Identifying Global Icebergs in Distributed Streams

Emmanuelle Anceaume, Yann Busnel, Nicolò Rivetti, Bruno Sericola

Identifying Global Icebergs in Distributed Streams

Emmanuelle Anceaume
CNRS / IRISA, France
emmanuelle.anceaume@irisa.fr

Yann Busnel
Crest (Ensai), France
yann.busnel@ensai.fr

Nicolò Rivetti
LINA / Université de Nantes, France
nicolo.rivetti@univ-nantes.fr

Bruno Sericola
Inria, France
bruno.sericola@inria.fr

***Abstract*—We consider the problem of identifying global iceberg attacks in massive and physically distributed streams. A global iceberg is a distributed denial of service attack, where some elements globally recur many times across the distributed streams, but locally, they do not appear as a deny of service. A natural solution to defend against global iceberg attacks is to rely on multiple routers that locally scan their network traffic, and regularly provide monitoring information to a server in charge of collecting and aggregating all the monitored information. Any relevant solution to this problem must minimise the communication between the routers and the coordinator, and the space required by each node to analyse its stream. We propose a distributed algorithm that tracks global icebergs on the fly with guaranteed error bounds, limited memory and processing requirements. We present a thorough analysis of our algorithm performance. In particular we derive a tight upper bound on the number of bits communicated between the multiple routers and the coordinator in presence of an oblivious adversary. Finally, we present the main results of the experiments we have run on a cluster of single-board computers. Those experiments confirm the efficiency and accuracy of our algorithm to track global icebergs hidden in very large input data streams exhibiting different shapes.**

***Index Terms*—data stream model; randomised approximation algorithm; generalised coupon collector problem; oblivious adversary; performance analysis.**

I. Introduction

A Denial of Service (DoS) attack tries to take down an Internet resource by flooding this resource with more requests than it is capable of handling. A Distributed Denial of Service (DDoS) attack is a DoS attack triggered by many machines that have been infected by a malicious software, causing immediately the total shutdown of targeted web resources (*e.g.*, e-commerce websites). A common approach to detect and to mitigate DDoS attacks is to monitor network traffic through routers and to look for highly frequent signatures that might suggest ongoing attacks. However, a recent strategy followed by the attackers is to hide their massive flows of requests by subtly distributing them in a multitude of routes, so that locally, the malicious sub-flows do not appear as frequent, while globally they represent a significant percentage Θ of the network traffic [14]. The term "global iceberg" has been introduced to describe such an attack as only a very small part of the latter can be observed from each single router [18]. A natural solution to track and detect global icebergs is to rely on multiple routers that locally scan their network traffic, and regularly provide monitoring information to a server in charge of collecting and aggregating all the monitored information. To be applicable, two issues must be solved. Firstly, routers must be capable of monitoring a very large number of flows to discover the presence of potential global icebergs and this must be done on the fly to have some chance to detect icebergs soon enough. Secondly, the frequency of the communications between routers and the server must be low enough to prevent the server from being overloaded by iterative exchanges with all the routers; however reducing the frequency of these exchanges must not jeopardise the detection latency of global icebergs and must not introduce false negatives (that is, the non-detection of global icebergs).

To address both points, one needs to design appropriate schemes for summarising the input streams [16], namely algorithms capable of efficiently and accurately maintaining some function f over the prefix of a huge – potentially unbounded – sequence of data items. By efficient, we mean that such algorithms must compute function f by scanning only once each data item of the input stream and by storing only a very small amount of information with respect to the number of data items received so far. By accurate, we expect that, despite their remarkable space-efficiency, such algorithms approximate function f with a guaranteed approximation error. Finally, to cope with adversarial strategies, such algorithms must be insensitive to the order in which data items are received.

The solution we propose to efficiently and accurately detect and identify global icebergs while preserving the server from being overloaded by all the monitored information, consists in relying on sketches and in locally organising items according to their estimated frequency. Only the most recent and high enough flows, *i.e.*, those that locally represent at least a fraction Θ of the local stream, are tracked and sent to the server (called coordinator in the following), while the others are thrown away. This is achieved by locally splitting the interval $[\Theta, 1]$ into ℓ sub-intervals in a geometric way, and by maintaining a small number ℓ of buffers, $\Gamma_1, \ldots, \Gamma_\ell$, such that each Γ_k will contain data items whose probability of occurrence matches the k-th interval. Each time one of these buffers is full, its content is sent to the coordinator. The coordinator polls all the other nodes to determine if global icebergs are indeed present. We show that our solution (ε, δ)-approximates the detection of global icebergs, for any $\varepsilon \in [0, 1]$ and $\delta \leq 1/2$ in a space efficient way. Moreover, we provide a tight upper bound on the number of bits exchanged by our distributed algorithm. Finally, we have implemented our solution on a testbed of single-board computers fed with different shapes of streams (both synthetic and real traces). Numerical results clearly confirm that for any value of Θ, all the global icebergs are detected (*i.e.*, no false negatives). They also show that the communication overhead due to our protocol (*i.e.*, the number of bits communicated between the nodes and the coordinator to detect global icebergs) is very small (less than 1.2% of the global

number of bits received by the routers). Finally, these experiments show that less than 3% of the global stream need to be read to detect all the global icebergs.

The outline of this paper is the following. Section II presents related works. Section III formally states the problem and describes the model we rely on to derive bounds of our algorithms. Section IV presents and analyses an omniscient and full-space algorithm that solves the global iceberg problem, while Section V presents a knowledge-free and space-efficient algorithm and analyses its quality. Finally, numerical results obtained from extended experiments conducted on single-board computers are studied in Section VI. Section VII concludes.

II. Related Work

The detection of heavy hitters or global icebergs originating from multiple streams has first been studied by Manjhi *et al.* [14]. In their paper, the authors propose a solution for detecting recent global icebergs by relying on a multi-level structure where each node must guarantee a degree of precision that depends on its level in the hierarchical structure. This structure helps to minimise the communication between nodes and the central coordinator. On the other hand, and in contrast to our work, they make the assumption that global icebergs are locally detectable at each node. In [18], the authors do not assume any more that nodes can locally detect global iceberg, however they suppose that there exists a gap between non iceberg frequencies and iceberg ones. That is, items that appear at least T times from the inception of the stream are global icebergs, while there are no items whose frequency belongs to an interval $(\lambda T, T)$, with $0 < \lambda < 1$. Based on this assumption, the authors can accurately detect the presence of global icebergs. Note that, in contrast to our work, they do not identify the global icebergs items, they only inform the coordinator that some global icebergs exist. In addition, in our work, we also (ε, δ)-approximate the size of each global iceberg. In [19], the authors propose a solution that identifies items whose aggregated frequency over the distributed streams exceeds some given threshold, irrespective of the size of the input streams. Thus, their motivation is to detect global icebergs only during a fixed time interval (note that the same definition has been adopted by [18]). Their approach combines sampling and sketching through Bloom filters [4], [6], [10], [13], [15] to respectively sample and count items. Similarly to our solution, their strategy is robust to any adversarial items splitting, however we address a much more challenging model. The authors in [19] suppose that each node locally knows prior to executing its algorithm the exact frequency of all the items it will receive. This model is commonly called the "distributed bag model". Finally, Yi and Zhang in [17] propose a solution in the spirit of the ones proposed in functional monitoring to minimise the communication cost between the nodes and the coordinator. However this is achieved by requiring that each node maintains at any time the exact frequency of each received item in its stream, which is definitively a very strong assumption.

III. Model of the system and addressed problem

A. The distributed functional monitoring model

We present the computational model used to analyse our algorithms. This model follows the *distributed functional monitoring model* proposed by Cormode *et al.* [7], which combines features of the data streaming model and communication complexity. Specifically, we consider a set $\mathcal{S}$ of nodes such that each node receives a continuous stream (or sequence) of items drawn from a large universe $\mathcal{N}$. Items arrive quickly, in any order and may recur an arbitrary and unknown number of times in each stream. This allows us to model the reception of high speed TCP/IP streams with different throughputs at each router. As TCP DDoS attacks are tracked, then the relevant items are the packet destination IP addresses. In the following we denote the different items by integers $1, \ldots, n$, with $n = |\mathcal{N}|$. We denote by σ_s the stream received by node s, $s \in \mathcal{S}$, and by m_s the number of items received from σ_s so far. For any two nodes s and s' in $\mathcal{S}$, m_s can be different from $m_{s'}$. Any input stream σ_s implicitly defines an empirical probability distribution on the set of items it contains; the probability $p_{j,s}$ of occurrence of item j in σ_s is approximated by $f_{j,s}/m_s$, where $f_{j,s}$ represents the number of times item j has occurred in σ_s since the inception of σ_s. It is important to note that nodes do not have any a priori knowledge on the probability distribution of items they receive. Indeed, due to memory constraints, 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 to keep pace with the data stream. Thus streams need to be processed sequentially and *online*, that is, any item of the stream that has not been locally stored for any further processing cannot be read again.

Nodes communicate solely with a dedicated node, called *coordinator*. Communications are initiated by nodes upon receipt of an item or a sequence of them in their input stream. This in turn triggers additional communications from the coordinator to some of the S nodes, where $S = |\mathcal{S}|$. The goal of the coordinator is to continuously determine whether a given function applied on the union of received streams approximately exceeds some given threshold $\Theta \in (0, 1]$. This is formalised in the next section. The key property of the model is to maintain this knowledge by minimising both the space used by the S nodes and the number of bits exchanged between these nodes and the coordinator.

B. The global iceberg detection problem

The global iceberg detection problem has first been formalised by Estan and Varghese [11], and then adapted to different contexts [14], [18], [19]. In this paper, we extend this problem to the general distributed functional monitoring model [7]. Informally, the global iceberg detection problem lies, for the coordinator, in quickly identifying any item j whose aggregated number of occurrences over the union of the distributed streams approximately exceeds some given fraction of the total size of all the streams since their inception. Such an item j is called a *global iceberg*. Specifically, we denote by σ the union of the S data streams σ_s since the inception of σ_s. We have $\sigma = \sigma_1 \cup \ldots \cup \sigma_S$. Let m be the number of items in σ. We have $m = \sum_{s \in \mathcal{S}} m_s$. Finally, let f_j be the total number of occurrences of item j in σ.

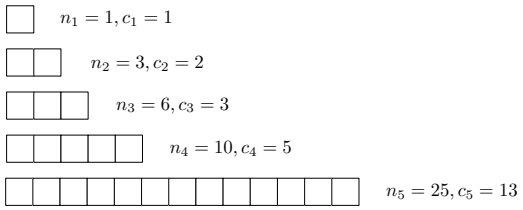

Fig. 1. Data structure on node $s \in \mathcal{S}$, with $\Theta = 0.04$, $r = 0.5$ and $S = 20$.

Definition 1 (The global iceberg detection problem)**.** *For any given threshold $\Theta \in (0, 1]$, approximation parameter $\varepsilon \in [0, 1]$, and probability of failure $\delta \leq 1/2$, the global iceberg detection problem consists for the coordinator C*

- *in outputting item j if $f_j \geq \Theta m$, and*
- *in never outputting item j if $f_j < (1 - \varepsilon)\Theta m$.*

This must be achieved by minimising both the space used by the S nodes and the number of bits exchanged between these nodes and the coordinator.

C. Adversary

We assume the presence of an *adaptive* adversary that tries to prevent the detection of global icebergs by judiciously distributing them in the S data streams so that none of the S monitoring nodes can locally detect an ongoing attack. By adaptive we mean that the adversary can adaptively insert any number of items to increase or decrease the number of global icebergs according to the current states of both the coordinator C, the S nodes and threshold Θ. Notice that the adversary may also strategise to make the coordinator a potential DDoS target by maximising the number of interactions between the S nodes and the coordinator. This is achieved by adversarially ordering items in the streams, and/or by injecting well chosen items. On the other hand, both the S nodes and the coordinator follow the prescribed protocols, that is, they are correct. We finally suppose that any algorithm run by any correct node is public knowledge to avoid some kind of security by obscurity.

IV. An omniscient algorithm to track global icebergs

Prior to presenting our space-efficient and knowledge-free solution to the global iceberg problem, we first describe our algorithm by ideally assuming that each time a data item j is received in the input stream σ_s, item j is tagged with the exact probability $p_{j,s}$ with which it will appear in σ_s. Note however that the algorithm does not know ahead of time the items that will appear in σ_s. Furthermore we suppose that each node s locally keeps track of items frequency count, *i.e.*, the number of times each item j has been received so far at node s.

A. Principles of the solution

The pseudo-code of the algorithms run by each of the S nodes and by the coordinator are respectively provided in Algorithms 1 and 2. These algorithms work as follows. Each node $s \in \mathcal{S}$ reads on the fly and sequentially its input stream σ_s. For each received item j, node s increments its frequency count $F[j]$, and maintains the current size m_s of its input stream, which is the number of items received so far in σ_s. If $p_{j,s} \geq \Theta$ then j is a potential candidate for being a global iceberg and thus s keeps track of item j if not already done (see lines 9–28 of Algorithm 1). This is achieved by maintaining ℓ buffers $\Gamma_1, \ldots, \Gamma_\ell$, where $\ell = \lfloor \log_2 S \rfloor + 1$.[1] The interval $[\Theta, 1]$ is split into ℓ sub-intervals in a geometric way. Each buffer Γ_k, whose size is denoted by c_k, only contains items j whose probability $p_{j,s}$ matches the k-th interval. That is, buffer Γ_k, with $1 \leq k \leq \ell - 1$, contains items j whose probability $p_{j,s}$ verifies $\Theta + (1-\Theta)/2^k < p_{j,s} \leq \Theta + (1-\Theta)/2^{k-1}$, and Γ_ℓ contains items j whose probability $p_{j,s}$ satisfies $\Theta \leq p_{j,s} \leq \Theta + (1 - \Theta)/2^{\ell-1}$. The size c_k of each of buffer Γ_k, with $1 \leq k \leq \ell$, is set to $\lceil n_k \times r \rceil$, where n_k is the maximum number of items that could fit the k-th buffer, and $0 < r \leq 1$. Since Γ_k, for $1 \leq k \leq \ell - 1$, contains items whose probability $p_{j,s}$ verifies $\Theta + (1 - \Theta)/2^k < p_{j,s} \leq \Theta + (1 - \Theta)/2^{k-1}$, we set n_k to $\lfloor 1/(\Theta + (1 - \Theta)/2^k) \rfloor$ and $n_\ell = \lfloor 1/\Theta \rfloor$. Note that for $r = 1$, that is for all k, $c_k = n_k$, if some buffer say Γ_h is filled with c_h items then it means that all the items j in the stream are uniformly distributed with $p_{j,s} = \Theta + (1-\Theta)/2^h$ (and $p_{j,s} = \Theta$ for $h = \ell$), and thus none of the other buffers can receive any single item. Thus in practice r is set to a value less than or equal to $1/2$. When some buffer Γ_k is full (*i.e*, Γ_k contains c_k items), node s sends all the items of Γ_k, together with their frequency counts and the stream size m_s to the coordinator (see lines 20–26 of Algorithm 1), and empties Γ_k. Upon receipt of such a buffer, the coordinator queries the other $S - 1$ nodes to get the frequency of each of the items sent by node s, if any (see lines 7–10 of Algorithm 2).

Finally, by combining all these pieces of information, the coordinator checks whether any of these items is a global iceberg or not (see lines 15–19 of Algorithm 2). Note that in contrast to [18], the coordinator is capable of identifying which are the global icebergs that is an important feature when tracking DDoS attacks. Finally, since the distribution of the items in σ_s is unknown, one cannot guarantee that c_k distinct items with a probability that matches Γ_k exist in the stream σ_s. Actually, even no item can appear in the stream with those probabilities. Thus to guarantee that all the potential icebergs in Γ_k are at some point sent to the coordinator, a timer τ_k is set upon receipt of the first item in Γ_k, and is incremented each time node s reads an item from the input stream σ_s. Timeout of τ_k is set to $H_{\lfloor 1/\Theta \rfloor}/\Theta$, where H_n is the n-th harmonic number defined by $H_0 = 0$ and $H_n = 1 + 1/2 + \cdots + 1/n$. Lemma 1 shows the derivation of that timeout. Each time the coordinator detects a global iceberg j, it informs all the S nodes that j is a global iceberg (this amounts to returning *Freq* in Line 21 of Algorithm 2 to both the application and the S nodes). Nodes locally enqueue j (in FIFO order) in a specific buffer, denoted by Λ_s, whose size is equal to $\lceil r/\Theta \rceil$. Hence, when a node s locally detects that some item j is a potential global iceberg, then s does not inform the coordinator if j belongs to Λ_s. Note that as the oldest detected global icebergs

[1]For the sake of clarity, we will use the notation log to denote the logarithm in base 2 for the rest of this paper, unless otherwise specified.

Algorithm 1: Omniscient and full-space algorithm run at any node $s \in \mathcal{S}$

Input: An arbitrary input stream σ_s; Θ: threshold qualifying an item as a global iceberg; r: proportion size parameter of the ℓ buffers;

foreach $k \in \{1, \ell - 1\}$ **do**
  $\Gamma_k \leftarrow \emptyset$; $n_k \leftarrow \lfloor 1/(\Theta + (1-\Theta)/2^k) \rfloor$; $c_k \leftarrow \lceil r n_k \rceil$;
  $\tau_k \leftarrow$ (not-active, 0);
end
$\Gamma_\ell \leftarrow \emptyset$; $n_\ell \leftarrow \lfloor 1/\Theta \rfloor$; $c_\ell \leftarrow \lceil r n_\ell \rceil$; $\tau_\ell \leftarrow$ (not-active, 0) ;
$m_s \leftarrow 0$; *Freq* $\leftarrow \emptyset$; $\Lambda_s \leftarrow \emptyset$;
for $j \in \sigma_s$ **do**
  $m_s \leftarrow m_s + 1$;
  $F[j] \leftarrow F[j] + 1$;
  if $p_{j,s} \geq \Theta$ **then**
    if $p_{j,s} \leq \Theta + (1-\Theta)/2^{\ell-1}$ **then**
      $k \leftarrow \ell$;
    else
      $k \leftarrow \min\{h \in \{1, \ell - 1\} \mid \Theta + (1-\Theta)/2^h < p_{j,s}\}$;
    end
    if $j \notin \Gamma_k$ **then**
      if $\Gamma_k = \emptyset$ **then**
        $\tau_k \leftarrow$ (active, 0);
      end
      $\Gamma_k \leftarrow \Gamma_k \cup \{j\}$;
      if $|\Gamma_k| = c_k$ **then**
        for each $\{j\} \in \Gamma_k$ **do**
          Freq $\leftarrow$ *Freq* $\cup\{(j, F[j])\}$;
        send message ("identify", m_s, *Freq*) to C;
        $\Gamma_k \leftarrow \emptyset$; *Freq* $\leftarrow \emptyset$; $\tau_k \leftarrow$ (not-active, 0);
      end
    end
  end
  foreach active timer τ_k, $k \in \{1, \ell\}$ **do**
    $\tau_k \leftarrow$ (active, $\tau_k + 1$);
    if $\tau_k > H_{\lfloor 1/\Theta \rfloor}/\Theta$ **and** $\Gamma_k \neq \emptyset$ **then**
      for each $\{j\} \in \Gamma_k$ **do**
        Freq $\leftarrow$ *Freq* $\cup\{(j, F[j])\}$;
      send message ("identify", m_s, *Freq*) to C;
      $\Gamma_k \leftarrow \emptyset$; *Freq* $\leftarrow \emptyset$; $\tau_k \leftarrow$ (not-active, 0);
    end
  end
end
upon receipt message ("freq ?", *Freq*) from the coordinator **do**
  for each $j \in Freq$ **do**
    $Freq \leftarrow (j, F[j])$;
    if $\exists k \in \{1, \ell\}, j \in \Gamma_k$ **then**
      $\Gamma_k \leftarrow \Gamma_k \setminus \{j\}$;
    end
  send message ("freq", m_s, *Freq*) to C;
upon receipt message ("Global_Iceberg", F) from the coordinator **do**
  enqueue F in Λ_s;

are progressively dequeued, Λ_s contains at any time the last $\lceil r/\Theta \rceil$ global icebergs. The rationale of this feedback is to prevent nodes from repeatedly informing the coordinator of the presence of a global iceberg already detected by the coordinator.

B. Analysis of the omniscient algorithm

In this section we provide an upper bound of the number of bits communicated between the S nodes and the coordinator (see Theorem 2), and show that the omniscient algorithm solves the global iceberg problem (see Theorem 3).

Algorithm 2: Algorithm run at the coordinator

Input: Θ: threshold qualifying an item as a global iceberg;
Output: The set F of global icebergs;

upon receipt message ("identify", m', *Freq*) from s **do**
  $m \leftarrow m'$;$F \leftarrow$ *Freq*;
  for each $(k, f_k) \in$ Freq **do**
    $(k, f_k) \leftarrow (k, \texttt{null})$;
  for each $s' \neq s \in \mathcal{S}$ **do**
    send message ("freq ?", *Freq*) to s';
  for each received message ("freq", m', *Freq*) from $s', s' \neq s$ **do**
    $m \leftarrow m + m'$;
    for each $(k, f_k) \in F$ **do**
      Update F: $f_k \leftarrow f_k + f'_k$ with $(k, f'_k) \in$ *Freq*;
  for each $(k, f_k) \in F$ **do**
    if $f_k < \Theta m$ **then**
      $F \leftarrow F \setminus \{(k, f_k)\}$;
    end
  if $F \neq \emptyset$ **then**
    returns F;
    send message ("Global_Iceberg", F) to each $s \in \mathcal{S}$;
  end

1) Adversarial Strategies: We investigate whether malicious nodes can prevent the global iceberg problem from being solved. This amounts to showing that for any given threshold $\Theta \in (0, 1]$, and approximation parameter $\varepsilon \in (0, 1]$, the adversary cannot compel the coordinator to return any items j such that $f_j < (1 - \varepsilon)\Theta m$ and cannot prevent the coordinator from returning all items j such that $f_j \geq \Theta m$ by finely distributing the occurrences of item j among the S streams. Theorem 3 shows that Algorithms 1 and 2 exactly guarantee such properties.

Maximising the communication between the S nodes and the coordinator requires for the adversary to manipulate the S input streams so that the time needed to locally fill the buffers is minimised, and thus the frequency at which nodes communicate with the coordinator is maximised. Theorem 2 shows that this is achieved if, for all $s \in \mathcal{S}$, all the global icebergs appear with the same probability in σ_s.

2) Communication complexity of the omniscient algorithm: Communication between the S nodes and coordinator C is triggered each time a buffer becomes full or upon timeout. To analyse the communication cost, we study a generalisation of the coupon collector problem. Indeed, we have to determine the number of items, in expectation, that must be received at any node $s \in \mathcal{S}$ to locally fill buffer Γ_k, $1 \leq k \leq \ell$.

a) Preliminary results: In the general formulation of this problem, we have a set of n coupons with p_i being the probability that coupon i is drawn and $p_1 + \cdots + p_n = 1$. The problem amounts to determine the distribution of the number T_n of coupons that need to be drawn from set $\{1, 2, \ldots, n\}$ with replacement, till obtaining the full collection of the n different coupons. In our case, we consider a generalisation of this formulation, where we need to determine the distribution of the number $T_{c,n}$ of coupons that must be drawn, with replacement, to collect

$c \leq n$ different coupons from the set $\{1,\ldots,n\}$. The expected value of $T_{c,n}$ has been considered in [12]. In addition we suppose that $p=(p_1,\ldots,p_n)$ is not necessarily a probability distribution, that is $p_1+\cdots+p_n \leq 1$ and we define $p_0 = 1-(p_1+\cdots+p_n)$. This models the presence of a null coupon 0 that is drawn with probability p_0 but that is not allowed to belong to the collection of c items, $1 \leq c \leq n$, among $\{1,\ldots,n\}$. In the following, we denote by $S_{i,n}$ all the sets of items of $\{1,\ldots,n\}$ whose size is exactly equal to i, that is $S_{i,n} = \{J \subseteq \{1,\ldots,n\} \mid |J| = i\}$. Note that we have $S_{0,n} = \{\emptyset\}$. For every $J \in S_{i,n}$, we define $P_J = \sum_{j\in J} p_j$, with $P_\emptyset = 0$. Finally, since the distribution of $T_{c,n}$ depends on the vector $p=(p_1,\ldots,p_n)$, we will use the notation $T_{c,n}(p)$ instead of $T_{c,n}$, meaning by the way that the dimension of vector p is n. We have from [2] that the expectation of $T_{c,n}(p)$ is given for every $n \geq 1$ and $c = 1,\ldots,n$ by

$$\mathbb{E}[T_{c,n}(p)] = \sum_{i=0}^{c-1} (-1)^{c-1-i} \binom{n-i-1}{n-c} \sum_{J\in S_{i,n}} \frac{1}{1-(p_0+P_J)}, \quad (1)$$

and that for every $p=(p_1,...,p_n)$ with $p_0 = 1-\sum_{i=1}^n p_i$ and $0 < \Theta \leq p_i$ for $i=1,\ldots,n$,

$$\mathbb{E}[T_{c,n}(v)] \leq \mathbb{E}[T_{c,n}(p)] \leq \mathbb{E}[T_{c,n}(q)],$$

where vector $v = ((1-p_0)/n,\ldots,(1-p_0)/n)$, and vector $q = (\Theta,\ldots,\Theta,1-p_0-(n-1)\Theta)$. Note that the second inequality is true only for distribution p such that $p_i \geq \Theta$. Vector v represents a scenario where all the coupons $\{1,\ldots,n\}$ are drawn uniformly, while vector q represents a scenario where they all occur with probability Θ except a single one that occurs with high probability $1-p_0-(n-1)\Theta$. Intuitively, vector v is the one that minimises the time to get a collection of $c \leq n$ coupons, while vector q is the one that maximises it when $p_i \geq \Theta > 0$, for $i=1,\ldots,n$.

Lemma 1. *For every $n \geq 1$, $c = 1,\ldots,n$, and $0 < \Theta \leq p_i$ for $i=1,\ldots,n$, we have*

$$\mathbb{E}[T_{c,n}(q)] \leq H_{\lfloor 1/\Theta \rfloor}/\Theta.$$

Proof: For space reasons, the proof is omitted from this paper, but appears in the companion paper [1]. ∎

b) Calculation of the communication cost to track global icebergs: We first derive the number of items that need to be received in each stream σ_s, $s \in \mathcal{S}$, to fill buffer Γ_k, $1 \leq k \leq \ell$. This is done by applying the general formulation of the coupon collector to each buffer Γ_k, $k = 1,\ldots,\ell$.

For each $k = 1,\ldots,\ell-1$, we denote by J_k the set of all the items j in σ_s whose probability of occurrence $p_{j,s}$ verifies $\Theta + (1-\Theta)/2^k < p_{j,s} \leq \Theta + (1-\Theta)/2^{k-1}$. We set $j_k = |J_k|$, and we have $j_k \leq n_k$ (recall from Section IV-A, that n_k represents the maximal number of items whose probability of occurrence is equal to the lower bound of the range, that is $n_k = \lfloor 1/(\Theta + (1-\Theta)/2^k) \rfloor$). For $k = \ell$, we denote by J_ℓ the set of all the items j in σ_s whose probability of occurrence $p_{j,s}$ verifies $\Theta < p_{j,s} \leq \Theta + (1-\Theta)/2^{\ell-1}$. We set $j_\ell = |J_\ell|$, and we have $j_\ell \leq n_\ell$ with $n_\ell = \lfloor 1/\Theta \rfloor$. Hence, $S_{i,j_k} = \{J \subseteq J_k \mid |J| = i\}$, and for every $J \in S_{i,j_k}$, $P_J = \sum_{j\in J} p_{j,s}$.

Based on this, for each node $s \in \mathcal{S}$, the expected number of items that need to be received from σ_s to fill buffer Γ_k is given by $\mathbb{E}[T_{c_k,j_k}(p)]$, with vector $p = (p_{j,s})_{j\in J_k}$. From Relation (1), we have $\mathbb{E}[T_{c_k,j_k}(p)] =$

$$\sum_{i=0}^{c_k-1} (-1)^{c_k-1-i} \binom{j_k-i-1}{j_k-c_k} \sum_{J\in S_{i,j_k}} \frac{1}{1-(p_0+P_J)},$$

where $p_0 = 1-P_{J_k}$.

We now determine the timer settings. The instant at which each timer τ_k, $1 \leq k \leq \ell$, should fire must be short enough to bound the global iceberg detection latency, but sufficiently large to prevent the content of a buffer from being sent too often, that is well before it contains potential global icebergs. Our idea is to set the timer for a value that allows a buffer to be full, in expectation, whatever the frequency at which items recur the input stream. To simplify notation, we define $\tau(\Theta) = H_{\lfloor 1/\Theta \rfloor}/\Theta$, and thus in accordance with Lemma 1, we set for every $k = 1,\ldots,\ell$, $\tau_k = \tau$.[2]

Theorem 2. ***(Upper bound on the communication cost)*** *The omniscient algorithm (see Algorithms 1 and 2) exchanges in average no more than*

$$2mS(\log m + \log n) \sum_{k=1}^{\ell} \frac{c_k}{\sum_{i=0}^{\tau-1} \mathbb{P}\{T_{c_k,n_k}(v) > i\}} \text{ bits.} \quad (2)$$

Proof: From Algorithms 1 and 2, node $s \in \mathcal{S}$ triggers a transmission with the coordinator each time one of its buffers is full or upon timeout. Thus, assuming that s maintains a single buffer Γ_k, then for m_s large, the expected number of times node s sends the content of Γ_k to the coordinator, denoted $C_{k,s}$, verifies

$$C_{k,s} \leq \frac{m_s}{\mathbb{E}[\min(T_{c_k,j_k}(p)),\tau]} = \frac{m_s}{\sum_{i=0}^{\tau-1} \mathbb{P}\{T_{c_k,j_k}(p) > i\}},$$

where $\mathbb{P}\{T_{c_k,j_k}(p) > h\} =$

$$\sum_{i=0}^{c_k-1} (-1)^{c_k-1-i} \binom{j_k-i-1}{j_k-c_k} \sum_{J\in S_{i,j_k}} (p_0+P_J)^h.$$

It has been observed (Theorem 3 in [2]) that for vectors $p = (p_{j,s})_{j\in J_k}$ and with $p_0 = 1-P_{J_k}$ and $0 < \Theta \leq p_{j,s}$,

$$\mathbb{P}\{T_{c_k,j_k}(v) > h\} \leq \mathbb{P}\{T_{c_k,j_k}(p) > h\}.$$

Thus, we have

$$C_{k,s} \leq \frac{m_s}{\sum_{i=0}^{\tau-1} \mathbb{P}\{T_{c_k,j_k}(p) > i\}} \leq \frac{m_s}{\sum_{i=0}^{\tau-1} \mathbb{P}\{T_{c_k,n_k}(v) > i\}}.$$

Finally, we need to determine the number of bits sent each time buffer Γ_k is full or upon timeout. From Algorithm 1, the sending of message "identify" requires $\log m_s + c_k(\log n + \log m_s)$ bits to be transmitted to the coordinator (where $\log m_s$ represents the number of bits needed to code item frequencies, and $\log n$ the one needed to code item identifiers). By Algorithm 2, this triggers a round trip communication between the coordinator

[2]For the sake of clarity, unless otherwise specified, we will use the notation τ to denote $\tau(\Theta)$ if there is no ambiguity.

and the remaining $S-1$ nodes to collect the frequencies of all potential icebergs of Γ_k (Line 47 of Algorithm 1 and Line 7 of Algorithm 2). This generates respectively $S-1$ messages of $c_k \log n$ bits from the coordinator and a message of $\log m_{s'} + c_k(\log m_{s'} + \log n)$ bits from each node $s' \neq s$ in $\mathcal{S}$, where $m_{s'}$ is the size of $\sigma_{s'}$. As the sum of all the local streams is equal to m, the number of bits sent because Γ_k is full is less than $2Sc_k(\log m + \log n)$. Thus, the fact that $C_{k,s}$ must be computed for all the buffers at every node $s \in \mathcal{S}$, allows us to complete the proof of the lemma.

Note that when $\Theta \to 0$, which is the case in the global iceberg problem, we have $\tau \to \infty$, and the denominator in Relation (2) tends to $n_k(H_{n_k} - H_{n_k - c_k})$ [3]. ∎

3) Correctness of the omniscient algorithm: We now prove that the omniscient algorithm solves the global iceberg problem.

Theorem 3. ***(Correctness)*** *The omniscient algorithm (see Algorithms 1 and 2) deterministically and exactly solves the global iceberg problem (*i.e*, $\delta = 0$ and $\varepsilon = 0$).*

Proof: The proof consists in showing that the algorithm does not output any items j such that $f_j < \Theta m$ (*i.e.*, no false positive) and returns all items j such that $f_j \geq \Theta m$ (*i.e.*, no false negative).

Let us first focus on false positives. Suppose by contradiction that the coordinator returns at time t some item j such that $f_j < \Theta m$. Then, by Algorithm 2, this means that the coordinator has received from some node $s \in \mathcal{S}$ a message "identify" for which $j \in$ *Freq*. By Algorithm 1, this can only happens if node s has identified j as a potential global iceberg, that is, $p_{j,s} \geq \Theta$ (see Line 9 of Algorithm 1). Now, upon receipt of *Freq*, the coordinator collects from the other $S-1$ nodes the frequency of each item $i \in$ *Freq*, and in particular j frequency, as well as the current size of their input stream (see Lines 7–10 of Algorithm 2). By Lines 16 and 17, the coordinator removes from *Freq* all the items whose frequency is less than Θm, and in particular item j. Thus the coordinator cannot return j at time t.

A false negative means that the coordinator does not return a true global iceberg. Suppose that there exists j such that $f_j \geq \Theta m$ and j is not returned. Thus there exists at least one stream σ_s such that $p_{j,s} \geq \Theta$. By Lines 10–19 of Algorithm 1, j is inserted in the buffer Γ_k that matches its occurrence probability $p_{j,s}$ (if not already present). By Lines 24 and 35, j is sent to the coordinator after at most $H_{\lfloor 1/\Theta \rfloor}/\Theta$ reading. By applying an argument similar to the above case, we get a contradiction with the assumption of the case. ∎

To summarize, Theorem 3 has shown that the omniscient algorithm accurately tracks global icebergs, even if they are hidden in distributed streams. In addition, we have provided with Theorem 2 an upper bound on the communication cost between the S nodes and the coordinator. This bound is reached when each buffer is fed with items that all occur with the same probability.

V. Knowledge-free algorithm to track global icebergs

The algorithm we have proposed in Section IV relies on the assumption that upon receipt of item j at node $s \in \mathcal{S}$, its probability of occurrence $p_{j,s}$ in the full stream σ_s is known and that node s has no memory space restriction. Clearly both assumptions are unrealistic, in particular in presence of an adversary that may modify on the fly the occurrence probability of any items in the streams to affect the global iceberg detection algorithm.

A. Principles of the knowledge-free algorithm

We now propose an algorithm, called hereafter *knowledge-free algorithm*, that solves the global iceberg problem without making any assumption on the probability of occurrence of items in σ_s, with $s \in \mathcal{S}$. Instead, this algorithm builds an estimation of this knowledge from a compact synopsis of σ_s computed on the fly and with a little space (with respect to the size of the input stream and the size of the domain from which items are drawn). This compact synopsis, called in the following *Count-Min Sketch* $\hat{F}$, is built from the algorithm proposed by Cormode and Muthukrishnan [6]. In the following this algorithm is called the CM algorithm. For self-containment reasons, prior to presenting our algorithm, we recall some definitions and describe the main features of the CM algorithm.

1) Preliminaries:

a) (ε, δ)-approximation: A randomised algorithm $\mathcal{A}$ is said to be an (ε, δ)-approximation of a function ϕ on a stream σ if for any sequence of items in σ, $\mathcal{A}$ outputs $\hat{\phi}$ such that $\mathbb{P}\{\mid \hat{\phi} - \phi \mid \geq \varepsilon\phi\} \leq \delta$, where $0 < \varepsilon, \delta < 1$ are given as parameters of the algorithm.

b) 2-universal Hash Functions: A collection $\mathcal{H}$ of hash functions $h : \{1, \ldots, n\} \to \{0, \ldots, n'\}$ is said to be *2-universal* if for every two different items $i, j \in \{1, \ldots, n\}$, $\mathbb{P}_{h \in \mathcal{H}}\{h(i) = h(j)\} \leq 1/n'$. Note that this is the probability of collision obtained if the hash function assigned truly random values to any $i \in \{1, \ldots, n\}$. Carter and Wegman [5] provide an efficient method to build large families of hash functions approximating the 2-universal property.

c) The Count-Min (CM) sketch algorithm [6]: The Count-Min Sketch algorithm approximates the frequencies of each item present in a stream. This approximation is done on the fly (items are read only once) and requires sub-linear memory space in the size of the input stream and in the size of the domain of the items. Briefly, the CM algorithm maintains a two-dimensional array $\hat{F}$ of $s_1 \times s_2$ counters with $s_1 = \lceil \log(1/\delta) \rceil$ and $s_2 = \lceil e/\varepsilon \rceil$, and by using 2-universal hash functions $h_1, \ldots, h_{s_1}$ (where $e = \exp(1)$). Parameters ε and δ, with $0 < \varepsilon, \delta < 1$, represent respectively the accuracy of the approximation, and the probability with which the accuracy holds. Both ε and δ are imposed by the user. Each time an item j is read from the input stream, this causes one counter per line to be incremented, *i.e.*, $\hat{F}[u][h_u(j)]$ is incremented for all $u \in \{1, \ldots, s_1\}$. Thus at any time, the sum of the counters of any given line is equal to the number of items m read from the input stream. When a query is issued to get an estimate $\hat{f}_j$ of the number of times item j has occurred since the inception of the stream, the returned value corresponds to the minimum among the s_1 values of $\hat{F}[u][h_u(j)]$, $1 \leq u \leq s_1$. Algorithm 3 presents the pseudo-code of the Count-Min Sketch algorithm. The error of the estimator in answering a query for $\hat{f}_j$ is within a factor of $\varepsilon(m - f_j)$. The space required by this algorithm is proportional to $\log(1/\delta)/\varepsilon$, and the update time per element is sub-linear in

Algorithm 3: Estimating the Frequency of Items in a Stream (Count-Min Sketch algorithm) [6]

Input: An input stream σ; real values δ and ε;
Output: The estimate $\hat{f}_j$ for the frequency of any item j read from the input stream

$s_1 \leftarrow \lceil \log(1/\delta) \rceil$;
$s_2 \leftarrow \lceil e/\varepsilon \rceil$;
$\hat{F}[s_1][s_2] \leftarrow 0$;
Choose s_1 2-universal hash functions $h_1, \ldots, h_{s_1} : \mathcal{N} \rightarrow \{1, \ldots, s_2\}$;
for $j \in \sigma$ **do**
  for $v = 1$ **to** s_1 **do**
    $\hat{F}[v][h_v(j)] \leftarrow \hat{F}[v][h_v(j)] + 1$;
  end
end
Upon query of *Estimate*(f_j): **return** $\hat{f}_j = \min_{1 \leq v \leq s_1} \hat{F}[v][h_v(j)]$;

Algorithm 4: Knowledge-free algorithm run at any node $s \in \mathcal{S}$

Input: An arbitrary input stream σ_s; real values δ and ε; Θ: threshold qualifying an item as a global iceberg; r: proportion size parameter of the ℓ buffers ;

Initialisation of the data structure $\hat{F}$ of Algorithm 3 with $s_1 \leftarrow \lceil \log(1/\delta) \rceil$ and $s_2 \leftarrow \lceil 2(1-\Theta)/\varepsilon\Theta \rceil$;
for $j \in \sigma_s$ **do**
  Task T_1 **:**
    $\hat{F} \leftarrow$ Algorithm presented in Figure 3 fed with j;
  end
  Task T_2 **:**
    execute Algorithm 1 such that
    lines 9–13 are modified as below:
    if $\hat{f}_{j,s}/m_s \geq \Theta$ **then**
      if $\hat{f}_{j,s}/m_s \leq \Theta + (1-\Theta)/2^{\ell-1}$ **then**
        $k \leftarrow \ell$;
      else
        $k \leftarrow \min\{h \in \{1, \ell-1\} \mid \Theta + (1-\Theta)/2^h < \hat{f}_{j,s}/m_s\}$;
      end
    end
    [. . .]
    lines 21–22 and lines 32–33 are modified as below:
    foreach $\{j\} \in \Gamma_k$ **do**
      $Freq \leftarrow Freq \cup (j, \hat{f}_{j,s})$;
    end
    [. . .]
    lines 41–42 are modified as below:
    foreach $j \in Freq$ **do**
      $Freq \leftarrow (j, \hat{f}_{j,s})$;
      [. . .]
    end
  end
end

the size of the sketch [6]. Specifically, after reading m items from the stream, we have

$$\forall j \in \mathcal{N},\ \mathbb{P}\left\{\hat{f}_j - f_j \geq \varepsilon\,(m - f_j)\right\} \leq \delta. \tag{3}$$

2) The Knowledge-free Algorithm: The knowledge-free algorithm we propose combines the algorithm run by the coordinator (see Algorithm 2) and the algorithm presented in Algorithm 4. This algorithm, local to each node $s \in \mathcal{S}$ consists in the execution of two parallel tasks: Task$_1$ and Task$_2$. Task$_1$ maintains an estimation of the frequency of occurrence of each received item in σ_s and Task$_2$ is a simple adaptation of the omniscient algorithm presented in Algorithm 1. The adaptation consists in substituting the estimation $\hat{f}_{j,s}$ of item j frequency in σ_s for the exact frequency $F[j]$ and in replacing the probability of occurrence $p_{j,s}$ in the full stream with $\hat{f}_{j,s}/m_s$, where m_s is the number of items received in the stream σ_s so far. The CM algorithm strongly relies on 2-universal hash functions, to guarantee an (ε, δ)-approximation of item frequencies. If all the S nodes use the same set of hash functions in $\mathcal{H}$, one may argue that the adversary could launch a dictionary attack to easily tamper with these estimations. In our solution, each of the S nodes locally and independently chooses at random its hash functions from $\mathcal{H}$. We show in Section V-B that this does not prevent the coordinator from (ε, δ)-approximating the detection of global icebergs.

B. Analysis of the knowledge-free algorithm

The following lemma shows that the knowledge-free algorithm is an (ε, δ)-approximation of the omniscient one, for any $\varepsilon \in (0, 1)$ and probability of failure $\delta \leq 1/2$. This is achieved by sharpening the bounds obtained in [6] and by relying on Theorem 3.

Lemma 4. *For a stream σ, for any $\varepsilon \in (0, 1)$ and probability of failure $\delta \leq 1/2$, Algorithm 3 run with $s_1 = \lceil \log(1/\delta) \rceil$ and $s_2 = \lceil 2(1-\Theta)/\varepsilon\Theta \rceil$ guarantees that*

$$\forall j \in \sigma, \begin{cases} \mathbb{P}\left\{\hat{f}_j - f_j \geq \varepsilon f_j\right\} \leq \delta & \text{if } p_j \geq \Theta \\ \mathbb{P}\left\{\hat{f}_j - f_j \geq \varepsilon\,(m - f_j)\right\} \leq \delta & \text{otherwise.} \end{cases}$$

Proof: For space reasons the proof is omitted from this paper but appears in the companion paper [1]. ■

Theorem 5. *The knowledge-free algorithm presented in Algorithms 2 and 4 uses $\mathcal{O}((\log n + \log m)\log(1/\delta)(1/\Theta - 1)/\varepsilon + (\log S \log n)/\Theta)$ bits of space on each node to (ε, δ)-approximate tracking of global icebergs in arbitrarily distributed input streams.*

Proof: By Lemma 4 applied to any stream σ_s of length m_s, with $s \in \mathcal{S}$, Algorithm 3 returns an (ε, δ)-approximation of $f_{j,s}$ for all $j \in \sigma_s$. Thus, using $((1/\varepsilon\Theta)\log(1/\delta)(\log n + \log m_s))$ bits of space, Algorithm 4 (ε, δ)-approximates the tracking of potential global icebergs at each node s. We now demonstrate that the coordinator correctly (ε, δ)-approximates the frequency of each item that has been locally detected as potential global icebergs at any node s. We have,

$$\begin{aligned} &\mathbb{P}\left\{\sum_{s \in \mathcal{S}} \hat{f}_{j,s} - f_j \geq \varepsilon(m - f_j)\right\} \\ &\quad = \mathbb{P}\left\{\sum_{s \in \mathcal{S}} \left(\hat{f}_{j,s} - f_{j,s}\right) \geq \varepsilon(m - f_j)\right\} \\ &\quad = \mathbb{P}\left\{\sum_{s \in \mathcal{S}} \min_{u \in \{1, \ldots, s_1\}} X_{j,s}^{(u)} \geq \varepsilon(m - f_j)\right\}, \end{aligned}$$

where $X_{j,s}^{(u)}$ denote the random variable that measures, given $u \in \{1, \ldots, s_1\}$, the excess of a counter $\hat{F}[u][h_u^s(j)]$ on a

specific node s. We clearly have $\min_{u\in\{1,\ldots,s_1\}} \sum_{s\in\mathcal{S}} X_{j,s}^{(u)} \geq \sum_{s\in\mathcal{S}} \min_{u\in\{1,\ldots,s_1\}} X_{j,s}^{(u)}$. Then, by the mutual independence of the s_1 estimators, and since $\mathbb{E}[X_{j,s}^{(u)}] \leq \frac{m_s - f_{j,s}}{s_2}$ [6], we have

$$\begin{aligned}
&\mathbb{P}\left\{\sum_{s\in\mathcal{S}} \hat{f}_{j,s} - f_j \geq \varepsilon(m - f_j)\right\} \\
&\leq \prod_{u=1}^{s_1} \mathbb{P}\left\{\sum_{s\in\mathcal{S}} X_{j,s}^{(u)} < \varepsilon(m - f_j)\right\} \leq \prod_{u=1}^{s_1} \frac{\mathbb{E}\left[\sum_{s\in\mathcal{S}} X_{j,s}^{(u)}\right]}{\varepsilon(m - f_j)} \\
&= \prod_{u=1}^{s_1} \frac{\sum_{s\in\mathcal{S}} (m_s - f_{j,s})}{s_2\varepsilon(m - f_j)} = \prod_{u=1}^{s_1} \frac{m - f_j}{s_2\varepsilon(m - f_j)} \leq \frac{1}{2^{s_1}} \leq \delta.
\end{aligned}$$

Similarly, by applying the same argument as above and the one of Lemma 4, we get that $\mathbb{P}\left\{\sum_{s\in\mathcal{S}} \hat{f}_{j,s} - f_j \geq \varepsilon f_j\right\} \leq \delta$ if $f_j \geq \Theta m$ in the global stream σ. Hence, Algorithms 2 and 4 (ε, δ)-approximate the omniscient approach presented in Section IV-A. By Theorem 3, the omniscient strategy implements a global iceberg detection, robust to any biased distributed input streams, which completes the second part of the proof. Finally, by Lemma 4, on each node, $\mathcal{O}((\log n + \log m_s)\log(1/\delta)(1/\Theta - 1)/\varepsilon)$ bits of space are required to (ε, δ)-approximate item frequencies. Moreover, the space required to locally track potential icebergs is the sum of the size used by each buffer Γ_k, that is $\sum_{k=1}^{\ell} c_k \log n$ bits. By construction $c_k \leq 1/\Theta$. Thus, an upper bound of the total space used is equal to $\mathcal{O}((\log S \log n)/\Theta)$. ■

VI. Experimental Evaluation

This section describes the main results obtained from the experiments run with the knowledge-free algorithm. All these experiments have been achieved on a testbed of $S = 20$ single-board computers (Raspberry Pi Model B) and two servers connected through a Gigabit Ethernet network. Single-board computers are small computers with limited memory and storage capacities (100 Mbps, 250MB RAM and 700MHz CPU). Each Raspberry Pi hosts a node, one of the two servers hosts the coordinator, while the other one generates all the S streams. Objective of these experimentations is a proof of concept. Actually, real routers (as for example, Cisco or Juniper type M or T, 10 Gbps throughput, from 768MB to 4GB of memory, Pentium CPU 1GHz) would definitively handle the code run by the small Raspberries. Experiments have been conducted on different types of streams and for different parameters settings. The single-board computers have been fed with both real-world datasets and with synthetic traces. This allows to capture phenomenons that may be difficult to obtain from real-world traces, and thus allows to check the robustness of our algorithm. Each run has been executed a hundred times, and we provide the mean over the repeated runs, after removing the 1-st and 10-th deciles to avoid outliers.

The following metrics are evaluated: (i) recall and precision of our solution, (ii) frequency estimation of global icebergs, (iii) communication cost induced by our solution to detect global icebergs and, (iv) detection latency of global icebergs.

a) ***Simulation results with synthetic traces****:* Synthetic streams have been generated using Zipfian distributions with $\alpha \in \{0.5; 1.0; 2.0; 3.0\}$, denoted respectively by Zipf-0.5, Zipf-1, Zipf-2, and Zipf-3. Each stream is made of $m_s = 100,000$ items (*i.e.*, the global stream is made of $m = 2,000,000$ items) picking them from an universe $\mathcal{N}$ whose size is equal to $10,000$. Each node receives around $4,000$ items per second. Several values of Θ have been considered, namely, $\Theta \in [0.005; 0.1]$, with logarithmic steps, as well as different values of r, *i.e.*, $r \in [0.005; 1.0]$. For clarity reasons we show only the results of the experiments with the two extremum values of r, *i.e*, $r = 0.005$ (each buffer Γ_k contains a single item, which amounts for the nodes to directly send each potential global iceberg to the coordinator), and $r = 1$ (all the buffer Γ_k have their maximal size n_k, see Section 4). In the following both cases will respectively be referred to as *no buffer* and *with buffers*.

Θ	Count-Min	$\sum_{k=1}^{5} \lvert\Gamma_k\rvert$	$\lvert\Lambda_s\rvert$	Space Memory Gain
0.1	3.48 kB	0.77 kB	0.32 kB	98.57%
0.005	69.59 kB	7.20 kB	6.40 kB	74.00%

TABLE I
Memory usage with buffers, and counters of 32 bytes

The probability of failure δ and the error ε of the knowledge-free algorithm have been respectively set to $\delta = 0.1$ and $\varepsilon = 0.1$. Using our proven bounds (see Lemma 4), this leads for each node to a memory usage of up to 7.5% of the one that would require a naive algorithm that would maintain a counter for each received item to determine which of them are potential icebergs or not. Actually, we show that using as little as 1.43% of the space of the naive algorithm is sufficient (see Table I). This is achieved by reducing the number of columns s_2 in the Count-Min sketch from $s_2 = \lceil 2(1-\Theta)/(0.1\Theta) \rceil$ (see Lemma 4) to $s_2 = \lceil e/\Theta \rceil$.[3] Notice that in all the subsequent plots, the points are linked together by lines although they should appear as points. This has been only done to improve the readability of the plots.

b) ***Precision and recall****:* Figure 2 shows the precision and recall of our solution. By precision, we mean the number of global icebergs detected by our solution divided by the total number of detected items. By recall, we mean the number of global icebergs detected by our solution divided by the total number of generated global icebergs. The main result is that recall is always equal to 1 whatever the input streams features. This is a very important property of our solution as it shows that all the global icebergs are perfectly detected (there are no false negatives), even if global icebergs are well hidden in all the distributed streams (*i.e.*, $\Theta = 0.005$). Now, this figure shows that precision is also very high when input streams follow Zipfian distributions with $\alpha \geq 1$ while for $\alpha = 0.5$ it decreases. This is easily explained by the fact that with slightly skewed distributions (case with $\alpha \leq 0.5$), there are very few frequent items (one or two), and the gap between those frequent items and the most frequent sparse items (the ones with a relative frequency close but less than Θ) is very small (see Table II). Thus even a small over-approximation of Count-Min can wrongly tag them as frequent items. Tagging two items as frequent items when there is a single one makes a precision

[3]For simplicity reason, we present memory usage with classical 32-bit coding. This is an overestimation of the requirement as only $\log n$ and/or $\log m$ are sufficient.

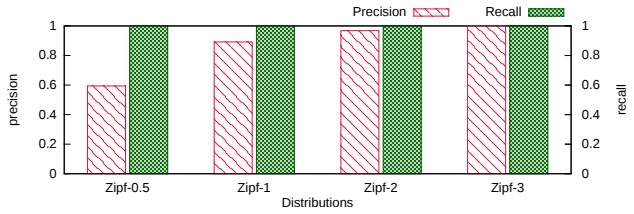

Fig. 2. Precision and recall as a function of the input distributions for $\Theta = 0.005$ - with buffers.

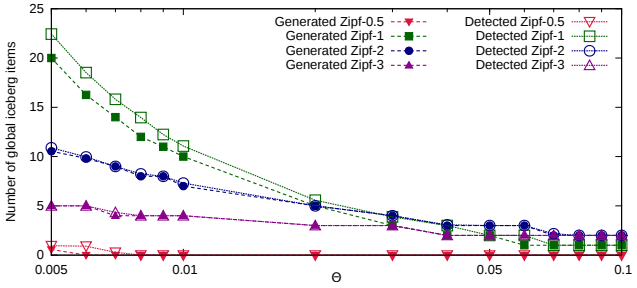

Fig. 3. Number of detected and generated global icebergs as a function of Θ, with buffers.

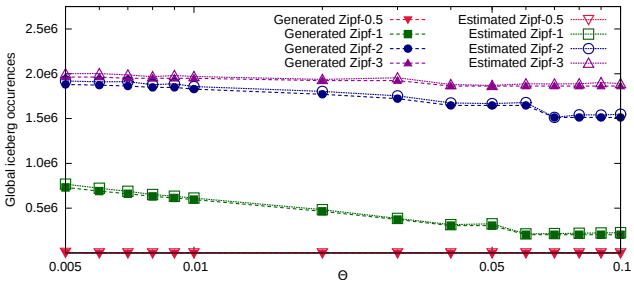

Fig. 4. Total number of occurrences of global icebergs (estimated and generated) as a function of Θ, with buffers.

Θ	Zipf-0.5	Zipf-1	Zipf-2	Zipf-3
0.1	N/A	5.1×10^{-2}	8.4×10^{-2}	7.3×10^{-2}
0.005	1.5×10^{-3}	2.4×10^{-4}	8.0×10^{-4}	2.8×10^{-3}

TABLE II
FREQUENCY GAPS.

equal to 0.5. Notice that both the precision and the recall are independent from the size of the local buffers Γ_k.

Figure 3 shows more details on the precision of our solution by showing the number of global icebergs that should be detected (referred to as *generated*) and the number of items that have been effectively detected as global icebergs by our knowledge-free algorithm (referred to as *detected*) as a function of Θ. By construction of the Count-Min sketch algorithm, item frequencies are over-estimated, thus the difference between the number of detected and effectively generated global icebergs corresponds to the number of false positives of our solution with the ε error. Note that the number of false positives may slightly increase with the size of the system due to the over-approximation of the sketch algorithm at each node. However, the absence of false negatives is guaranteed. For $\Theta = 0.005$, Figure 3 shows the results presented in Figure 2. When Θ increases, the precision of our solution drastically augments whatever the form of the input distributions.

*c) **Frequency estimation of global icebergs**:* Figure 4 compares the total number of occurrences of all the global icebergs as estimated by the knowledge-free algorithm with the total number of occurrences effectively generated. The overestimation of frequent items (*i.e.*, those whose relative frequency exceeds Θ) is negligible as expected by Lemma 4.

*d) **Communication cost induced to detect global icebergs**:* Figure 5 shows the ratio between the number of bits exchanged by the knowledge-free algorithm and the number of bits received in the input streams as a function of Θ. The primary remark is the negligible communication overhead induced by the algorithm to accurately detect global icebergs: strictly less than 8.5% of the size of all the distributed streams is exchanged by the nodes and the coordinator. This holds even for slightly skewed distributions, which by Theorem 2 are the distributions that lead to the upper communication bound. Notice the impact of buffers Γ_k on the communication overhead for these slightly skewed distributions, for small values of Θ. Note that in terms of messages, the ratio is even better (less than $1,12\%$ of the content of all the streams), which is due to the fact that messages exchanged between the different parties carry a set of items, while an item is equal to a message in the input streams (these plots appear in [1]). All these results are very impressive compared to [18], which for highly skewed input distributions, get a ratio to raw data equal to 75% (see Sect. 7.B [18]).

*e) **Detection latency**:* Figure 6 shows the latency of our solution to detect global icebergs. This latency is computed as the time needed to detect all the global icebergs divided by the time needed to receive all the streams. The important and very remarkable feature of our solution is that less than 3% of the global stream need to be received to detect all the global icebergs (when the streams are randomly ordered). The second remark is the impact of buffers Γ_k on the detection latency: locally keeping items in Γ_k prior to sending them to the coordinator increases the detection latency by a factor 4 while it decreases the communication overhead by a factor 2 (see Zipf-1 with $\Theta = 0.005$ in Figure 5. There must exist some optimal value of r that should minimise the detection latency and communication overhead. The study of this optimum is an open question.

*f) **Real trace results**:* We have fed our distributed algorithm with a real world dataset tracked during a DDoS attack; it has been retrieved from the CAIDA repository [8], [9]. The total raw data size over the 20 nodes is $m = 2 \times 10^8$ (with 32 bytes for flow size. There are in total $n = 825,695$ unique destination address in this dataset (items in these experiments are destination addresses). This represents a 15-minute trace of traffic, monitored on OC192 Internet backbone links. Note that this dataset is definitely larger than the synthetic ones, drastically increasing the number of collisions in the Count Min sketch. This dataset has been split into 20 streams, each one sent to a different node of our testbed. The main results drawn from the analysis of these streams are summarised in Table III.

The DDoS target is the unique global iceberg, with a prob-

Stream Size (m)	2×10^8
Distinct Items (n)	$825,695$
Global Icebergs	1
Detected Global Icebergs	1
False Positives	0
False Negatives	0
Global Iceberg Frequency	3.2×10^6
Estimated Global Iceberg Frequency	3.4×10^6
Communication Ratio (bits) without buffer	3.6×10^{-4}
Communication Ratio (bits) with buffers	3.2×10^{-4}

TABLE III
REAL-WORLD TRACE RESULTS - $\Theta = 0.1$

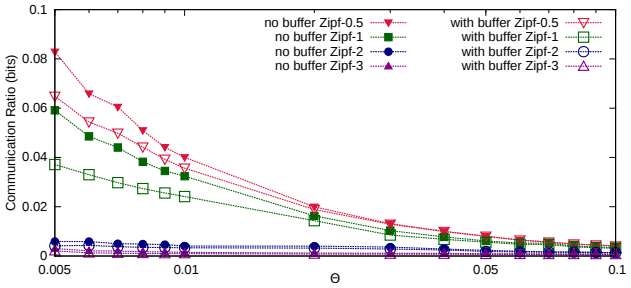

Fig. 5. Ratio between the number of bits exchanged by the knowledge-free algorithm and the number of bits (of the items) received in the 20 input streams as a function of Θ.

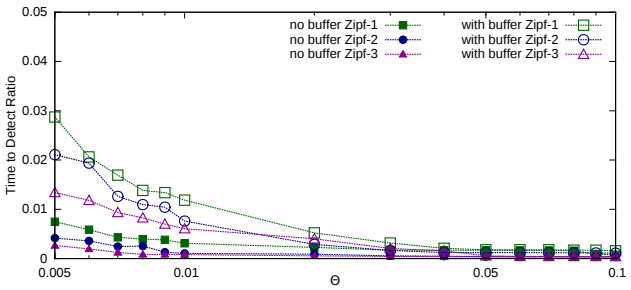

Fig. 6. Ratio between the time needed by the knowledge-free algorithm to detect all the iceberg items and the time needed to receive all the 20 input streams as a function of Θ.

ability of occurrence equals to 0.15 ($\Theta = 0.1$), as any other items occur with a probability lower than 5×10^{-3}. As illustrated in Table III, the DDoS target is correctly detected (no false positives or negatives). The communication ratio is at most equal to 3.6×10^{-4} (note that the ratio is computed as the size of the information exchanged between the parties divided by the size of the destination addresses contained in the full messages). These good results are mainly due to the large gap between probabilities of occurrence of the unique global iceberg and the other items, which occurs in classical DDoS attack.

VII. CONCLUSION

We have presented a distributed algorithm that deterministically detects all global icebergs either finely hidden or massively present in massive and physically distributed data streams. Moreover, we also (ε, δ)-approximate the size of each global iceberg. We have derived a thorough performance analysis by deriving bounds on the algorithm and performing real experiments on a cluster of single-board computers. These experiments have illustrated the enjoyable properties of our solution in terms of precision, recall, communication overhead and detection latency.

As future work, we plan to extend our solution to fit the requirements of long-lasting applications, including sensor-based monitoring ones. In this context, global icebergs do not necessarily reflect attacks but rather significative short-lived events. Extending our solution to the windowing data stream model should fit the specificities of such applications.

REFERENCES

[1] E. Anceaume, Y. Busnel, N. Rivetti, and B. Sericola. Identifying global icebergs in distributed streams. Technical Report https://hal.archives-ouvertes.fr/hal-01141829, HAL, 2015.

[2] E. Anceaume, Y. Busnel, E. Schulte-Geers, and B. Sericola. Optimization results for a generalized coupon collector problem. *Journal of Applied Probability*, 2016. To appear.

[3] E. Anceaume, Y. Busnel, and B. Sericola. New results on a generalized coupon collector problem using markov chains. *Journal of Applied Probability*, 52(2), 2015.

[4] B. H. Bloom. Space/time trade-offs in hash coding with allowable errors. *Communications of the ACM*, 13(7):422–426, 1970.

[5] J. L. Carter and M. N. Wegman. Universal classes of hash functions. *Journal of Computer and System Sciences*, 18:143–154, 1979.

[6] G. Cormode and S. Muthukrishnan. An improved data stream summary: the count-min sketch and its applications. *Journal of Algorithms*, 55(1):58–75, 2005.

[7] 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.

[8] The CAIDA UCSD "Anonymized Internet Traces 2008" Dataset. http://www.caida.org/data/passive/passive-2008-dataset.xml. Cooperative Association for Internet Data Analysis, April 2008.

[9] The CAIDA UCSD "DDoS Attack 2007" Dataset. http://www.caida.org/data/passive/ddos-20070804-dataset.xml. Cooperative Association for Internet Data Analysis, February 2010.

[10] E.D. Demaine, A. Lopez-Ortiz, and J.I. Munro. Frequent estimation of Internet packet streams with limited space. In *Proceedings of the 11th European Symposium on Algorithms (ESA)*, 2003.

[11] C. Estand and G. Varghese. New directions in traffic measurement and accounting: Focusing on the elephants, ignoring the mice. *ACM Transactions on Computer Systems (TOCS)*, 21(3):270–313, 2003.

[12] P. Flajolet, D. Gardy, and L. Thimonier. Birthday paradox, coupon collectors, caching algorithms and self-organizing search. *Discrete Applied Mathematics*, 39:207–229, 1992.

[13] R.M. Karp, S. Shenker, and C.H. Papadimitriou. A simple algorithm for finding frequent elements in streams and bags. *ACM Transactions on Database Systems (TODS)*, 28(1):51–55, 2003.

[14] A. Manjhi, V. Shkapenyuk, K. Dhamdhere, and C. Olston. Finding (recently) frequent items in distributed data streams. In *Proceedings of the 21st IEEE International Conference on Data Engineering (ICDE)*, 2005.

[15] G.S. Manku and R. Motwani. Approximate frequency conuts over data streams. In *Proceedings of the 28th International Conference on Very Large Data Bases (VLDB)*, 2002.

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

[17] K. Yi and Q. Zhang. Optimal tracking of distributed heavy hitters and quantiles. *Algorithmica*, 65:206–223, 2013.

[18] Q. Zhao, A. Lall, M. Ogihara, and J. Xu. Global iceberg detection over distributed streams. In *Proceedings of the 26th IEEE International Conference on Data Engineering (ICDE)*, 2010.

[19] Q. Zhao, M. Ogihara, H. Wang, and J. Xu. Finding global icebergs over distributed data sets. In *Proceedings of the 25th ACM SIGACT- SIGMOD-SIGART Symposium on Principles of Database Systems (PODS)*, 2006.