A Distributed Information Divergence Estimation over Data Streams

Emmanuelle Anceaume, Yann Busnel

A Distributed Information Divergence Estimation over Data Streams

Emmanuelle Anceaume and Yann Busnel

Abstract—In this paper, we consider the setting of large scale distributed systems, in which each node needs to quickly process a huge amount of data received in the form of a stream that may have been tampered with by an adversary. In this situation, a fundamental problem is how to detect and quantify the amount of work performed by the adversary. To address this issue, we propose a novel algorithm AnKLe for estimating the Kullback-Leibler divergence of an observed stream compared with the expected one. AnKLe combines sampling techniques and information-theoretic methods. It is very efficient, both in terms of space and time complexities, and requires only a single pass over the data stream. We show that AnKLe is an (ε, δ)-approximation algorithm with a space complexity $\tilde{\mathcal{O}}\left(\frac{1}{\varepsilon}+\frac{1}{\varepsilon^2}\right)$ bits in "most" cases, and $\tilde{\mathcal{O}}\left(\frac{1}{\varepsilon}+\frac{n-\varepsilon^{-1}}{\varepsilon^2}\right)$ otherwise, where n is the number of distinct data items in a stream. Moreover, we propose a distributed version of AnKLe that requires at most $\mathcal{O}\left(r\ell\left(\log n+1\right)\right)$ bits of communication between the ℓ participating nodes, where r is number of rounds of the algorithm. Experimental results show that the estimation provided by AnKLe remains accurate even for different adversarial settings for which the quality of other methods dramatically decreases.

Index Terms—Data Stream; Kullback-Leibler Divergence; Randomized approximation algorithm; Byzantine Adversary; Performance Analysis

1 Introduction

The main objective of this paper is the online estimation of the similarity between observed data streams and expected (*i.e.* idealized) ones in order to detect in real time the presence of intrusions in network traffic. More precisely, we propose a distributed algorithm that approximates with guaranteed error bounds in a single pass and with both a small amount of storage memory and processing capacity, the relative entropy between massive and high frequency distributed sequences of data. This works perfectly fits the IP network traffic context, however it could be applied to any other data issued from distributed applications such as sensors readings. The problem of detecting changes in a data stream is similar to the problem of identifying patterns that do not conform to the expected behavior, which has been an active area of research for many decades. A comprehensive survey of these techniques, their advantages and their drawbacks is given in [1]. A common feature of these techniques is their space complexity and their computational cost, as they rely on full space algorithms for analyzing their data.

Given our settings — the real time monitoring of network traffic with little capacities in terms of storage and processing — relying on full space algorithms for analyzing input data is not feasible. In contrast, two main approaches exist to monitor in real time massive data streams. The first one consists in regularly sampling the input streams so that only a limited amount of data items is locally kept [2], [3], [4]. This allows to exactly compute functions on these samples. However, accuracy of this computation fully depends on the volume of data that has been sampled and their locations in the stream. Worse, an adversary may easily take advantage of the sampling policy to hide its attacks among packets that are not sampled, or in a way that prevents its "malicious" packets to be correlated. In contrast, the streaming approach consists in scanning each piece of data of the input stream on the fly, and in locally keeping only compact synopses or sketches that contain the most important information about data items. This approach enables to derive some data streams statistic with guaranteed error bounds without making any assumptions on the order in which data items are received at nodes (*i.e.*, data items ordering can be manipulated by an omnipotent adversary [5]). Most of the research done so far with this approach has 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 [6], [7], [8], the frequency moments [9], the most frequent data items [9], [10], or the entropy of the stream [11], [12]. Note that in [13], the authors propose a characterization of the information divergences that are not sketchable. They have proven that any distance that has not "norm-like" properties is not sketchable.

- *E. Anceaume is with the Centre national de la recherche scientifique (CNRS), IRISA, France.*
  E-mail: Emmanuelle.Anceaume@irisa.fr
- *Y. Busnel is with the LINA, Department of Computer Sciences, Université de Nantes, France.*
  E-mail: Yann.Busnel@univ-nantes.fr

- *Shorter preliminary versions of this paper appeared in the* 9th European Dependable Computing Conference (EDCC 2012)*, May 2012 and in the* 11th IEEE International Symposium on Network Computing and Applications (IEEE NCA 2012)*, August 2012.*

On the other hand, very few works have tackled the distributed streaming model, also called the functional monitoring problem [14], which combines features of both the streaming model and communication complexity models. As in the streaming model, the input data is read on the fly, and processed with a minimum workspace and time. In the communication complexity model, each node receives an input data stream, performs some local computation, and communicates only with a coordinator who wishes to continuously compute or estimate a given function of the union of all the input streams. The challenging issue in this model is for the coordinator to compute the given function by minimizing the number of communicated bits [14], [15], [16]. Cormode *et al.* [14] pioneer the formal study of functions in this model by focusing on the estimation of the first three frequency moments F_0, F_1 and F_2 [9]. Arackaparambil *et al.* [15] consider the empirical entropy estimation [9] and improve the work of Cormode by providing lower bounds on the frequency moments, and finally distributed algorithms for counting at any time t the number of items that have been received by a set of nodes from the inception of their streams have been proposed in [17], [18].

In this paper, we go a step further by presenting a distributed algorithm that approximates the relative entropy among ℓ data streams by using $\mathcal{O}\left(\log n + \frac{1}{\varepsilon^2} + \left(\frac{1}{\varepsilon} + \frac{\log m}{\varepsilon^2}\log\frac{1}{\delta}\right)(\log n + \log m)\right)$ bits of space for each ℓ nodes, and $\ell{+}1$ floats for the coordinator, where n is the number of distinct items, and m is the largest data stream observed at one of the ℓ nodes. This distributed algorithm relies on an estimator called AnKLe proposed in a previous work [19], [20]. AnKLe estimates the Kullback-Leibler (KL) divergence between a stream and a uniform one streams. This divergence can be viewed as an extension of the Shannon entropy and is often referred to as the relative entropy [21]. Finally, we guarantee that the number of bits exchanged between the nodes in the system is upper bounded by $\mathcal{O}(r\ell(\log n + 1))$. To the best of our knowledge, such a work has never been done so far.

The paper is organized as follows. First, Section 2 describes the system model and some background on information theory. Section 3 presents the buildings blocks of our algorithm AnKLe, while Section 4 analyses its quality. Section 5 extends AnKLe to a distributed setting and evaluates its quality. In Section 6, we empirically evaluate the accuracy of the estimation provided by AnKLe. Finally, we conclude in Section 7. For space constraint reasons, extended related work on the estimation of the relative entropy of data streams is presented in the companion paper [22].

2 Model and Background

2.1 Model

We present the computation model under which we analyze our algorithms and derive lower and upper bounds. We consider a set of ℓ nodes $S^{(1)}, \ldots, S^{(\ell)}$ such that each node $S^{(i)}$ receives a large sequence $\sigma^{(i)}$ of TCP/IP packets [23] modeled as data items or symbols $u, v, w, \ldots$ drawn from a large universe N. For a given stream $\sigma^{(i)}, i \in \{1,..,\ell\}$, we denote by $n^{(i)}$ the number of distinct items that appear in $\sigma^{(i)}$ and by $m^{(i)}$ the size of $\sigma^{(i)}$, that is the number of items in it[1]. Items arrive regularly and quickly, and due to memory constraints (*i.e.*, nodes can locally store only a small amount of information with respect to the size of their input stream and perform simple operations on them), items must be processed sequentially and in an online manner. Nodes cannot communicate among each other. On the other hand, there exists a specific node, called *coordinator*, with which each node may communicate. We assume that communication is instantaneous. We refer the reader to [24] for a detailed description of data streaming models and algorithms.

Model of the Adversary: We suppose that the adversary is omnipotent in the sense that it may actively tamper with the data stream of any node by observing, inserting, dropping or re-ordering items of their input stream. We suppose that the algorithm used by a node to estimate the divergence is public knowledge (*i.e.*, to avoid some kind of security by obscurity), however the adversary has not access to the local random values generated during the algorithm (*e.g.*, the random choice of hash functions from a collection of 2-universal hash functions, see Section 3).

2.2 Preliminaries

We first present notations (summarized in Table 1) and background on data streams analysis that make this paper self-contained. Note that some of these notations are formally defined in the remaining of the paper.

2.2.1 Entropy

Intuitively, the entropy is a measure of the randomness of a data stream σ. The entropy H_σ is minimum (*i.e.*, equal to zero) when all the items in the stream are the same, and it reaches its maximum when all the items in the stream are distinct. Specifically, we have $H_\sigma = -\sum_{u\in N} p_u \log p_u$, where for each $u \in N$, $p_u = m_u/m$ with m_u representing the number of times the value u appears in the stream σ, and $m = \sum_{u\in N} m_u$ being the size of the stream. Thereafter, we will denote by "log" the logarithm in base 2, and by convention, we pose $0\log 0 = 0$. Without loss of generality, we assume that items are ordered so that $m_1 \geq m_2 \geq \ldots \geq m_n$. Note that the number of times m_u item u appears in a stream is commonly called the frequency of item u. The norm of the entropy is defined as $F_H = \sum_{u\in N} m_u \log m_u$.

[1] Note that when a single stream is considered, notations $\sigma^{(i)}$, $n^{(i)}$ and $m^{(i)}$ are simply denoted σ, n and m, which will be the case up to Section 5.

2.2.2 Kullback-Leibler Divergence

The Kullback-Leibler (KL) divergence [25], also called the relative entropy, is a robust metric for measuring the statistical difference between two data streams. The KL-divergence is a member of a larger class of distances known as the Ali-Silvey distances [26]. Given two probability distributions p and q on N, the KL-divergence between p and q is defined as the expected value of the likelihood ratio with respect to q:

$$\mathcal{D}(p||q) = \sum_{u \in N} p_u \log \frac{p_u}{q_u} = H(p,q) - H(p),$$

where $H(p) = -\sum p_u \log p_u$ is the (empirical) entropy of p and $H(p,q) = -\sum p_u \log q_u$ is the cross entropy of p and q. As we use a logarithm in base 2, the divergence is measured in bits. It is zero if and only if $p = q$. Let $p^{(\mathcal{U})}$ be the uniform distribution corresponding to a uniform stream, and q be the probability distribution corresponding to the input stream. In the rest of this paper we consider $\mathcal{D}(q||p^{(\mathcal{U})})$ as a measure of the divergence between the current stream and the ideal one.

2.2.3 Frequency Moments

Frequency moments are important statistical tools that have been introduced by Alon *et al.* [9]. Computing frequency moments F_k allows to quantify the amount of skew in a data stream. For each $k \geq 0$, the k-th frequency moment F_k of σ is defined as $F_k = \sum_{u \in N} m_u^k$. Among the remarkable moments, F_0, F_1 and $''F_\infty''$ represent respectively the number of distinct elements in σ, the size of σ, and the most frequent item in σ [7].

2.2.4 2-universal Hash Functions

In the following, we intensively use hash functions randomly picked from a 2-universal hash functions family. A collection $\mathcal{H}$ of hash functions $h : \{1, \dots, M\} \to \{0, \dots, M'\}$ is said to be *2-universal* if for every two different items $x, y \in [M]$, $\mathbb{P}_{h \in \mathcal{H}}\{h(x) = h(y)\} \leq \frac{1}{M'}$, which is the probability of collision obtained if the hash function assigned truly random values to any $x \in [M]$.

2.2.5 Randomized (ε, δ)-approximation Algorithm

A randomized algorithm $\mathcal{A}$ is said to be an (ε, δ)-approximation of a function ϕ on σ if for any sequence of items in the input stream σ, $\mathcal{A}$ outputs $\hat{\phi}$ such that $\mathbb{P}\{\mid \hat{\phi} - \phi \mid > \varepsilon\phi\} < \delta$, where $\varepsilon, \delta > 0$ are given as parameters of the algorithm.

3 Detecting Adversarial Behaviors via the KL-divergence Estimation

3.1 Building Blocks

We describe three algorithms that form the building blocks of the AnKLe (Attack-tolerant eNhanced Kullback-Leibler divergence Estimator) algorithm. All these algorithms have been designed in the stream data model (*cf.* Section 2.1). For self-containment reasons, we briefly review these building blocks and describe their theoretical guarantees.

Notation	Meaning
ℓ	Number of data streams in the system (Section 2.1)
$S^{(i)}$	Site at which $\sigma^{(i)}$ is received (Section 2.1)
$\sigma^{(i)}$	Input stream (Section 2.1)
$m^{(i)}$	Size of stream $\sigma^{(i)}$ (Section 2.1)
$n^{(i)}$	Number of distinct data items in $\sigma^{(i)}$ (Section 2.1)
N	Domain of data items (Section 2.1)
$u, v, \dots$	Data items of N (Section 2.1)
$m_u^{(i)}$	Number of occurrences of data item u in $\sigma^{(i)}$ (Section 2.2)
$\mathcal{H}$	Collection of 2-universal hash functions (Section 2.2)
ε, δ	Quality parameters of the approximation algorithms (Section 2.2)
p, q	Probability distributions (Section 2.2)
$p^{(\mathcal{U})}$	Uniform probability distribution (Section 2.2)
F_k	Frequency moments of a stream (Section 2.2)
$H(p)$	Entropy of distribution p (Section 2.2)
F_H	Norm of the entropy (Section 2.2)
H_σ	Entropy of a stream σ (Section 2.2)
$H(p,q)$	Cross entropy of p relative to q (Section 2.2)
$\mathcal{D}(p\|\|q)$	Divergence of Kullback-Leibler between p and q (Section 2.2)
$M_1, \dots, M_\kappa$	Register used in the LogLog algorithm (see Section 3.1.2)
X, X^s	Estimators used in AnKLe algorithm (Section 3.2)
κ, c, s_1, s_2, k	Parameters of the AnKLe algorithm (Section 3.2)
K	Set of the most frequent data items in a stream (Section 4)
F_H^s	Norm of the entropy of the sparse data items' sub-stream (Section 4)
$Y_{\widehat{F}}$	Estimator of the norm of the entropy on frequent data items (Section 4)
Y_s	Estimator of the norm of the entropy on sparse data items (Section 4)
$\widehat{D_i}$	Estimator of $\mathcal{D}(q_{\sigma^{(i)}}\|\|p^{(\mathcal{U})})$ (Section 5)
r	Number of rounds in the distributed algorithm (Section 5)

Table 1
List of symbols and notations

3.1.1 Estimating the k^{th} Moment of a Stream

The AnKLe algorithm adopts the structure of the method proposed by Alon *et al.* [9] (called in the following the AMS algorithm), to approximate the KL-divergence of a stream. The AMS algorithm estimates the k-th frequency moment of a stream as follows. It computes a basic estimator which takes the form of a random variable X whose mean value is exactly equal to the k^{th} frequency moment of a stream and whose variance is very small. Specifically, X is defined as $X = m(g^k - (g-1)^k)$, where g is the exact number of times element v appears in the stream from a uniformly and randomly chosen position p (we have $a_p = v$) in the stream onwards. To improve the accuracy of the estimation, several independent basic estimators are computed on the stream (specifically $s_1 \times s_2$ basic estimators X_{ij}, for $1 \leq i \leq s_1$ and $1 \leq j \leq s_2$, for $s_1 \times s_2$ positions uniformly chosen at random in the stream σ), and the final estimator Y is set to be $Y = \mathrm{median}_{1 \leq j \leq s_2} \left(1/s_1 \sum_{i=1}^{s_1} X_{ij}\right)$.

Theorem 1 ([9]) *For any* $\varepsilon, \delta \in (0,1)$, *if* $s_1 \geq Var[X]/(\varepsilon^2 E[X]^2)$ *and* $s_2 = 4\log(1/\delta)$, *then* Y *is a* (ε, δ)*-approximation of* $E[X]$, i.e., $\mathbb{P}\{\mid E[X] - Y \mid > \varepsilon E[X]\} < \delta$.

3.1.2 Estimating the Number of Items

The problem of estimating the number of distinct elements has received a lot of attention in the data stream model. First, the seminal work of Flajolet et al [7] has shown that it is possible to compute such an estimate using only logarithmic space in n by relying on properties of hash functions. Afterwards, follow-up enhancements have improved the quality of F_0 [8], [6], [27]. A comprehensive survey describing the literature on distinct elements in the data stream model is presented by Gibbons in [28]. The Durand and Flajolet algorithm [27] (referred to as the LogLog algorithm in the following) and described in Figure 1 computes an estimation $\hat{F_0}$ of the number of distinct items F_0 (denoted as n in this paper) in a stream. This algorithm builds upon the approaches proposed in [7] and [9]. Briefly, it maintains a collection of κ registers $M_1, \ldots M_\kappa$, where κ a power of 2. It uses also a hash function $h(.)$ that maps each distinct item read from the stream to i with probability $2^{-(i+1)}$, where $i \in [0, L]$ and L is typically equal to 32. For each stream item v, the $\log_2 \kappa$ least significant bits of $h(v)$ are used to select one of the k registers M_i, and the maximum between the remaining $L - \log_2 \kappa$ bits of $h(v)$ and M_i are used as the value of M_i. Once all the items of the stream have been read, then the arithmetic mean $\frac{1}{\kappa}\sum_{j=1}^{k} M_j$ approximates $\log_2(n/\kappa)$ plus an additive bias. The estimate of n returned by the LogLog algorithm is $\hat{F_0} = \alpha_\kappa \kappa 2^{\frac{1}{\kappa}\sum M_j}$, where α_κ is a constant derived by the analysis [27] which allows to correct the systematic bias of the arithmetic mean in the asymptotic limit. Kane et al. [8] improve this algorithm by assuming a truly random hash function and κ-wise independent hash functions, which provides the following estimation quality.

Theorem 2 ([8]) *For any ε, their algorithm outputs $\hat{F_0}$ such that $\mathbb{P}\{|\hat{F_0} - F_0| > \varepsilon\} < \delta$ where $\delta = 2/3$. The worst-case running time for each input symbol is $\mathcal{O}(1)$, and the total space required by the algorithm is $\mathcal{O}(1/\varepsilon^2 + \log n)$ bits, which makes this algorithm optimal.*

3.1.3 Determining the Most Frequent Identifiers

The problem of determining the c most frequent items in a stream has also been studied extensively in the data stream literature. Thereafter, we describe a deterministic algorithm that outputs items that occur more than $\frac{m}{c}$ in a stream. This algorithm, due to Misra and Gries [29], maintains c counters such that for each counter, its key is the item read from the stream and its value is related to the frequency of items. Initially, all the counters are set to $(–, 0)$. Afterwards, when an item is read from the stream, if that item has already a counter associated to it, then this counter is incremented. If this is not the case and if there are still free counters available, then one of these free counters is allocated to this new item and its value is set to 1. Otherwise, all the allocated counters are decremented by one, and if after this operation some of them are equal to 0 then their keys are erased and the counters are released. The pseudo-code of the Misra Gries algorithm is presented in Figure 2.

Figure 1. LogLog algorithm

```
Input: An input stream $\sigma$; precision parameter $\kappa$;
Output: Estimate $\hat{F_0}$ in $\sigma$
Pick one 2-universal hash function $h : [1..N] \to [L]$;
$\kappa$ registers $M_j = 0$ with $\kappa > 64$ a power of 2;
$\alpha_\kappa = 0.39701$;
for $a_i \in \sigma$ do
    $v = a_i$;
    $x = h(v) \mod \kappa$;
    $b$=the largest $r \geq 0$ such that the $r$ rightmost bits in
    $\lfloor h(v)/\kappa \rfloor$ are all 0;
    $M_x = \max(M_x, b)$;
return $\hat{F_0} = \alpha_\kappa \kappa 2^{\frac{1}{\kappa}\sum_{j=1}^{\kappa} M_j}$;
```

Figure 2. Misra-Gries algorithm

```
Input: An input stream $\sigma$; a precision parameter $c$;
Output: The $c$ most frequent items in $\sigma$ and an estimate of their
        frequency
for $j \in [0..c]$ do $A[j] \leftarrow (\perp, \perp)$;
for $a_i \in \sigma$ do
    $v = a_i$;
    if $\exists u$ such that the item of $A[u]$ is $s$ then increment the
    count value of $A[u]$;
    else
        if $\exists u'$ such that $A[u'] = (\perp, \perp)$ then
            $A[u'] = (v, 1)$
        else for $i = 1$ to $c$ do
            Decrement the count of $A[i]$;
            if the count value of $A[i] = 0$ then
                $A[i] = (\perp, \perp)$
return $A$;
```

Theorem 3 ([29]) *The Misra and Gries algorithm with parameter c returns for any item u an estimate $\hat{m}_u$ such that $m_u - \frac{m}{c} \leq \hat{m}_u \leq m_u$ with $\mathcal{O}(c(\log m + \log n))$ bits of space.*

3.2 The AnKLe algorithm

This section presents AnKLe, the algorithm we propose for computing the KL-divergence of a stream. Our starting point is the re-writing of the KL-divergence as follows. From Definition 1, we have

$$\begin{aligned}\mathcal{D}(q_\sigma||p^{(\mathcal{U})}) &= \sum_{i=1}^{n} q_i \log(q_i) - \sum_{i=1}^{n} q_i \log\left(p_i^{(\mathcal{U})}\right) \\ &= \log(n) - \log(m) + \frac{1}{m}\sum_{i=1}^{n} m_i \log(m_i). \quad (1)\end{aligned}$$

Thus estimating the KL-divergence amounts in *(i)* estimating the number of distinct items in the stream (*i.e.*, F_0) in order to obtain a good approximation of $\log(n)$, and *(ii)* estimating $\sum_{i=1}^{n} m_i \log(m_i)$, which corresponds to the norm of the entropy F_H. While the first point is solved by relying on the KNW [8] algorithm, the second point is tackled by extending the approach proposed by Alon *et al.* [9] to deal with arbitrary distributions of items in the input stream.

Figure 3. AnKLe Algorithm

```
Input: An input stream $\sigma$ of length $m$, $k$ (number of frequent
       items), $\kappa$ (see Figure 1), $c$ (see Figure 2), $s_1, s_2$ (see
       Section 3.1.1)
Output: Estimation of $\mathcal{D}(q_\sigma||p^{(\mathcal{U})})$
1  Choose $s_1 \times s_2$ random integers in $[1..m]$;
2  for $u_1 \in [0..s_1], u_2 \in [0..s_2]$ do $S[u_1, u_2] \leftarrow (\perp, \perp)$;
3  for $a_j \in \sigma$ do
4      $v = a_j$;
5      Task $T_1$ :
6          $\hat{F}_0 \leftarrow$ Algorithm in Fig. 1 (KNW [8]) fed with $v$;
7      Task $T_2$ :
8          $\hat{F} \leftarrow$ Algorithm in Fig. 2 (Misra-Gries [29]) fed with $v$;
9      Task $T_3$ :
10         forall entries $i$ of matrix $S$ such that $(s_i, r_i) \neq (\perp, \perp)$ do
11             if $s_i = v$ then $r_i \leftarrow r_i + 1$;
12         if $j$ is one the $s_1 \times s_2$ random integers then
13             assign $(v, 1)$ to the first unused entry of $S$;
14 $\hat{F} \leftarrow$ the $k$ most frequent items $(s_i, r_i)$ of $\hat{F}$ and such that $r_i > e$;
15 forall entries $i$ of matrix $S$ do
16     if $(s_i, -) \in \hat{F}$ then $(s_i, r_i) \leftarrow (s_i, -)$ ;
17     else $(s_i, r_i) \leftarrow (s_i, m\,(r_i \log r_i - (r_i - 1)\log(r_i - 1)))$;
18 $Y_S \leftarrow \text{median}_{1 \le j \le s_2} \left(\frac{1}{s_1} \sum_{i=1}^{s_1} S_{ij}\right)$;
19 $Y_{\hat{F}} \leftarrow \sum_{(s_i, r_i) \in \hat{F}} r_i \log r_i$;
20 $p \leftarrow 1 - \max\left(0, \dfrac{\min\left(Y_s, Y_{\hat{F}}\right) - m}{10 \cdot m}\right)$;
21 return $\hat{D} = \log \hat{F}_0 - \log m + \frac{p}{m}\left(Y_S + Y_{\hat{F}}\right)$;
```

The pseudo-code of AnKLe is presented in Figure 3 and consists of two phases. The first one (lines 3–13) is executed upon reception of the items of the stream, while the second one (lines 14–21) is run when m items have been read from the stream. The first phase is composed of three tasks (T_1, T_2 and T_3), executed in parallel. Task T_1 (line 5) estimates the number of distinct items present in the stream, Task T_2 (line 7) identifies the k most frequent items in the stream, and Task T_3 samples random items in the stream in order to compute their exact frequency. Specifically, Task T_3 (lines 9–13) consists in running a sampling estimator X on the stream. The basic estimator $X = X_{i,j}$ is designed so that its mean value is equal to the norm of the entropy F_H and its variance is small. More precisely, we have

$$X = m(g \log g - (g-1)\log(g-1)), \tag{2}$$

where g is the random variable representing the number of occurrences of an item ℓ in the stream. This item ℓ is such that its position j in the stream is a random number in $[m]$. The random variable g counts the number of times ℓ appears in the stream from position j onwards. Formally, g is defined as $g =\mid \{j : j \geq \ell, a_j = a_\ell\} \mid$.

We can show as done in [9], [12], that the estimator X is unbiased (*i.e.*, its expectation is equal to F_H), *i.e.*,

$$E[X] = \frac{1}{m}\sum_{i=1}^{n}\sum_{j=1}^{m_i} m(j \log j - (j-1)\log(j-1)) = F_H. \tag{3}$$

To improve the accuracy of the estimation, $s_1 \times s_2$ such basic estimators X_{ij} (for $1 \leq i \leq s_1$ and $1 \leq j \leq s_2$) are used, each one sampling a random position in the stream. From the implementation point of view, tracking these estimators consists in storing $s_1 \times s_2$ counters, each one counting the number of occurrences of an item whose position has been randomly chosen in the stream. When item u is read from the input stream, if u has already one or more counters assigned to it then all these counters are incremented. In addition, if the position at which u has been read in the stream is one of the chosen locations, then a counter is assigned to u, and its value is set to 1. Thus for each of these "tracked" items, an exact count of their frequency is continuously maintained starting from a random position in the stream.

The post-processing phase of AnKLe algorithm estimates the KL-divergence of the input stream according to Relation (1). This phase is executed when m items have been read from the input stream. In this work, we suppose that m is a parameter of the algorithm, however by using techniques proposed in Chakrabarti *et al.* [11] we can extend our solution to streams whose size is *a priori* unknown. To accurately estimate the KL-divergence of the stream, one needs to cope with patterns in which a small number of items occur with a very high frequency with respect to the other items. When such patterns occur, the basic estimator X alone is unable to compute the norm of the entropy in bounded space [11]. Indeed, by analogy of the calculation performed in [9], the variance of the estimator grows with the norm of the entropy. We extend [11] method to deal with any stream distribution in order to guarantee that whatever the strategy of the adversary, the error on the estimation is kept small (as shown in Section 6). Specifically, the estimator X is computed on unfrequent items (*cf.* lines 14–18) as done in Relation (3), while the contribution of highly frequent items on the norm of the entropy is directly computed as $\sum_{(s_i,r_i)\in\hat{F}} r_i \log r_i$ (*cf.* lines 19). The set $\hat{F}$ represents the set of highly frequent items dynamically computed in Task T_2. Finally, to prevent some of the items to appear in both terms, we weight the contribution of both terms by p (*cf.* line 21).

4 Analysis

In this section, we analyze the properties of the AnKLe algorithm given in Figure 3. This analysis is split into three phases. We first evaluate the quality of $Y_{\hat{F}}$ through Lemma 5, we then evaluate the quality of Y_S through Lemmata 6 and 7, and finally derive the quality of AnKLe algorithm by combining the previous results with the one of [8] with Lemma 8.

In order to state the main theorem, we introduce the following notations. Let K be the set of the most frequent items i that satisfy $\hat{m}_i > e$ (if any) returned at line 12 in Figure 3. Let n_s and m_s be respectively defined as $n_s = n - |K|$ and $m_s = m - \sum_{k\in K} m_k$. Parameter n_s represents the number of "sparse" items (*i.e.*, the remaining items of the stream after having removed the most frequent ones as identified by Task T_2 and after the execution of line 14). In the same way, m_s represents the size of the sub-stream in the original stream occupied by these sparse items. Finally, the norm of the entropy of this sub-stream is denoted by F_H^s.

Theorem 4 *For any δ and ε such that $1/3 < \delta < 1$ and $0 < \varepsilon < \frac{1}{2}$, and for any constant $\Delta > 0$, the AnKLe algorithm gives an (ε, δ)-approximation of the KL-divergence, using*

$$\mathcal{O}\left(\log n + \frac{1}{\varepsilon^2} + \left(\frac{1}{\varepsilon} + \frac{\mu}{\varepsilon^2}\log\frac{1}{\delta}\right)(\log n + \log m)\right)$$

bits of space where $\mu = (\log m + \log e - 1)$ if $F_H^s \geq \frac{m^2}{\Delta m_s}$, and $\mu = (n - \frac{1}{\varepsilon} - 1)$ otherwise.

In particular, taking Δ to be a constant, we get a poly-logarithmic space algorithm that works on streams whose F_H^s is not "too small". Note that this is the case for most of the streams, as Task T_2 aims at removing the most frequent items, raising then the norm of the entropy of the sparse sub-stream.

Proof: The first part of the proof is directly derived from Lemma 8. Regarding the space complexity of AnKLe, it is given by the sum of the complexity of each Task T_1, T_2 and T_3 added up with the space required for the post-processing phase, which is $\mathcal{O}(1)$. From respectively [8], [9], [29], we get that the space complexity of AnKLe is $\mathcal{O}(\mathcal{C}_{KNW} + \mathcal{C}_{MG} + \mathcal{C}_{AMS})$, where:

$$\begin{cases} \mathcal{C}_{KNW} & = & \mathcal{O}\left(\log n + \frac{1}{\varepsilon^2}\right) & [8] \\ \mathcal{C}_{MG} & = & \mathcal{O}\left(c(\log n + \log m)\right) & [9] \\ \mathcal{C}_{AMS} & = & \mathcal{O}\left(s_1 s_2(\log n + \log m)\right) & [29]. \end{cases}$$

Lemmata 5 and 6, and the hypotheses on c, s_1 and s_2 presented in Equation 4 end the proof. □

We now show a series of results that prove Lemmata 5, 6 and 8.

4.1 Evaluation of $Y_{\hat{F}}$

The following lemma computes the quality of $Y_{\hat{F}}$.

Lemma 5 *For any $\varepsilon > 0$, we have $\mathbb{P}\{|Y_{\hat{F}} - Y_F| > \varepsilon Y_F\} = 0$, where $Y_{\hat{F}}$ is defined at line 17 in Figure 3.*

Proof: The proof combines the proof presented in [29] and the fact that $c > k$ counters are used. For space constraint reasons, the proof is presented in the companion paper [22]. □

4.2 Evaluation of Y_s

Let X^s be the same estimator as X (which has been defined in Section 3.2), but X^s is defined only on sparse items in the stream. From a derivation similar to the one used in Relation (1), X^s is an unbiased estimator of F_H^s.

Lemma 6 *$\forall \varepsilon, \delta > 0$, it exists s_1 and s_2 such that*

$$\mathbb{P}\{|Y_s - F_H^s| > \varepsilon F_H^s\} < \delta.$$

Proof: For space constraint reasons, the proof is presented in [22]. □

We now derive a relation between $E[X^s]$ and $V[X^s]$ to deduce an estimation on the size of s_1.

Lemma 7 *Let us consider the sub-stream m_s populated by sparse items. Then, for any constant $\Delta > 0$ if $F_H^s \geq \frac{m^2}{\Delta m_s}$ then*

$$V[X^s] \leq \Delta(\log m + \log e - 1)E[X^s]^2,$$

otherwise

$$V[X^s] \leq (n_s - 1)E[X^s]^2.$$

Proof: For space constraint reasons, the proof is presented in [22]. □

4.3 Evaluation of $\hat{D}$

As $|K| < 1/\varepsilon$, we are now able to explicitly give the value of all the parameters of tasks T_2 and T_3:

$$\begin{cases} c = \mathcal{O}\left(\frac{1}{\varepsilon}\right) \\ s_1 = \begin{cases} \mathcal{O}\left(\frac{\log m + \log e - 1}{\varepsilon^2}\right) & \text{if } F_H^s \geq \frac{m^2}{\Delta m_s} \\ \mathcal{O}\left(\frac{n_s - 1}{\varepsilon^2}\right) & \text{otherwise} \end{cases} \\ s_2 = \mathcal{O}\left(\log\frac{1}{\delta}\right) \end{cases} \tag{4}$$

Using these values, we have the necessary material to derive the global quality of AnKLe. First of all, by linearity of expectation, the random variable $\hat{D}$ is an unbiased estimator of $\mathcal{D}(q_\sigma || p^{(\mathcal{U})})$, given by Equation 1.

Lemma 8 *Given $\varepsilon > 0$ and $\delta > \frac{1}{3}$, we have*

$$\mathbb{P}\left\{|\hat{D} - \mathcal{D}(q_\sigma||p^{(\mathcal{U})})| > \varepsilon\mathcal{D}(q_\sigma||p^{(\mathcal{U})})\right\} < \delta.$$

Proof: Let $\delta' = \delta - \frac{1}{3}$. Then we have $\delta' > 0$.

Combining the independence of Y_s and $Y_{\hat{F}}$ and Lemmata 5 and 6, we have:

$$\begin{aligned} &\mathbb{P}\{|Y_s + Y_{\hat{F}} - E[Y_s + Y_{\hat{F}}]| > \varepsilon E[Y_s + Y_{\hat{F}}]\} \\ &\quad \leq \mathbb{P}\{|Y_s - E[Y_s]| > \varepsilon E[Y_s]\} \\ &\qquad + \mathbb{P}\{|Y_{\hat{F}} - E[Y_{\hat{F}}]| > \varepsilon E[Y_{\hat{F}}]\} \\ &\quad < \delta' \end{aligned}$$

By definition of $\mathcal{D}(q_\sigma||p^{(\mathcal{U})})$ in Relation 1 and $\hat{D}$ in AnKLe algorithm at line 21, we have :

$$\begin{aligned} &\mathbb{P}\{|\hat{D} - \mathcal{D}(q_\sigma||p^{(\mathcal{U})})| > \varepsilon\mathcal{D}(q_\sigma||p^{(\mathcal{U})})]\} \\ &\leq \mathbb{P}\{|\log\hat{F}_0 - \log F_0| > \varepsilon\log F_0\} \\ &\qquad + \mathbb{P}\left\{\frac{1}{m}|\hat{F_H} - F_H| > \frac{\varepsilon}{m}E[F_H]\right\} \\ &\leq \frac{1}{3} + \mathbb{P}\left\{\frac{1}{m}|\hat{F_H} - F_H| > \frac{\varepsilon}{m}E[F_H]\right\} \quad \text{(from [8])} \\ &\leq \frac{1}{3} + \mathbb{P}\left\{|Y_s + Y_{\hat{F}} - E[Y_s + Y_{\hat{F}}]| > \varepsilon E[Y_s + Y_{\hat{F}}]\right\} \\ &< \frac{1}{3} + \delta' = \delta \end{aligned}$$

that concludes the proof. □

5 Distributed Approximation Algorithm

In this section, we propose an algorithm that computes the KL-divergence among a set of ℓ distributed data streams with respect to the expected one, so that the number of bits communicated between the ℓ sites and the coordinator is minimized. This amounts for the coordinator to compute an approximation of the global KL-divergence composed by the joint distribution of ℓ data streams. Specifically, let $\mathbb{X} = \{X_1, X_2, \ldots, X_\ell\}$ be the random vector representing the set of discrete independent random variables $X_1, \ldots, X_\ell$ describing respectively the streams $\sigma_1, \ldots, \sigma_\ell$. As the KL-divergence is additive for independent distributions, if each node S_i computes locally an estimation $\widehat{D_{\sigma_i}}$ using AnKLe, we have $\widehat{D} = \sum_{1\leq i\leq \ell} \widehat{D_{\sigma_i}}$.

A simple motivating application that would benefit from our distributed algorithm is the node sampling application. Suppose that the ℓ sites of a large scale and dynamic system are in charge of locally running a uniform node sampling service[2], and locally feed AnKLe with the continuous stream σ'_i of identifiers generated by their node sampling service. By doing so each site S_i continuously estimates the KL-divergence $\widehat{D_{\sigma'_i}}$ of σ'_i to detect any deviation from a uniform stream. Indeed, in absence of malicious behaviors $\widehat{D_{\sigma'_i}}$ should be close to 0 as any node of the system has the same probability to be output by the uniform node sampling service. Now, in presence of attacks, the adversary may judiciously bias the ℓ input streams $\sigma_1, \ldots, \sigma_\ell$ so that for each site S_i, $\widehat{D_{\sigma'_i}}$ remains small enough not to be considered locally as an attack. On the other hand, by periodically aggregating $\widehat{D_{\sigma'_1}}, \ldots, \widehat{D_{\sigma'_\ell}}$, the coordinator can quickly detect whether the cumulated $\widehat{D_{\sigma'_i}}$ exceeds some given threshold, and if this is the case informs the ℓ sites to temporarily stop using their sampling service until $\widehat{D}$ shrinks to an acceptable value.

Our algorithm proceeds in rounds until all the data streams have been read in their entirety. In the following, we denote the substream of σ_i received by S_i during the round r by $\sigma_i^{(r)}$, and the cardinal of this substream by d_r. Specifically, in a bootstrap phase corresponding to round $r = 1$ of the algorithm, each site S_i computes a first estimation of the KL-divergence $\widehat{D_{\sigma_i}}$ of the received data stream σ_i as described in Algorithm 3. Once node S_i has received d_1 data items (where d_1 should typically be set to 100 [15]), then node S_i sends its first estimation $\widehat{D_{\sigma_i^{(1)}}}$ to the coordinator, and starts a new round $r = 2$. Upon receipt of $\widehat{D_{\sigma_i^{(1)}}}$ from any S_i, the coordinator asks all the $\ell - 1$ other nodes S_j to send their own sketch $\widehat{D_{\sigma_j^{(1)}}}$. Once the coordinator has received all $\widehat{D_{\sigma_i^{(1)}}}$, for $1 \leq i \leq \ell$, it sets $\forall i \in [\ell], \widehat{D_{\sigma_i}} \leftarrow \widehat{D_{\sigma_i^{(1)}}}$ and computes $\widehat{D} = \sum_{1\leq i\leq \ell} \widehat{D_{\sigma_i}}$.

At round $r > 1$, each node S_i updates its estimation $\widehat{D_{\sigma_i^{(r)}}}$ with the sequence of data streams received since the beginning of round r. Let $d_r = 2d_{r-1}$ be an upper bound on the number of received items during round r. When node S_i has received d_r data items since the beginning of round r, it sends its current estimation $\widehat{D_{\sigma_i^{(r)}}}$ to the coordinator and starts a new round $r + 1$. Note that during round r, S_i regularly compares $\widehat{D_{\sigma_i^{(r)}}}$ with $\widehat{D_{\sigma_i^{(r-1)}}}$ to detect whether significant variations in the stream have occurred before having received d_r items. This allows to inform the coordinator as quickly as possible that some attack might be undergoing. Upon receipt of the first $\widehat{D_{\sigma_i^{(r)}}}$ from any S_i, the coordinator asks all the $\ell - 1$ other nodes S_j to send it back their own $\widehat{D_{\sigma_j^{(r)}}}$. The coordinator locally updates the ℓ estimations and the global estimated divergence value. The communication flows between the ℓ sites and the coordinator is illustrated in the companion paper [22].

Theorem 9 *The distributed AnKLe algorithm gives an (ε, δ)-approximation of the KL-divergence between ℓ distributed input streams.*

Proof: The statement of this theorem directly derives from Theorem 4 and the fact that the KL-divergence is additive for independent distributions. □

Lemma 10 *The distributed AnKLe algorithm gives an approximation of the global KL-divergence, using*

$$\mathcal{O}\left(\log n + \frac{1}{\varepsilon^2} + \left(\frac{1}{\varepsilon} + \frac{\mu}{\varepsilon^2}\log\frac{1}{\delta}\right)(\log n + \log m)\right)$$

bits of space for each ℓ nodes, and $\mathcal{O}\left(\log n(\ell+1)\right)$ bits of space for the coordinator, where $\mu = (\log m + \log e - 1)$ if $F_H^s \geq \frac{m^2}{\Delta m_s}$, and $\mu = (n - \frac{1}{\varepsilon} - 1)$ otherwise.

Proof: The statement of this theorem directly derives from Theorem 4 and the fact that the coordinator maintains ℓ memory slots to store each estimation of all the ℓ nodes plus the global estimation itself. Each of these slots requires $\mathcal{O}(\log n)$ bits of space as the maximum value of the relative entropy with respect to the uniform distribution is $\log n + (n-1)\log m$ [30]. □

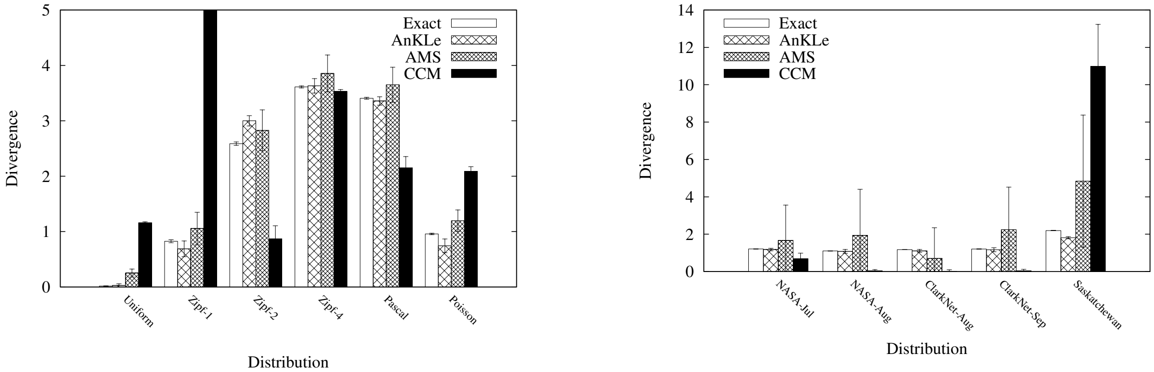
Lemma 11 *The distributed AnKLe algorithm gives an approximation of the global KL-divergence $\widehat{D}$ by sending $\mathcal{O}\left(r\ell\left(\log n + 1\right)\right)$ bits, where r is the number of rounds and n is the number of distinct items in all streams.*

Proof: Suppose that the number of rounds of the algorithm is equal to r. An upper bound of the number of bits sent by any node during round r is given by the size of its local estimation of the KL-divergence, which is $\log n$ (*cf.* Lemma 10). Finally, at the end of each round, the coordinator sends 1 bit to at most $\ell - 1$ nodes. □

6 Performance Analysis

In this section, we evaluate the accuracy of AnKLe by comparing its estimation with the exact value of the KL-divergence computed between the observed input

[2] A uniform node sampling service run at site S_i continuously outputs the identifier of a random node that belongs to S_i input stream σ_i, which is made of the nodes ids exchanged within the system.



(a) Comparisons between the Uniform distribution and the Uniform, Zipf-1, Zipf-2, Zipf-4, Pascal and Poisson distributions.

(b) Comparisons between a "uniform" data set and the real data ones.

Figure 4. Comparison between the exact value of the KL-divergence and its estimations computed with AnKLe, AMS and CCM.

stream and the uniform one. We also compare AnKLe with the estimator-based algorithms of Alon *et al.* [9] and Chakrabarti *et al.* [11]. In the former case, the original estimator computes the k-th frequency moment of a stream, while in the latter case, the original estimator measures the entropy of a stream. We have adapted both algorithms to compute, instead, the norm of the entropy. We have fed these algorithms 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 metric. Real data have been downloaded from the repository of Internet network traffic [31]. Two of them represent two weeks logs of HTTP requests to the Internet service provider ClarkNet WWW server – ClarkNet is a full Internet access provider for the Metro Baltimore-Washington DC area – the other two ones contain two months of HTTP requests to the NASA Kennedy Space Center WWW server, and the last one represents seven months of HTTP requests to the WWW server of the University of Saskatchewan, Canada. In the following these data sets will be respectively referred to as ClarkNet, NASA, and Saskatchewan traces. Table 2 presents some statistics of these data traces, in terms of stream size (*cf.* "# items"), number of distinct items in each stream (*cf.* "# distinct items") and the number of occurrences of the most frequent item (*cf.* "max. freq."). Note that all these benchmarks share a Zipfian behavior, with a lower α parameter for Saskatchwan.

Consider now the distributions of synthetic traces. All the generated streams have a length of $m = 200,000$ items. We have tested 750 different settings of the following parameters: n, the number of distinct items in the stream, s_1 and s_2, which are related to the size of the estimator matrix in Task T_3, and k, the number of counters used in Task T_2. For each setting, we have conducted 10 trials of the same experiment and computed the average and the standard deviation.

Data trace	# items (m)	# distinct items (n)	max. freq.
NASA (July)	1,891,715	81,983	17,572
NASA (August)	1,569,898	75,058	6,530
ClarkNet (August)	1,654,929	90,516	6,075
ClarkNet (September)	1,673,794	94,787	7,239
Saskatchewan	2,408,625	162,523	52,695

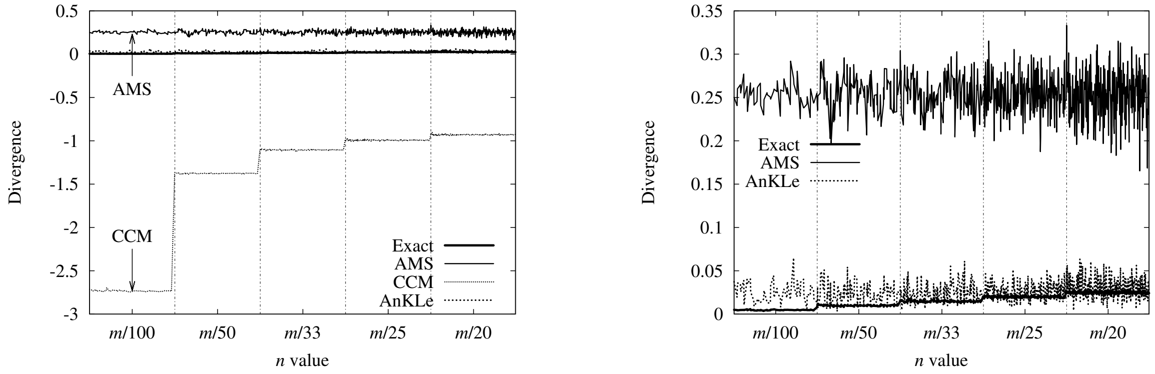
Table 2
Statistics of real data traces.

Except for the uniform and the zipfian-1 distributions that respectively model an ideal stream in which each item appears with the same frequency and a realistic one in absence of any attacks, the other four distributions capture different adversarial strategies. Specifically,

- the "Poisson" distribution allows to generate streams in which the frequency of a large quantity of items is significantly higher than the frequency of the remaining items. This type of stream might reflect an attack during which the adversary aims at over-representing a large number of its own node identifiers.
- the "Zipf-2" distribution engenders streams in which there is a small number of highly frequent items. This type of stream might correspond to an eclipse attack in which the objective of the adversary is to poison the routing tables of honest nodes.
- the "Pascal" and "Zipf-4" distributions produce streams in which a very small number of items have a very high frequency[3]. These distributions might illustrate streams in which very few items (typically 1 to 3) are over-pushed by the adversary.

Figure 4 summarizes the results obtained for the AnKLe, AMS and CCM estimators, averaged over 55,000 experiments (*i.e.* 750 different settings with 10 repetitions for each setting, over 6 synthetic distributions and 1000 settings for each data set). For clarity reasons, the average value of CCM for Zipf-1 has been cropped in Figure 4(a) since the estimated value of the KL-divergence by CCM is around 8.3. These results clearly

[3] Pascal distribution is also known as Negative Binomial distribution.



(a) Uniform distribution

(b) Uniform distribution (closer view)

Figure 5. KL-divergence estimation as a function of the size of the system n.

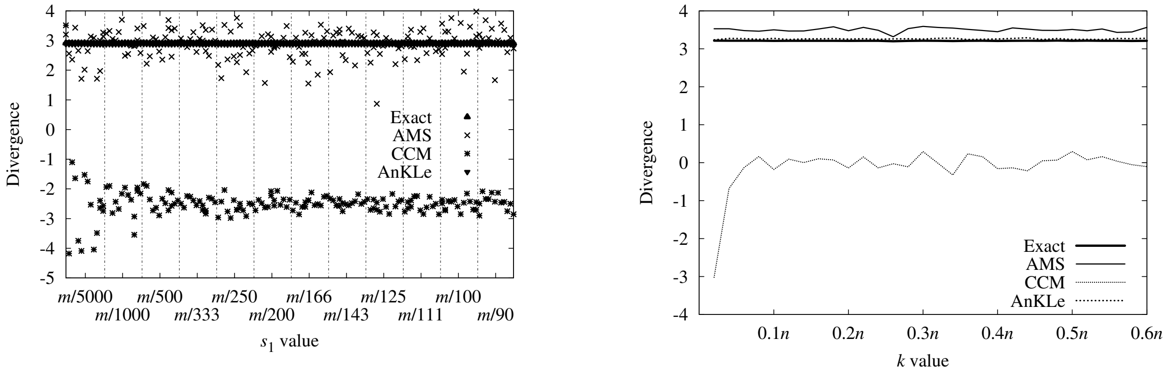
show that AnKLe outperforms the estimator CCM for all the distributions, even in scenario in which CCM should excel (*i.e.*, Zipf-4), as this corresponds to a stream in which a very frequent item exists in the observed stream. Compared with the AMS estimator, the results obtained with AnKLe are for most of them better except for the Zipf-2 distribution. But even for this specific distribution, the standard deviation of AnKLe is four times smaller than the one of AMS (*i.e.*, 0.09 versus 0.36), thus demonstrating that AnKLe provides a more robust and stable estimation than AMS on this distribution. Figure 4(b) confirms the relevance and efficiency of AnKLe when fed with real traces. Whatever the input traces, it clearly outperforms CCM and AMS estimators in terms of both average and standard deviation. Further results are presented in [22].

Figure 5 shows the evolution of the KL-divergence estimation as a function of n. The x-coordinate represents the number of distinct items in the stream as a ratio of its length m. The main observation that can be drawn from Figure 5(a) is that the CCM estimator behaves relatively badly in presence of a small number of distinct items with frequency uniformly distributed in the stream. However, its accuracy increases when the number of distinct items increases. The other two estimators are very close to the real value of the KL-divergence, with a clear advantage for AnKLe. This observation is confirmed in Figure 5(b) that corresponds to a zoom of Figure 5(a). This figure demonstrates that the estimation provided by AnKLe is very good. In average, the AnKLe estimation overlaps with the real value of the KL-divergence, contrary to AMS, and its standard deviation remains small, for any values of n. The evolution of the KL-divergence estimation as a function of k, and s_1 are presented in [22], in particular, we have observed that CCM is not adapted to uniform and near uniform streams, while AMS and AnKLe provide very good estimates for these distributions.

Figure 6 shows the KL-divergence estimation as a function of s_1 and k parameters. Several observations can be drawn from both figures. First, regarding Figure 6(a), the robustness of CCM estimator greatly improves with increasing values of s_1, as the cone-shaped curves converge for $s_1 > m/500$. On the first hand, the value towards which the CCM converges underestimates the KL-divergence. Thus, s_1 has a greater impact on CCM robustness than on its accuracy. On the other hand, variations of s_1 have not impact on AMS robustness. This feature does not appear in AnKLe as the weight given to Task T_2 makes it preponderant with respect to Task T_3, limiting accordingly the lack of robustness of Task T_3. Finally, the main observation drawn from Figure 6(b) is that AnKLe fully overlaps with the exact value of the KL-divergence, which clearly demonstrates the robustness of this estimator in presence of any input streams. Regarding CCM, we can observe that when the number of counters k is less than $0.1n$, then the Misra-Gries algorithm under-estimates the k most frequent items, which degrades the estimation of CCM. This confirms the theoretical bound of $k \geq \lceil 7\varepsilon^{-1} \rceil$ shown in [11]. On the other hand, variations of parameter k has not impact on AMS as this estimator does not decompose its computation according to items frequency characteristics. More results are presented in [22] with respect to the Poisson distribution. To summarize, experiments have validated the impressive accuracy and robustness of AnKLe in presence of a very large spectrum of distributions. This illustrates the importance of the weighting factor applied to both terms of the estimator.

7 Conclusion and Future Works

We have proposed AnKLe, a novel algorithm for estimating the KL-divergence between the observed stream and the expected one. AnKLe is very efficient both in terms of space and time, and requires only a single pass over the data stream. We have characterized how the different parameters impact the precision of the estimation and the space complexity of AnKLe. We have shown that AnKLe is an (ε, δ)-approximation algorithm with a space complexity $\tilde{\mathcal{O}}\left(\frac{1}{\varepsilon^2} + \frac{1}{\varepsilon}\right)$ bits in "most" of the cases. The distributed version of AnKLe can be used to compare different input streams, which may reveal as



(a) $n = m/125$, $k = 0.1n$

(b) $n = m/100$, $s1 = s2 = m/80$

Figure 6. KL-divergence as a function of s_1 and s_2 (on the left) and k (on the right) for the Pascal distribution.

an interesting ingredient to detect isolated intrusions. Finally, simulations have also shown that AnKLe performs always better, in terms of accuracy and robustness, than other state-of-the-art estimator-based algorithms.

Acknowledgments

The authors would like to warmly thank Sébastien Gambs for his valuable discussions at the early steps of this work [20].

References

[1] V. Chandola, A. Banerjee, and V. Kumar, “Anomaly detection: A survey,” *ACM Computing Surveys*, vol. 41, no. 3, pp. 1–58, 2009.

[2] B. K. Subhabrata, E. Krishnamurthy, S. Sen, Y. Zhang, and Y. Chen, “Sketch-based change detection: Methods, evaluation, and applications,” in *Internet Measurement Conference*, 2003, pp. 234–247.

[3] V. Karamcheti, D. Geiger, Z. Kedem, and S. Muthuskrishnan, “Detecting malicious network traffic using inverse distribution of packet contents,” in *Proc. of the workshop on Mining Network Data (MineNet) co-located with ACM SICOMM*, 2005.

[4] A. Lakhina, M. Crovella, and C.Diot, “Mining anomalies using traffic feature distributions,” in *Proc. of the ACM SIGCOMM*, 2005.

[5] E. Anceaume, Y. Busnel, and S. Gambs, “Uniform and Ergodic Sampling in Unstructured Peer-to-Peer Systems with Malicious Nodes,” in *Proc. of the 14th international conference on Principles of distributed systems (OPODIS)*, vol. 6490, 2010, pp. 64–78.

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

[7] 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.

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

[9] N. Alon, Y. Matias, and M. Szegedy, “The space complexity of approximating the frequency moments,” in *Proc. of the 28th annual ACM symposium on Theory of computing (STOC)*, 1996, pp. 20–29.

[10] 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.

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

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

[13] S. Guha, P. Indyk, and A. Mcgregor, “Sketching information divergences,” *Machine Learning*, vol. 72, no. 1-2, pp. 5–19, 2008.

[14] G. Cormode, S. Muthukrishnan, and K. Yi, “Algorithms for distributed functional monitoring,” in *Proc. of the 19th annual ACM-SIAM Symposium On Discrete Algorithms (SODA)*, 2008.

[15] C. Arackaparambil, J. Brody, and A. Chakrabarti, “Functional monitoring without monotonicity,” in *Proc. of the 36th ACM International Colloquium on Automata, Languages and Programming: Part 1*, 2009.

[16] P. B. Gibbons and S. Tirthapura, “Estimating simple functions on the union of data streams,” in *Proc. of the Thirteenth Annual ACM Symposium on Parallel Algorithms and Architectures (SPAA)*, 2001, pp. 281–291.

[17] Z. Haung, K. Yi, and Q. Zhang, “Randomized algorithms for tracking distributed count, frequencies and ranks,” in *Proc. of 31st ACM Symposium on Principles of Database Systems (PODS)*, 2012.

[18] Z. Liu, B. Radunovic, and M. Vojnovic, “Continuous distributed counting for non-monotonic streams,” in *Proc. of 31st ACM Symposium on Principles of Database Systems (PODS)*, 2012.

[19] E. Anceaume and Y. Busnel, “An information divergence estimation over data streams,” in *Proc. of the IEEE International Symposium on Network Computing and Applications (NCA)*, 2012.

[20] E. Anceaume, Y. Busnel, and S. Gambs, “Ankle: Detecting attacks in large scale systems via information divergence,” in *Proc. of the European Dependable Computing Conference (EDCC)*, 2012.

[21] T. Cover and J. Thomas, “Elements of information theory,” *Wiley New York*, 1991.

[22] E. Anceaume and Y. Busnel, “A distributed information divergence estimation over data streams,” supplementary materials.

[23] E. D. Demaine, R. López-Ortiz, and J. I. Munro, “Frequency estimation of internet packet streams with limited space,” in *In Proceedings of the 10th Annual European Symposium on Algorithms*. Springer-Verlag, 2002, pp. 348–360.

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

[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] S. M. Ali and S. D. Silvey, “General Class of Coefficients of Divergence of One Distribution from Another,” *Journal of the Royal Statistical Society. Series B (Methodological)*, vol. 28, no. 1, pp. 131–142, 1966.

[27] M. Durand and P. Flajolet, “Log-log counting of large cardinalities,” in *Proc. of the 11th European Symposium on Algorithms (ESA)*, 2003.

[28] P. Gibbons, *Data Streams Management: Processing High-Speed Data Streams*. Elsevier, 2007.

[29] J. Misra and D. Gries, “Finding repeated elements,” *Science of Computer Programming*, vol. 2, no. 2, pp. 143–152, 1982.

[30] E. Anceaume, Y. Busnel, and S. Gambs, “Characterizing the adversarial power in uniform and ergodic node sampling,” in *Proc. of the International Workshop on Algorithms and Models for Distributed Event Processing (AlMoDEP)*. ACM, 2011.

[31] the Internet Traffic Archive, “http://ita.ee.lbl.gov/html/traces.html,” Lawrence Berkeley National Laboratory, Apr. 2008.

Emmanuelle Anceaume received her Phd degree in Computer Science from the University Paris-Orsay (Paris-XI – France) for her work on dependability systems. She then spent one year at Cornell University and worked on unreliable failure detectors. She is a CNRS researcher at IRISA-UMR6074 lab. Her main research interests are in the field of distributed systems and algorithms with a focus on fault tolerance issues. In the context of dynamic and scalable systems, she is currently investigating dependability and security issues.

Yann Busnel is currently an Associate Professor in University of Nantes (France), involved in the Large-Scale Data Management team. He obtained his PhD degree in Computer Science in 2008 from the University of Rennes (France). Then, in 2009, he spent one year as a Researcher at MIDLAB, involved in the distributed systems research group of the University of Rome “La Sapienza” (Italy). He works essentially on very large scale dynamic distributed algorithm and data management in large-scale systems. He also a referee of several international journal and conferences.