow 0$;
4  Choose $s$ 2-universal hash functions
   $h_1..h_s : \{1..2^r\} \to \{1..k\}$;
5  for $j \in \sigma$ do
6      for $s = 1$ to $s$ do
7          $\hat{F}[s][h_s(j)] \leftarrow \hat{F}[s][h_s(j)] + 1$;
8  Upon query of Estimate($f_j$) return
   $\hat{f}_j = \min_{1 \le s \le t} \hat{F}[s][h_s(j)]$;
```

Algorithm 3: Knowledge-free Node Sampling Strategy run at any correct node $i \in \mathcal{N}$

```
   Input: An arbitrary input stream $\sigma_i$;
   Output: A modified output stream $\sigma'_i$;
   Data: $\Gamma_i$ a set of maximum size $c$;
   Data: $\hat{F}$ the Count-min Sketch matrix
1  cobegin on $\sigma_i$
2      execute Algorithm 2;
3  cobegin on $\sigma_i$
4      for $j \in \sigma_i$ do
5          $\hat{f}_j \leftarrow$ Estimate($f_j$);
6          $min_\sigma \leftarrow \min_{1 \le s \le t} \min_{1 \le r \le k} \hat{F}[s][r]$;
7          if $|\Gamma_i| < c$ then
8              $\Gamma_i \leftarrow \Gamma_i \cup \{j\}$;
9          else
10             with probability $a_j = \frac{min_\sigma}{\hat{f}_j}$ do
11                 choose $k$ from $\Gamma_i$ with probability $r_k = \frac{1}{c}$;
12                 $\Gamma_i \leftarrow (\Gamma_i \setminus \{k\}) \cup \{j\}$;
13         choose $k'$ from $\Gamma_i$ with probability $\frac{1}{c}$;
14         write $k'$ in the output stream;
```

1) Estimating the frequency of each element in the stream: For any item j in the input stream σ, the algorithm proposed by Cormode and Muthukrishnan [23] outputs an estimation $\hat{f}_j$ of the number of times j has occurred in the stream so far. The error of the estimator in answering a query for $\hat{f}_j$ is within a factor of ε with probability δ. The estimation is computed by maintaining a two-dimensional array $\hat{F}$ of $k \times s$ counters with $k = \lceil e/\varepsilon \rceil$ and $s = \lceil \log_2(1/\delta) \rceil$, and by using a collection of 2-universal hash functions $\{h_1, \ldots, h_s\}$ (where e is the base of the natural logarithm function ln). Each time an item j is read from the input stream, this causes one counter per line to be incremented, *i.e.*, $\hat{F}[v][h_v(j)]$ is incremented for all $v \in [s]$. When a query is issued to get an estimate $\hat{f}_j$ of the frequency of j (*i.e.*, the number of occurrences of j read so far from the stream), the returned value corresponds to the minimum among the s values of $\hat{F}[v][h_v(j)]$ $(1 \le v \le s)$. Algorithm 2 presents the pseudo-code of the Count-Min Sketch algorithm. Space required by Algorithm 2 is proportional to $\frac{1}{\varepsilon} \log_2 \frac{1}{\delta}$, and the update time per element is significantly sublinear in the size of the sketch [23], which makes this algorithm fully adapted to our context.

2) The knowledge-free strategy: The knowledge-free strategy is a simple extension of the omniscient one, where the insertion probability a_j for any received $j \in \sigma$ is computed by using the estimation $\hat{f}_j$ provided by Algorithm 2. Algorithm 3 presents its pseudo-code. Note that the instructions cobegin at lines 1 and 3 mean that codes of Algorithm 2 and lines (4-14) are executed in parallel (at any discrete time t the first id of σ_i is read by both codes). We analyze in Section V the resilience of Algorithm 3 against representative attacks, and evaluate in Section VI from extensive simulations the quality of the knowledge-free strategy w.r.t. the omniscient one.

V. The reason why the knowledge-free strategy is robust against collusion of malicious nodes

As previously said, we suppose that the adversary has enough resources to generate a large number of node identifiers, and to judiciously inject them in the input stream σ of any correct node, in order to prevent the sampler service of this node to output a uniform stream. In this section we derive the minimum number of identifiers the adversary has to generate to subvert the node sampling service.

We have shown in the previous section that whatever the power of the adversary, the omniscient strategy is capable of unbiasing any manipulated input stream. Indeed, we have shown with Corollary 5 that there exist two positive vectors $(a_j)_{j \in \mathcal{N}}$ and $(r_j)_{j \in \mathcal{N}}$ — respectively describing the insertion probability of node identifiers $j \in \mathcal{N}$ in Γ and their removal one from it —, which guarantee that given the occurrence probability $(p_j)_{j \in \mathcal{N}}$ of the node identifiers in the input stream, the sampler outputs a stream that meets both Uniformity and Freshness properties. Thus from Algorithm 3, the only latitude given to the adversary in biasing the output stream of any correct node is to increase the error made on the estimations $\hat{f}_j$ with $j \in \mathcal{N}$. By construction of Algorithm 2, each received element j is mapped to exactly one entry in each row of matrix

$\hat{F}$, and each of these entries is incremented by one. Thus to disrupt the estimation of any $\hat{f}_j$, the adversary has to generate sufficiently many node identifiers $o_1, \ldots, o_\ell$ such that for all $v \in [s]$, there exists $i \in [\ell]$ such that $h_v(o_i) = h_v(j)$. Recall that the s hash functions are locally chosen, thus the adversary cannot know which identifiers map to $h_1(j), \ldots, h_s(j)$. By injecting numerous times these node ids $o_1, \ldots, o_\ell$, the estimation $\hat{f}_j$ will be arbitrarily overestimated, and thus, by Algorithm 3, j will recur in the output stream with an arbitrary smaller frequency. We call this attack a *targeted attack*.

Whereas the goal of the adversary is to bias the frequency of a single targeted (correct) node identifier (refer as j in the argumentation above), the knowledge-free strategy forces the adversary to inject a series of node identifiers $o_1, \ldots, o_\ell$ that will also bias the frequency estimation of other node identifiers that are mapped to the same entries (*i.e.*, the ones mapped by $o_1, \ldots, o_\ell$). Thus, the adversary will blindly bias the frequency estimation of several node identifiers, including its owns. In the worst case, these additional bias inflicted to these other nodes will be at most as large as the one caused to j. In Section V-B, we will study the gap that exists between the effort needed by the adversary to perform a targeted attack and the one needed to bias all the frequencies estimation (of both correct and malicious nodes from what has been said above). We will refer to the latter attack as the *flooding attack*.

We now analyze the minimum effort that needs to be exerted by the adversary to make a targeted attack successful with probability $1 - \eta_T$ where $\eta_T < 1$.

A. Analysis of the effort needed to make a targeted attack successful

We model a targeted attack as a urn problem, where each entry of $\hat{F}$ is modeled as an urn and each received distinct node identifier as a ball. Consider a set of k urns initially empty in which we throw balls, one by one, according to the uniform distribution (by definition of 2-universal hash functions, each ball has an equal probability to be thrown in any of the k urns – *cf* Section III). We denote by N_ℓ the number of non empty urns at time ℓ, *i.e.*, just after the throwing of the ℓ-th ball and we consider the integer L_k which counts the number of balls needed to get a collision with a probability greater than $1-\eta_T$. More formally, for a given value of k and $\eta_T \in (0, 1)$, we have

$$L_k = \inf\{\ell \geq 2 \mid \mathbb{P}\{N_\ell = N_{\ell-1}\} > 1 - \eta_T\}.$$

In the knowledge-free algorithm, the previous experiment is executed identically and independently in s sets of k urns. At each time, we throw in parallel s balls, one in each set of k urns. For $i \in [s]$, the random variable $N_\ell^{(i)}$ counts the number of non empty urns among the i-th set of k urns at time ℓ and we consider the integer $L_{k,s}$ which counts the number of balls needed to get a collision in each set of k urns, with a probability greater than $1 - \eta_T$. We thus have in particular $L_k = L_{k,1}$. More formally, for given values of k, s and $\eta_T \in (0, 1)$, integer $L_{k,s}$ is defined by

$$\inf\{\ell \geq 2 \mid \mathbb{P}\{N_\ell^{(1)} = N_{\ell-1}^{(1)}, \ldots, N_\ell^{(s)} = N_{\ell-1}^{(s)}\} > 1 - \eta_T\}.$$

Since the s experiments in parallel are identical and independent, the random variables $N_\ell^{(1)}, \ldots, N_\ell^{(s)}$ are, for each $\ell \geq 1$, independent and identically distributed. It is thus sufficient to consider a single set of k urns and $L_{k,s}$ is then given by

$$L_{k,s} = \inf\{\ell \geq 2 \mid (\mathbb{P}\{N_\ell = N_{\ell-1}\})^s > 1 - \eta_T\}. \quad (2)$$

The random variable N_ℓ takes its values in the set $\{1, \ldots, k \wedge \ell\}$, where $k \wedge \ell$ denotes the minimum between k and ℓ. The distribution of N_ℓ is given, for every $k \geq 1$ and $\ell \geq 1$, by the following theorem which uses the Stirling numbers $S(\ell, i)$ of the second kind. These numbers are defined, for $\ell \geq 1$ and $i = 1, \ldots, \ell$, by the relations $S(1, 1) = 1$ and

$$S(\ell, i) = S(\ell-1, i-1)\mathbf{1}_{\{i \neq 1\}} + iS(\ell-1, i)\mathbf{1}_{\{i \neq \ell\}}. \quad (3)$$

It is well-known that this recursion leads to the explicit formula

$$S(\ell, i) = \frac{1}{i!}\sum_{h=0}^{i}(-1)^h \binom{i}{h}(i-h)^\ell. \quad (4)$$

Theorem *6:* For every $k \geq 1$, $\ell \geq 1$ and $i = 1, \ldots, k \wedge \ell$, we have

$$\mathbb{P}\{N_\ell = i\} = \frac{S(\ell, i)k!}{k^\ell (k-i)!}.$$

Proof: The relation is true for $\ell = 1$ since $\mathbb{P}\{N_1 = 1\} = 1$. For $\ell = 2, \ldots, k$ and $i = 1, \ldots, \ell$, we have

$$\begin{aligned}\mathbb{P}\{N_\ell = i\} &= \mathbb{P}\{N_\ell = i \mid N_{\ell-1} = i-1\}\mathbb{P}\{N_{\ell-1} = i-1\}\mathbf{1}_{\{i \neq 1\}} \\ &\quad + \mathbb{P}\{N_\ell = i \mid N_{\ell-1} = i\}\mathbb{P}\{N_{\ell-1} = i\}\mathbf{1}_{\{i \neq \ell\}} \\ &= \frac{k-i+1}{k}\mathbb{P}\{N_{\ell-1} = i-1\}\mathbf{1}_{\{i \neq 1\}} \\ &\quad + \frac{i}{k}\mathbb{P}\{N_{\ell-1} = i\}\mathbf{1}_{\{i \neq \ell\}}.\end{aligned}$$

Suppose that the relation is true for integer $\ell - 1$, *i.e.*, suppose that for every $i = 1, \ldots, \ell - 1$, we have

$$\mathbb{P}\{N_{\ell-1} = i\} = \frac{S(\ell-1, i)k!}{k^{\ell-1}(k-i)!}.$$

We then have, for $i = 1, \ldots, \ell$,

$$\begin{aligned}\mathbb{P}\{N_\ell = i\} &= \frac{k-i+1}{k}\frac{S(\ell-1, i-1)k!\mathbf{1}_{\{i \neq 1\}}}{k^{\ell-1}(k-i+1)!} \\ &\quad + \frac{i}{k}\frac{S(\ell-1, i)k!\mathbf{1}_{\{i \neq \ell\}}}{k^{\ell-1}(k-i)!} \\ &= \frac{S(\ell-1, i-1)k!\mathbf{1}_{\{i \neq 1\}}}{k^\ell (k-i)!} + \frac{iS(\ell-1, i)k!\mathbf{1}_{\{i \neq \ell\}}}{k^\ell (k-i)!} \\ &= \frac{\left[S(\ell-1, i-1)\mathbf{1}_{\{i \neq 1\}} + iS(\ell-1, i)\mathbf{1}_{\{i \neq \ell\}}\right]k!}{k^\ell (k-i)!}\end{aligned}$$

Relation (3) then gives the desired result.

For $\ell > k$, the term $\mathbf{1}_{\{i \neq \ell\}}$ is equal to 1. Actually, we have for $\ell > k$ and $i = 1, \ldots, k$,

$$\begin{aligned}\mathbb{P}\{N_\ell = i\} =& \frac{k-i+1}{k}\mathbb{P}\{N_{\ell-1} = i-1\}\mathbf{1}_{\{i \neq 1\}} \\ &+ \frac{i}{k}\mathbb{P}\{N_{\ell-1} = i\}.\end{aligned}$$

In the same way, this recursion leads, for $i = 1, \ldots, k$, to

$$\mathbb{P}\{N_\ell = i\} = \frac{S(\ell, i)k!}{k^\ell (k-i)!},$$

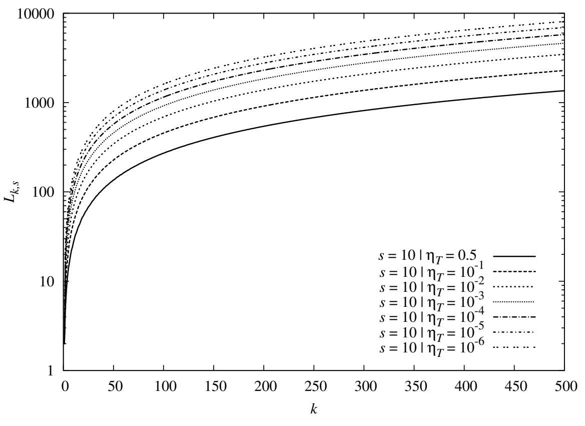


Fig. 3. Number of distinct malicious node identifiers $L_{k,s}$ as a function of the number of columns k and rows s of Matrix $\hat{F}$, and η_T.

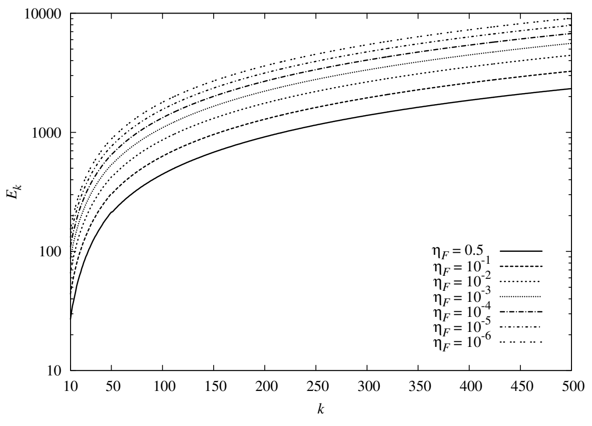


Fig. 4. Number of distinct malicious node identifiers E_k as a function of the number of columns k of Matrix $\hat{F}$, and η_F.

which completes the proof. ■

We are now able to compute, for every $\ell \geq 2$, the probabilities $\mathbb{P}\{N_\ell = N_{\ell-1}\}$. We have

$$\begin{aligned}\mathbb{P}\{N_\ell = N_{\ell-1}\} &= \sum_{i=1}^{k\wedge(\ell-1)} \mathbb{P}\{N_\ell = i \mid N_{\ell-1} = i\}\mathbb{P}\{N_{\ell-1} = i\} \\ &= \frac{1}{k}\sum_{i=1}^{k\wedge(\ell-1)} i\mathbb{P}\{N_{\ell-1} = i\} \\ &= \frac{E(N_{\ell-1})}{k}\end{aligned}$$

Figure 3 gives the number of distinct node identifiers $L_{k,s}$ (as defined in Relation 2) that the adversary has to inject to bias the identifier of at least one correct node. Recall that parameters k and s of Algorithm 2 are common knowledge (except the random local coins) and thus the adversary is capable of deriving $L_{k,s}$ according to the desired probability η_T. $L_{k,s}$ is linear in k and sublinear in s and η_T which explains why attacking a single node requires a significant number of distinct malicious node identifiers. For instance, when $k = 50$ and $s = 10$, the adversary has to inject in the input stream 150 distinct node identifiers to have no more than 50% of chance to get its targeted attack successful. On the other hand, with the same settings of k and s, 571 distinct node identifiers need to be injected to guarantee with probability 0.9999 a successful targeted attack.

Note that this analysis, as well as the one presented in Section V-B, derives the minimum number of distinct identifiers that need to be injected by the adversary in σ to bias the output stream. It does not consider the recurrence at which these identifiers must appear in the input stream σ. As said in Section III, the effort required by an adversary to bias the output stream is not in the repeated injection of node identifiers in σ but rather on the cost of creation of these identifiers. Indeed, to own an identifier, a node typically needs to interact with a central authority to receive a certificate assessing the validity and integrity of the identifier. The impact at which node identifiers recur in the input stream is analyzed in Section VI.

B. Analysis of the effort needed to make a flooding attack successful

We now analyze the minimum effort that needs to be exerted by the adversary to make a flooding attack successful with probability $1 - \eta_F$ where $\eta_F < 1$. As for the targeted attack, we model this attack as a urn problem, where as previously, each entry is modeled as an urn and each received distinct node identifier as a ball.

Let U_k be the number of balls needed in order to obtain all the k urns occupied, *i.e.*, with at least one ball. It is easily checked that $\mathbb{P}\{U_1 = 1\} = 1$ and that, for $\ell \geq k \geq 2$, we have

$$U_k = \ell \Longrightarrow N_{\ell-1} = k - 1.$$

We thus have

$$\begin{aligned}\mathbb{P}\{U_k = \ell\} &= \mathbb{P}\{U_k = \ell, N_{\ell-1} = k-1\} \\ &= \mathbb{P}\{U_k = \ell \mid N_{\ell-1} = k-1\}\mathbb{P}\{N_{\ell-1} = k-1\} \\ &= \frac{1}{k}\mathbb{P}\{N_{\ell-1} = k-1\}.\end{aligned}$$

From Theorem 6 and Relation (4), we get, for $k \geq 2$ and $\ell \geq k$,

$$\begin{aligned}\mathbb{P}\{U_k = \ell\} &= \frac{S(\ell-1, k-1)(k-1)!}{k^{\ell-1}} \\ &= \frac{1}{k^{\ell-1}}\sum_{r=0}^{k-1}(-1)^r\binom{k-1}{r}(k-1-r)^{\ell-1}.\end{aligned}$$

Finally, we consider the integer E_k which counts the number of balls needed to get a collision in all the $k \times s$ urns. Note that this number is independent of s as by definition, the s experiments in parallel are identical and independent. Thus, filling entirely a set of k urns leads to obtain all the s sets of k urns occupied. For given value of k and $\eta_F \in (0, 1)$, integer E_k is defined by

$$E_k = \inf\left\{\ell \geq k \,\middle|\, \sum_{i=k}^{\ell}\mathbb{P}\{U_k = i\} > 1 - \eta_F\right\}. \quad (5)$$

Figure 4 gives the number E_k of distinct ids the adversary has to inject in the input stream to introduce a bias on the

TABLE I. KEY VALUES OF $L_{k,s}$ AND E_k

Settings		η_T or η_F	$L_{k,s}$	E_k
k	s			
10 ($\varepsilon \sim 0.3$)	5 $\left(\delta \sim 10^{-2}\right)$	10^{-1}	38	44
10	5	10^{-4}	104	110
50 ($\varepsilon \sim 0.05$)	5	10^{-1}	193	306
50	10 $\left(\delta \sim 10^{-3}\right)$	10^{-1}	227	
50	40 $\left(\delta \sim 10^{-12}\right)$	10^{-1}	296	
50	5	10^{-4}	537	651
50	10	10^{-4}	571	
50	40	10^{-4}	640	
250 ($\varepsilon \sim 0.01$)	10	10^{-1}	1,138	1,617
250	10	10^{-4}	2,871	3,363

Note: Parameters ε and δ are respectively defined in Algorithm 2 as precision ($t = \lceil \log(1/\delta) \rceil$) and error ($k = \lceil e/\varepsilon \rceil$).

identifiers of all the correct nodes. This figure actually shows the upper bound of $L_{k,s}$ given k and $\eta_T = \eta_F$. Making a flooding attack successful with probability 0.9 when $k = 50$ requires around 300 malicious identifiers, while it requires around 650 node identifiers when the desired probability of success is equal to 0.9999.

The main results of both analyses are summarized in Table I. The most important one is that the effort that needs to be exerted by the adversary to subvert the sampling service can be made arbitrarily large by any correct node by just increasing the memory space of the sampler. The second one, which derives from the first one, is the absence of relationship between the effort of the adversary and the size of the population size. This astonishing result definitely guarantees the scalability of our node sampler service.

VI. Performance Evaluation of the Node Sampling Service

A. Settings of the Experiments

We have implemented both the omniscient and knowledge-free strategies of the node sampling service and have conducted a series of experiments on different types of streams and for different parameters settings. We have fed our algorithm with both real-world data sets and synthetic traces. Real data give a realistic representation of some existing systems, while the latter ones allow to capture phenomenon which may be difficult to obtain from real-world traces, and thus allow to check the robustness of our strategies. We have varied all the significant parameters of our algorithm, that is, the size m of the stream, the number of distinct data items n in each stream, the size of the local memory c, the number k of entries in each line of the count-min matrix, and the number s of lines of this matrix. For each parameters setting, we have conducted and averaged 100 trials of the same experiment, leading to a total of more than $100,000$ experiments for the evaluation of our algorithms. Real data have been downloaded from the repository of Internet network traffic [24]. We have used three large traces among the available ones. The first one represents one month of HTTP requests to the NASA Kennedy Space Center WWW server, the second one contains two weeks logs of HTTP requests to the Internet service provider ClarkNet WWW server (ClarkNet

TABLE II. STATISTICS OF REAL DATA TRACES.

Data trace	# ids (m)	# distinct ids (n)	max. freq.
NASA	1,891,715	81,983	17,572
ClarkNet	1,673,794	94,787	7,239
Saskatchewan	2,408,625	162,523	52,695

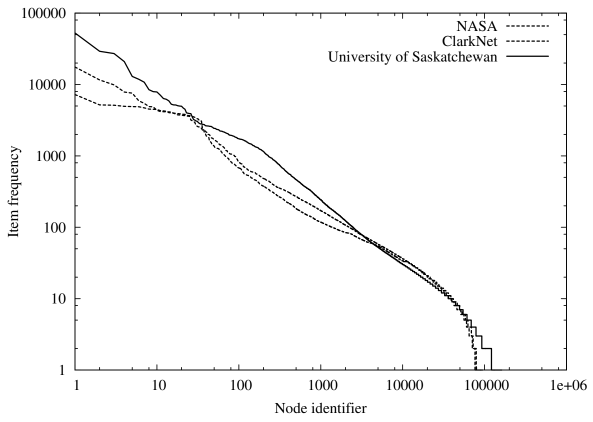


Fig. 5. Log-log scale distribution of frequencies for each real data trace.

is a full Internet access provider for the Metro Baltimore-Washington DC area), and the last one represents seven months of HTTP requests to the WWW server of the University of Saskatchewan, Canada. These data sets will be respectively referred to as NASA, ClarkNet, and Saskatchewan traces in the remaining of the paper. Table II presents some statistics of these data traces, in term of stream size (*cf.* "# ids"), population size in each stream (*cf.* "# distinct ids") and the number of occurrences of the most frequent id (*cf.* "max. freq."). Figure 5 illustrates the shape of each real data set distribution. Note that all these benchmarks share a Zipfian behavior, with a lower α parameter for the University of Saskatchwan.

In order to evaluate the accuracy of our algorithms, we measure the distance between the output streams and a uniform one. The distance we use is the *Kullback-Leibler (KL) divergence* [25], also called the relative entropy, which robustly measures the statistical difference between two data streams. Specifically, given v and w two frequency distributions, the Kullback-Leibler divergence is then defined as

$$\mathcal{D}_{KL}(v||w) = \sum_{i \in \mathcal{N}} p_i \log \frac{v_i}{w_i} = H(v,w) - H(v), \quad (6)$$

where $H(v) = -\sum v_i \log v_i$ is the (empirical) entropy of v and $H(v,w) = -\sum v_i \log w_i$ is the cross entropy from v to w. Note that when $v = w$, the KL divergence is minimal and is equal to zero. While all the distance measures in the Ali-Silvey distances are applicable to quantifying statistical differences between data streams, the KL divergence is particularly suited to our context since it gives rise to a small number of false positives when the two data streams are not significantly different.

B. Main Lessons drawn from the Experiments

We now present the main lessons drawn from these experiments. As said in the previous section, these experiments aimed at showing the impact of over-represented (malicious) node identifiers in the input stream of the sampler service.

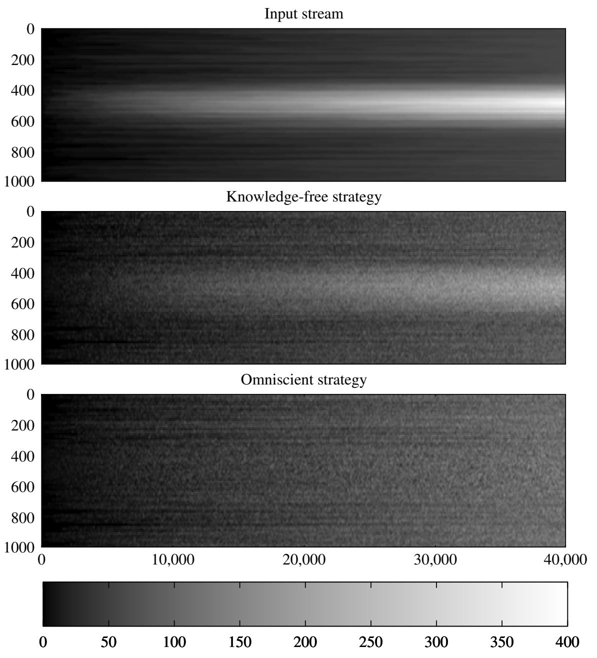
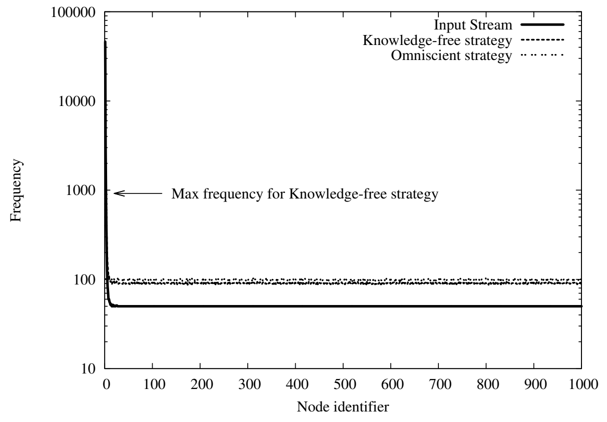


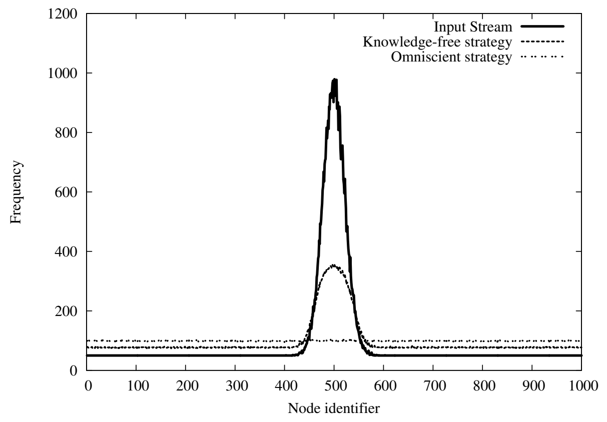
Fig. 6. Frequency distribution as a function of time. Settings: $m = 40,000$, $n = 1000$, $c = 15$, $k = 15$, $s = 17$.

Figure 6 presents a kind of isopleth in which the horizontal axis shows time, the vertical axis represents the node identifiers, and the body of the graph depicts the frequency of each node identifier (*i.e.*, the number of occurrences of each node identifier). A lighter color is representative of a very frequent node identifier. The figure at the top of Figure 6 represents the frequency of each node identifier in the input stream of the node sampler. This figure shows that at the inception of the stream, a few number of node identifiers have been received in the input stream which explains the dark color on the left. As time elapses, the number of received identifiers increases (up to 4000), and progressively the bias of the input stream appears: a small number of identifiers recur with a high frequency equal to 400, while the frequency of the other node identifiers sharply decreases. This is representative to a Poisson distribution with a small index. Now the two other figures represent the output of the node sampler run with respectively the knowledge-free strategy and with the omniscient one. Clearly the omniscient strategy succeeds in outputting a uniform stream, illustrated by a color that progressively and uniformly becomes lighter as the number of received identifiers augments. The knowledge-free strategy is not as performant as the omniscient one, nevertheless it succeeds in significantly decreasing the peak of high frequency identifiers with a very small memory (the sampling memory may contain up to 15 node identifiers, and the Count-Min data structure $\hat{F}$ is a 15×14 matrix.) w.r.t. the length m of the input stream.

Similarly to Figure 6, Figure 7 shows the frequency distribution of node identifiers in respectively the input and output streams as a function of the node identifiers. Figure 7(a) is representative of a particular attack, called *peak attack* in the following, in which the adversary injects $50,000$ times a single node identifier while all the other identifiers occur 50 times in the whole stream. Clearly the omniscient strategy fully



(a) The input stream is biased by a peak attack generated by Zipfian distribution with $\alpha = 4$.



(b) The input stream is biased by both targeted and flooding attacks generated by truncated Poisson distribution with $\lambda = \frac{n}{2}$.

Fig. 7. Frequency distribution as a function of node identifiers. Settings: $m = 100,000$, $n = 1,000$, $c = 10$, $k = 10$, $s = 5$.

tolerates such an attack by successfully outputting a uniform and fresh output stream. The knowledge-free strategy allows to reduce by a factor 50 the frequency peak with a small amount of memory space (the sampling memory contains 10 entries and the Count Sketch matrix contains 50 ones) with respect to the population size n and the length m of the input stream.

Figure 7(b) represents a scenario in which the adversary has successfully subverted the knowledge-free strategy by launching both a targeted and flooding attacks. Indeed, in this figure around 50 node identifiers are over represented in the input stream σ. Now from Table I, when $k = 10$, the minimum number of malicious node identifiers that need to be injected by the adversary to make a targeted attack successful with probability 0.9 and 0.9999 is respectively equal to $L_{k,t} = 38$ and $L_{k,t} = 104$, while it is equal to $E_k = 44$ and $E_k = 110$ to launch a flooding attack. Note that although both attacks are successful, the sampler service divides by 3 the frequencies of malicious node identifiers. Note that the omniscient strategy is fully robust against both attacks.

Figure 8 complements Figure 7(a) by showing the gain G_{KL} of the output stream over the input stream w.r.t. the Kullback-Leibler divergence. Specifically, let us denote by abuse of language $\mathcal{D}(\sigma,\mathcal{U})$ (respectively $\mathcal{D}(\sigma',\mathcal{U})$) as the

Kullback-Leiber divergence between a given input stream σ (respectively output stream σ') and a uniform one $\mathcal{U}$, then

$$G_{KL} = 1 - \frac{\mathcal{D}(\sigma',\mathcal{U})}{\mathcal{D}(\sigma,\mathcal{U})}.$$

This figure confirms the impressive robustness of the omniscient strategy, and shows that the pretty good resilience of the knowledge-free strategy against a peak attack in a very large system. Note that the inset graph in Figure 8 simply illustrates the KL distance between the input and output streams and the uniform stream.

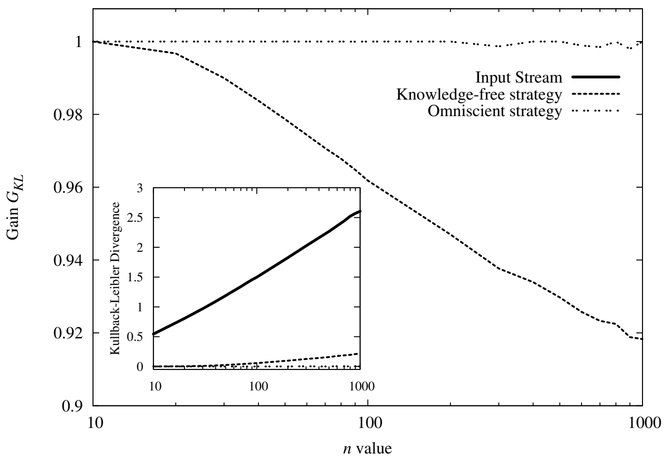


Fig. 8. G_{KL} as a function of the population size n. The input stream is biased by a peak attack generated by Zipfian distribution with $\alpha = 4$. Settings: $m = 100,000$, $k = 10$, $c = 10$, $s = 17$.

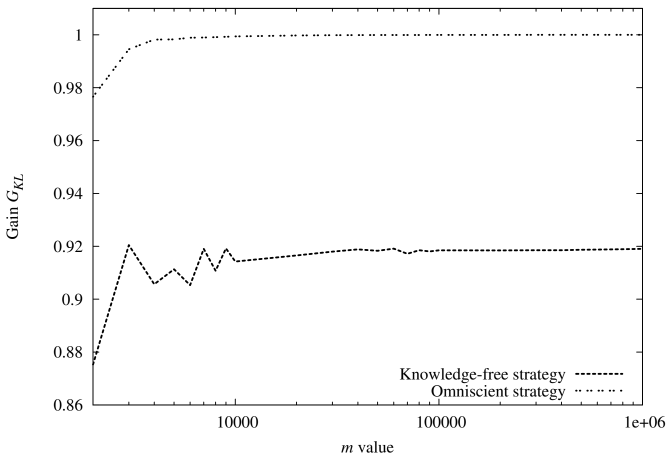
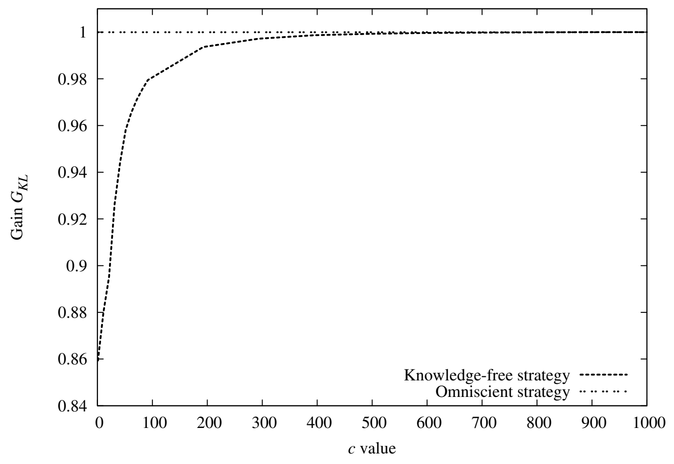
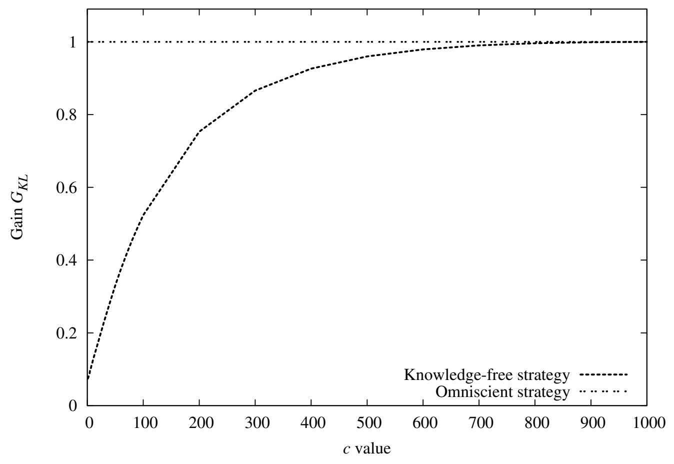


Fig. 9. G_{KL} as a function of the population size m. The input stream is biased by a peak attack generated by Zipfian distribution with $\alpha = 4$. Settings: $n = 1,000$, $k = 10$, $c = 10$, $s = 17$.

Figure 9 shows that both output streams (the one constructed with the omniscient strategy and the knowledge-free one) reach their stationary regime (*i.e.*, a uniform stream) very quickly. The first $3,000$ (non necessarily distinct and adversarially ordered) node identifiers are sufficient for the omniscient strategy to build a definitively unbiased output stream made of $1,000$ distinct ids, while it takes 3 times more elements of the input stream for the knowledge-free strategy to reach this stationary regime. This is still impressive given the very limited storage space of the latter strategy.



(a) The input stream is biased by a peak attack generated by Zipfian distribution with $\alpha = 4$.



(b) The input stream is biased by both targeted and flooding attacks generated by truncated Poisson distribution with $\lambda = \frac{n}{2}$.

Fig. 10. G_{KL} as a function of the number of entries c of the sampling memory Γ. Settings: $m = 100,000$, $n = 1,000$, $k = 10$, $s = 17$.

Figures 10(a) and 10(b) illustrate the intuitive fact that increasing the number of entries c of sampling memory is a very powerful defense against respectively peak attacks and targeted and flooding attacks. In the former case, the peak attack is completely masked by the knowledge-free strategy as soon as $c = 300$, while in the later case, both targeted and flooding attacks are masked for $c = 700$ (both with $k = 10$).

Figure 11 illustrates the gain obtained with the knowledge-free strategy as a function of the number of malicious node identifiers overrepresented in the input stream. Obviously, the strategy becomes vulnerable to malicious nodes once their number reaches 10% of the full population of the system. This behavior moves theory into practice according to the result presented in Table I.

Finally, Figure 12 illustrates the fact running the sampler service in a real environment provides very good results. Note the perfect behavior of the omniscient strategy.

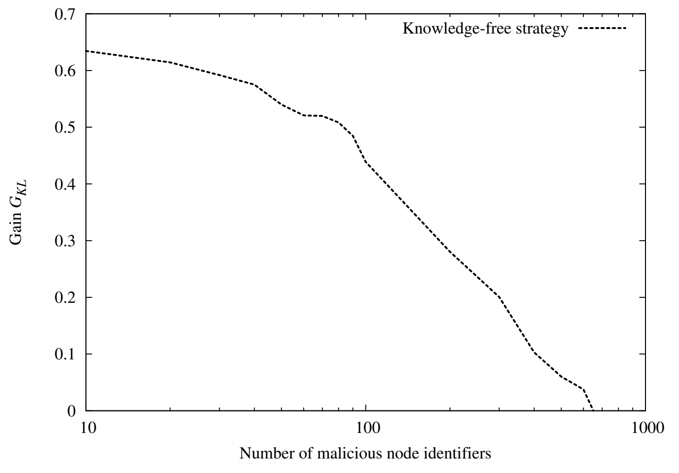


Fig. 11. G_{KL} as a function of the number of malicious node identifiers. Settings: $m = 100,000$, $n = 1,000$, $c = 50$, $k = 50$, $s = 10$.

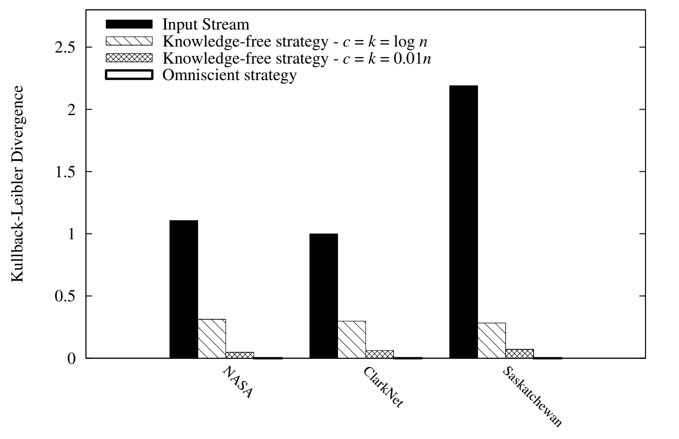


Fig. 12. Kullback-Leibler divergence between the different streams (input and output ones) and the uniform one.

VII. Conclusion

In this paper, we have studied the node sampling problem in presence of malicious nodes in a very large system by adopting a probabilistic approach. We have proposed and analyzed two online algorithms. The omniscient one is fully resilient to any attacks launched by a strong adversary, while the knowledge-free one is capable of drastically decreasing the impact of adversarial attacks by using very small memory space. As future work, we plan to analyze the transient behavior of the sampling service by using the results on weak lumpability in Markov chains described in [26] and [27].

References

[1] Q. Lv, P. Cao, E. Cohen, K. Li, and S. Shenker, "Search and Replication in Unstructured Peer-to-Peer Networks," in *Proceedings of the International Conference on Supercomputing (ICS)*, 2002, pp. 84–95.

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

[3] A. Demers, D. Greene, C. Hauser, W. Irish, J. Larson, S. Shenker, H. Sturgis, D. Swinehart, and D. Terry, "Epidemic algorithms for replicated database mangement," in *Proceedings of the 6th ACM Symposium on Principles of Distributed Computing (PODC)*, 1987.

[4] L. Massoulié, E. L. Merrer, A.-M. Kermarrec, and A. Ganesh, "Peer Counting and Sampling in Overlay Networks: Random Walk Methods," in *Proceedings of the 25th Annual Symposium on Principles of Distributed Computing (PODC)*. ACM Press, 2006, pp. 123–132.

[5] D. Stutzbach, R. Rejaie, N. Duffield, S. Sen, and W. Willinger, "On Unbiased Sampling for Unstructured Peer-to-Peer Networks," *IEEE/ACM Transactions on Networking*, vol. 17, no. 02, pp. 377–390, 2009.

[6] G. P. Jesi, A. Montresor, and M. van Steen, "Secure Peer Sampling," *Computer Networks*, vol. 54, no. 12, pp. 2086–2098, 2010.

[7] D. Liu, P. Ning, and W. Du, "Detecting Malicious Beacon Nodes for Secure Location Discovery in Wireless Sensor Networks," in *Proceedings of the 25th IEEE International Conference on Distributed Computing Systems (ICDCS)*, 2005.

[8] A. Singh, T.-W. Ngan, P. Druschel, and D. S. Wallach, "Eclipse Attacks on Overlay Networks: Threats and Defenses," in *Proc. of the 25th IEEE Intl Conference on Computer Communications (INFOCOM)*, 2006.

[9] J. Douceur and J. S. Donath, "The Sybil Attack," in *Proceedings of the 1rst International Workshop on Peer-to-Peer Systems (IPTPS)*, 2002, pp. 251–260.

[10] E. Bortnikov, M. Gurevich, I. Keidar, G. Kliot, and A. Shraer, "Brahms: Byzantine Resilient Random Membership Sampling," *Computer Networks*, vol. 53, pp. 2340–2359, 2009.

[11] E. Anceaume, Y. Busnel, and S. Gambs, "Uniform and Ergodic Sampling in Unstructured Peer-to-Peer Systems with Malicious Nodes," in *Proceedings of the 14th International Conference On Principles Of Distributed Systems (OPODIS)*, vol. 6490, 2010, pp. 64–78.

[12] Muthukrishnan, *Data Streams: Algorithms and Applications*. Now Publishers Inc., 2005.

[13] Z. Bar-Yossef, T. S. Jayram, R. Kumar, D. Sivakumar, and L. Trevisan, "Counting distinct elements in a data stream," in *Proceedings of the 6th International Workshop on Randomization and Approximation Techniques (RANDOM)*. Springer-Verlag, 2002, pp. 1–10.

[14] P. Flajolet and G. N. Martin, "Probabilistic counting algorithms for data base applications," *Journal of Computer and System Sciences*, vol. 31, no. 2, pp. 182–209, 1985.

[15] D. M. Kane, J. Nelson, and D. P. Woodruff, "An optimal algorithm for the distinct element problem," in *Proceedings of the Symposium on Principles of Databases (PODS)*, 2010.

[16] N. Alon, Y. Matias, and M. Szegedy, "The space complexity of approximating the frequency moments," in *Proceedings of the twenty-eighth annual ACM symposium on Theory of computing (STOC)*, 1996, pp. 20–29.

[17] M. Charikar, K. Chen, and M. Farach-Colton, "Finding frequent items in data streams," *Theoretical Computer Science*, vol. 312, no. 1, pp. 3–15, 2004.

[18] A. Chakrabarti, G. Cormode, and A. McGregor, "A near-optimal algorithm for computing the entropy of a stream," in *In ACM-SIAM Symposium on Discrete Algorithms*, 2007, pp. 328–335.

[19] A. Lall, V. Sekar, M. Ogihara, J. Xu, and H. Zhang, "Data streaming algorithms for estimating entropy of network traffic," in *Proceedings of the joint international conference on Measurement and modeling of computer systems (SIGMETRICS)*. ACM, 2006.

[20] E. Anceaume and Y. Busnel, "An information divergence estimation over data streams," in *Proceedings of the 11th IEEE International Symposium on Network Computing and Applications (NCA)*, 2012.

[21] N. Lynch, *Distributed Algorithms*. Morgan Kaufmann Publishers, 1996.

[22] P. B. Godfrey, S. Shenker, and I. Stoica, "Minimizing churn in distributed systems," in *Proceedings of the ACM SIGCOMM*, 2006.

[23] G. Cormode and S. Muthukrishnan, "An improved data stream summary: the count-min sketch and its applications," *Journal of Algorithms*, vol. 55, no. 1, pp. 58–75, 2005.

[24] T. I. T. Archive, "http://ita.ee.lbl.gov/html/traces.html," Lawrence Berkeley National Laboratory, Apr. 2008.

[25] S. Kullback and R. A. Leibler, "On information and sufficiency," *The Annals of Mathematical Statistics*, vol. 22, no. 1, pp. 79–86, 1951. [Online]. Available: http://dx.doi.org/10.2307/2236703

[26] J. G. Keneny and J. L. Snell, *Finite Markov Chains*. Springer-V., 1976.

[27] G. Rubino and B. Sericola, "On weak lumpability in Markov chains," *Journal of Applied Probability*, vol. 26, 1989.