Uncovering Global Icebergs in Distributed Monitors

Guanyao Huang*, Ashwin Lall†, Chen-Nee Chuah*, and Jun Xu†
* Department of Electrical and Computer Engineering
University of California, Davis
† College of Computing
Georgia Institute of Technology

Abstract—**Security is becoming an increasingly important QoS parameter for which network providers should provision. We focus on monitoring and detecting one type of network event, which is important for a number of security applications such as DDoS attack mitigation and worm detection, called distributed global icebergs. While previous work has concentrated on measuring local heavy-hitters using "sketches" in the non-distributed streaming case or icebergs in the non-streaming distributed case, we focus on measuring icebergs from distributed streams. Since an iceberg may be "hidden" by being distributed across many different streams, we combine a sampling component with local sketches to catch such cases. We provide a taxonomy of the existing sketches and perform a thorough study of the strengths and weaknesses of each of them, as well as the interactions between the different components, using both real and synthetic Internet trace data. Our combination of sketching and sampling is simple yet efficient in detecting global icebergs.**

I. Introduction

Recently, network security has become closely intertwined with quality-of-service. Various types of attacks on network security, such as worms, viruses, and denial-of-service (DoS) attacks, have been adversely affecting application performance. Since application performance is the primary goal of QoS, it is becoming increasingly important to develop better tools for security. Security solutions typically involve "measurement" and "detection" modules to uncover unwanted traffic or networking activity. In this paper we focus on detecting one such type of security event: distributed icebergs.

An iceberg is a network traffic flow (e.g., all the traffic coming from a particular source IP address) that has high aggregate volume across many different monitors, even if it does not appear large at any single monitor. Detecting this type of event is important for a number of applications, including detecting DDoS attacks [1], finding heavy-hitters in Content Delivery Networks [2], discovering worms and other anomalies [3], as well as ensuring SLA compliance [4]. Applications that detect DDoS attacks need to find destination IP addresses that occur frequently across multiple ingress points. In worm detection, the worm signature which appears widely and frequently can be viewed as a global iceberg among all the machines in the network. Global iceberg detection is therefore important for a variety of applications, especially those related to network measurement and security.

Monitoring and analyzing current Internet traffic is challenging due to the increasing link speed and traffic volume. To cope with the high speed, it is only viable to perform low rate sampling [5] or very succinct sketching. The sketches can be constructed using small but high-speed SRAM, by either intelligent sampling or data streaming algorithms. Recently, there has been a lot of study on detecting local heavy-hitters (also called elephants) in both networking and database scenarios [6], [7] based on sketching. The problem is difficult since there is no prior information about the identities of heavy-hitters and the sheer volume of the data precludes maintaining per-flow state. The proposed methods include the count-min sketch [6], the count-sketch [8], sticky sampling, lossy counting [9], shared-state sampling (SSS) [10], multi-stage filters and sample-and-hold [7]. They use hash-based sketches, intelligent sampling, or intricate data structures [11] to preserve the information of local elephants.

However, the aforementioned work only focused on detecting high frequency items (flows) at a single monitor. Since global icebergs may be adversarially split in some scenarios, they are not necessarily local heavy-hitters. It is therefore not enough to only report these local elephants. For example, a DDoS attack might be generated from many botnet zombies such that the traffic volume is moderate at local monitors to evade being detected. For these security-related applications we need to accurately measure the aggregated traffic volume from distributed monitors. The detection mechanism should be oblivious to how the iceberg is split.

To detect global icebergs, distributed monitors should first measure local traffic for iceberg candidates and then report the measured datasets to a central server, which aggregates count values and extracts the most frequent ones. There are therefore two aspects related to this problem: how to measure local traffic and how to report the datasets. For the first aspect, it is important to detect not only local heavy-hitters but also the split global icebergs. For the second aspect, it is necessary to reduce the communication cost.

Associated with these two aspects there are four performance metrics. We identify three QoS metrics for the identification of icebergs: false positive rate, false negative rate, and the average relative error of the size of the icebergs. Besides these, the communication cost should be reduced to orders of magnitude smaller than in the naive solution in which every local flow is stored and reported.

We attack the global iceberg detection problem using a combination of local sketching and uniform sampling at each distributed monitor. We will demonstrate that both these components are necessary for accurate detection of global icebergs. Uniform sampling across all the monitors guarantees that the split icebergs will also get detected. The local sketches are necessary for obtaining accurate estimates for the counts of the icebergs. We perform a comprehensive study on the combinations, showing how our solutions perform for each of the performance metrics. Our contributions are as follows:

- First, we introduce the new problem of detecting distributed icebergs in the streaming setting, i.e., with non-zero local measurement errors. We propose a combined sketching and sampling approach for detecting and measuring them. While sketches are useful in detecting local elephants, we verify the importance of uniform sampling in capturing distributed global icebergs.
- Second, we perform a thorough evaluation of the solution space that is based on sampling and sketching methodology that we described above. We develop a taxonomy of different sketches based on the techniques they use. According to this taxonomy, we develop and compare two main strategies that combine sketching and uniform sampling in local measurement using real network data. Our comparisons are done using the four metrics mentioned above.
- Third, we compare the performances of existing sketches for different split patterns of global icebergs. This helps in understanding the relationship between detection efficiency and the underlying traffic pattern. We show that our methodology is robust against several split patterns that may be used by an adversary to hide an attack.
- Finally, we modify an existing algorithm to improve global iceberg detection in every metric. Although this modification is not optimal for detecting local elephants, we demonstrate its efficacy in detecting global icebergs when it is combined with uniform sampling.

As mentioned earlier, one challenge for this problem is to minimize the amount of communication. We simplify this problem by fixing the amount of memory at the distributed monitors and reporting the entire memory content to the central server. Throughout this paper we focus on the task of improving the quality of local measurements, and leave to future work the task of more efficiently reporting flows in the local memory to reduce the communication cost.

The rest of the paper is structured as follows. Section II discusses related work. In Section III we formally define our problem. We present our solution in Section IV and also describe the sketches that we use and a taxonomy via which we can compare their properties. Section V presents our experimental study on comparing different sketches as well as different combinations. Both real Internet traffic traces and synthetic data are used. Our results and observations are discussed in Section VI. Contributions and results are summarized in Section VII.

II. Related Work

While there has been considerable work that focuses on identifying heavy-hitters at a single node [6], [8], [11], [9], [7], we believe that detecting global icebergs is a far more interesting and challenging problem. Detecting global icebergs in the streaming environment has been closely studied in the theoretical setting [12], [13], [14]. However, our goal is to come up with practical algorithms for both detecting and estimating the size of icebergs in real data sets. Recently, Cormode et al. [15] proposed the problem of functional monitoring, where the local nodes continuously send updates only insofar as needed to satisfy some global constraint (e.g., detecting all the icebergs). Our work differs from theirs since we assume fixed measurement periods, which potentially allows us to have more communication-efficient mechanisms. Manjhi et al. [16] studied the problem of discovering icebergs in a distributed environment when nodes are arranged in a multi-level communication hierarchy. We study the simpler, practically motivated single-level communication scheme instead. Additionally, their scheme is dependent on a global iceberg being frequent in one or more local sites, an assumption that we avoid.

An alternate way of defining an iceberg is to consider top-k queries rather than fixing a threshold [17]. This is studied by Babcock et al. [17] in the distributed setting, and extended by Olston et al. [18] to support sum and average queries. These approaches aim to keep the local elephants aligned with the global ones and hence face the same issue as the above [16] solution—icebergs that are finely distributed among the local nodes are hard to discover.

In [19], Zhao et al. propose two methods for discovering global icebergs in distributed data. Their first scheme involves size-based sampling where they derive the optimal sampling rate for local sites. In their second scheme, they use Bloom filters to summarize the information at local nodes, and demonstrate a quantization scheme that is independent of the manner in which the iceberg may be split. Our goal is to also design split-independent algorithms. However, the main difference of our work from [19] is that we study the much more challenging streaming version of the problem, where we cannot assume that the local frequencies are known exactly at the local sites.

III. Formulation

Let us assume that there are n monitors, each with comparable numbers of items. At each of the n distributed points there are streams of items presented as update pairs $\langle id, c\rangle$, where id is the identity and c is the update value for this pair. An item may appear many times at the same monitor and also at multiple different monitors. The *global count* for an item is defined as the aggregate count value of all pairs which belong to this item across all the monitors. We denote by S the sum of the global counts of all the items. An iceberg is defined to be any item whose count aggregated across all n points is at least $T = \theta S$, for some $\theta \in (0, 1)$.

Our primary goal is to detect all such icebergs. Secondly, for each iceberg detected, we want to estimate its aggregate size accurately. Finally, we would like to achieve all this while keeping the false positive rate as low as possible. It may be the case that our methods will erroneously count items with counts slightly less than the threshold as icebergs. Since this is unavoidable, we introduce a parameter $\omega < \theta$ and say that we will not count as a false positive any item whose count lies within $(\omega S, \theta S)$. For many security applications, it is more important to not miss an iceberg than it is to not report items smaller than the iceberg threshold.

We define I to be the set of items whose global count values are larger than T. Then, our goal is to find an estimation set I^* as well as their estimated count values. The solution should satisfy the following requirements:

- The estimated set I^* should have few to no false negatives and false positives. Meanwhile, the estimated count values of items in $I \bigcap I^*$ should be close to the real count values of these items.
- The communication cost from the n nodes to the central server should be considerably smaller than the cost when every node aggregates and reports all their counts exactly.

In order to find I^*, every monitor first measures and generates a local data set which is a summary of the local traffic. The nodes then send the distributed data sets to the central server. We call these *estimated distributed streams* because exact summarization of the local streams is impossible due to resource constraints at the local nodes.

Note that, for traffic data, we may be interested in either the total number of bytes that comprise a flow or simply in the total number of packets (i.e., every update is of the form $\langle id, 1\rangle$ in our above notation). Our solution is for the latter case, though we note that it can be easily extended to apply to the former case as well.

IV. Sketching and Sampling

By understanding the capability of both uniform sampling and sketching, we propose efficient combinations of them to attack the global iceberg detection problem. The combination works as follows. Every monitor first uses uniform sampling and a sketch to summarize its traffic. We denote by LS the items that are captured locally by uniform sampling. Similarly, LH denotes the set captured by the local sketch, i.e., the local heavy-hitters. We exclude items in LS which are already present in LH since the sketch is better at estimating the count values of local elephants. Define $SH = LS - LH$ to be the set of items that are found exclusively by sampling. Finally, the distributed monitors send the LH and SH lists as well as their estimated count values to the central server, which aggregates these estimates to get the global icebergs. This methodology is illustrated in Figure 1 and the rationale behind it will be explained in the following section.

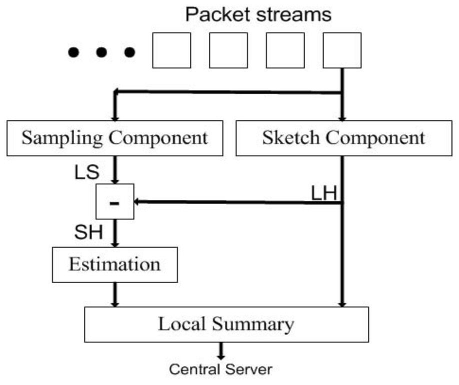


Fig. 1. Our solution

A. Advantages of Sketching and Sampling

In this section we summarize the properties of sketching and uniform sampling to demonstrate the importance of combining both of them in capturing global icebergs. The sketches we study include count-min [6], count-sketch [8], sticky sampling, lossy counting [9], SSS [10], sample-and-hold and multistage filters [7]. We note that the sketch in [11] can only extract the top k most frequent items without estimating their values in one pass. Since our goal includes estimating the size, we do not consider this sketch further. Recently, [20] proposed a sketch to summarize local streams and provide unbiased estimators for subpopulation sizes. We omit this sketch since it retains more information (i.e., distributions) than we require and hence will be more expensive.

Among the sketches we compare, sticky sampling, lossy counting, and sample-and-hold continuously remove the potential small and medium flows in order to vacate space for the (later-coming) elephants. SSS, multistage, count-min and count-sketch use arrays of counters to maintain the sizes of every flow; the counters are indexed by hash functions applied to the flow identities. An elephant is detected if the estimated size from the counters is above a threshold. We make the following observations on the seven sketches above:

Observation I*: In the seven sketches above, the estimation error for one item (flow) is independent of its real count value.*

For sketches that maintain counts using hashing, the estimation error is caused by hash collision which is independent of the real sizes. The error of sample-and-hold is introduced by sampling the initial packets before the item is first sampled, which depends only on the sampling probability. In sticky sampling, the error is introduced by missing the first packets as well as randomly deleting packets from the sketch, which are both independent of the real count value. In lossy counting, the error is introduced by estimating the count values of the item before it is inserted into the sketch, which is related to the arrival pattern instead of the size.

Therefore, the estimation error for one item can be viewed as a function of traffic pattern and sketch setting, instead of a function of its own count value. For instance, the error introduced by count-min is bounded by a constant multiplied by the first order norm of the total traffic volume [6], which is relatively small compared to large items; however, for small items, it might be larger than the error introduced by sampling.

Observation II*: The estimation variance by reverting uniform sampling is an increasing function of the true count value.*

The reason why we combine sketches and uniform sampling is therefore obvious. First, the sketches generally sacrifice the estimation accuracy of small- to medium-size items in order to preserve information for large items. They are insufficient since the global iceberg can be split and appear as non-elephant flows locally. Second, even if the global iceberg is split, the probability that this item is sampled by uniform sampling remains large. This suggests that we can use sketches to capture local heavy-hitters and use uniform sampling to capture local non-elephants which might be global icebergs.

B. Why Sampling and Sketching are Both Necessary

Based upon the observations above, we further justify our combination by analytically studying their different roles in detecting global icebergs. First, we show that local sketches can assist us in detecting global icebergs. Second, we show that sampling helps us estimate their counts when they are hidden as non-elephants locally. Namely, the sketching can capture iceberg identities while the uniform sampling helps more accurately estimate their sizes.

Recall that our goal is to find all icebergs with aggregate frequency at least θS, where S is the total packet count across all n monitors. Let us denote by $s_1, s_2, \ldots, s_n$ the packet counts at each of the monitors. We fix a single global iceberg with count $c \geq \theta S$ that is distributed across the monitors with counts $c_1, c_2, \ldots, c_n$. Suppose that the local elephant detection sketches can successfully identify all items that comprise at least θ' of the local traffic. Let us assume, without loss of generality, that our item of interest does not appear as a local elephant at monitors $1, \ldots, k$, i.e., for each $i \in \{1, \ldots, k\}, c_i \leq \theta' s_i$. Then, at all the other monitors, this item appears with count at least

$$\begin{aligned} \sum_{i=k+1}^{n} c_i &= c - \sum_{i=1}^{k} c_i \geq \theta S - \sum_{i=1}^{k} c_i \\ &\geq \theta S - \sum_{i=1}^{k} \theta' s_i \geq \theta S - \theta' S. \end{aligned}$$

Hence, assuming that the local sketches detect all the θ' local elephants and report their full counts, at least $(\theta - \theta')S$ of the iceberg's count gets reported to the central server. Now, if we choose $\theta' = \theta - \omega$, we get that at least ωS of the iceberg count gets reported to the central server, which can safely declare the item an iceberg.

There are two drawbacks to the above analysis. First, sketches are not guaranteed to return the θ' local elephants; they may miss some with some small probability. Second, while some local monitors successfully report the iceberg identity, its aggregated size may be severely underestimated since parts of it are not reported. Both these issues can be resolved by the sampling component. Since packets are sampled uniformly at all the measurement points, the results of sampling are independent of the manner in which the global iceberg is split between them. In the case where the global iceberg is a not a local elephant, uniform sampling will still identify many samples of it; the expectation of the sampled flow size is linear with the actual flow size.

Now, one may ask why we did not rely exclusively on uniform sampling. This can be answered by studying the estimation error for reverting sampling. With sampling rate p, an item of count c has estimation variance $c(1-p)/p$ [22]. In comparison, the sketches that we are using have considerably less error. The combination therefore has smaller estimation error than exclusively using uniform sampling.

Therefore, leveraging the synergy of local elephant detection sketches and uniform sampling is ideal for this problem.

C. Taxonomy of Sketches and Combinations

In this section we present different combinations of uniform sampling and sketching. We categorize the combinations based on a novel taxonomy of the seven sketches mentioned earlier. The sketches can be divided into two main categories:

- **Implicit Counters (IC)**, in which an identity can be used to query counters for an estimate. SSS, multistage, count-min, and count-sketch belong to this category.
- **Explicit Sketches (ES)**, which includes sample-and-hold, sticky sampling, and lossy counting. The sketch itself does not include any counter arrays.

SSS, multistage, count-min, and count-sketch can be viewed as multiple stages of counting Bloom filters, where multiple stages are used to reduce hash collisions. Since the arrays of counters store the information of all the underlying traffic, they can be used to estimate the size of an identity. In contrast, sample-and-hold, sticky sampling, and lossy counting have no such counters. If one sampled item is not contained in the sketch, its count value can only be estimated from the uniform sampling component.

Based on the taxonomy of the sketches, there are two main combinations of sampling and sketching:

- **RS+S (Revert sampling + sampling):** Use a sketch to detect and report local elephants as well as sampled non-elephants to the central server. The count values for sampled non-elephants are estimated by reverting the sampling rate. This combination works for the seven sketches.
- **QC+S: (Query counter + sampling):** For the implicit counters, the count value for local non-elephants can be estimated by querying the counters. Distributed monitors may therefore report the counters, and use the sampled identities to query the collected counters. This combination works for the four sketches belonging to Implicit Counters.

The combinations correspond to different methods in estimating sizes of items in SH in Figure 1. For QC+S, there are three approaches depending on whether to send the counters to the server and how to query the counters. Our detailed study [23] reveals that the only approach that works well is when local monitors report the local heavy-hitters and sampled non-elephant identities as well as their count values estimated locally by querying the counters.

We next show the results of comparative studies where we tested our methodology, including the various combinations, on both real and synthetic data.

V. Evaluation

In this section we evaluate our approach through experiments performed on real and synthetic Internet traffic data. We compare the performance of different sketches to find the the best one that can be combined with uniform sampling to give the best estimate of the set(size) of global icebergs. We also use synthetic data to study how the split of the global iceberg influences the detection accuracy.

We compare the seven sketches as well as the naive sampling and naive iceberg approach. In the naive sampling approach, uniform sampling alone is used and count values are estimated by reverting sampling. In the naive iceberg approach, only local heavy-hitters are collected by the sketch and reported to the central server. We show that our combination outperforms these naive approaches for comparable communication costs. Besides the comparison of different sketches and combinations, we improve multistage by a slight modification of the original algorithm. This modification helps us understand the capability of combining sampling and sketching in global iceberg detection.

Internet traffic data serves as a good candidate to test our methodology since there are several practical applications for measuring globally distributed icebergs, e.g., detecting DDoS attacks, SLA measurements, and detecting Internet worm proliferation. The item identities over which we can detect global icebergs can be any subset of the standard IP five-tuple ($\langle$srcIP, dstIP, srcPort, dstPort, protocol$\rangle$). We choose the destination address for our experiments because of its usefulness (e.g., for DDoS attack detection), though our methods will work equally well for any other combination.

In these experiments, we first compare the four IC sketches using the QC+S combination, after which we compare the seven sketches using the RS+S combination. We then compare both QC+S and RS+S combinations to find the best combination and the best sketch. Finally, we use synthetic data to examine our best sketch as well as to study the influence of different split patterns on global iceberg detection.

A. Experiment Settings

1) Data Trace: We use NetFlow records from 11 sites collected during the same five-minute measurement period from the Abilene network [24]. The records are sampled data with sampling interval 100. We revert this sampled data to construct the original data. Since sampled records miss many of the smaller flows, we generate and insert small flows such that 10% of the flows contribute about 80% of all the traffic. We used the time stamps of flows to determine the ordering of the items in each stream, assuming that interleaving flows that were temporally close to each other did not significantly affect the performance of the sketches. Note that these traffic reverting procedures do not affect our set of global icebergs, though they do affect the performance of our sketching and sampling components.

2) Iceberg Parameters: In our experiments, we attempted to detect icebergs whose sizes are larger than $\theta = 0.1\%$ and $\theta = 0.01\%$ of the total number of packets across all the monitors.

As we mentioned earlier, it is difficult to distinguish items whose counts are slightly below the threshold from those above the threshold, so we introduce a parameter ω that is the ratio above which we do not penalize the detection algorithm for detecting a false positive. In all our experiments we set $\omega = 2\theta/3$, i.e., two thirds of the iceberg threshold.

3) Metrics Examined: In our experiments, we measure the following three metrics in diminishing order of importance. First, we are concerned with the probability that we miss an iceberg. It is of paramount importance that an anomaly is detected in most applications. Second, we measure the average relative error of the detected icebergs. We define relative error in the standard way: the absolute error divided by the true size. Accurate estimation of the sizes of the icebergs is important for gauging the magnitude of the problem. Finally, we aim to minimize the false positive rate. This is also important because a false positive will cause resources to be unnecessarily wasted in drilling down a false alarm.

4) Settings of sampling and sketches: We vary the uniform sampling interval from 100 to $30,000$ to study the influence of different sampling intervals. In our data, the number of distinct flows is about $70,000$ at every point. For Implicit Counters, we vary the counter settings from 4×5000 to 5×6000, since these sizes can yield better results than other settings [23]. [7] measures in continuous intervals of 5 seconds, while our experiments are done for one interval of 5 minutes.

All our local sketches use the same amount of memory and hence have identical parameters. We set the local SRAM size to be 1Mbit, which is about 5000 entries for local elephants, as suggested by [7]. In Implicit Counters, the counter occupies additional space and thus can only store fewer records. For instance, when the counter size is 5×5000, there are only about $(5000 * 32 - 5 * 5000 * 4)/32 = 1875$ (32 bytes are used for one entry [7]) entries available to store elephants. In our experiment we try different parameters to satisfy the memory requirement. Details are omitted here and are included in the technical report [23].

B. Comparisons for QC+S

In this section, we compare Implicit Counters using QC+S. We compare different sketches of different counter sizes: $4 \times 6000, 4 \times 7000$ since they give better results than other settings. For all the experiments below, the graphs for $\theta = 0.1\%$ have similar trends as $\theta = 0.01\%$, therefore we omit the graphs for $\theta = 0.1\%$ and focus on the smaller threshold. Details can be found in [23].

From the results in Figure 2, we see that SSS generally has smaller average relative error and probability of false alarm than other sketches, but larger probability of false negative than count-min, for most counter size settings.

The trend of the figures are due to hash collisions in the counters; the count values are estimated by using the sampled identities to query the counter. Since smaller sampling rates cause more items to be missed, the average relative error and probability of false positives improve with smaller sampling rates; the effects of missing items and overestimating count values cancel each other out.

C. Comparison for RS+S

In this section, we first compare different counter sizes for Implicit Counters to find the best counter size setting. We

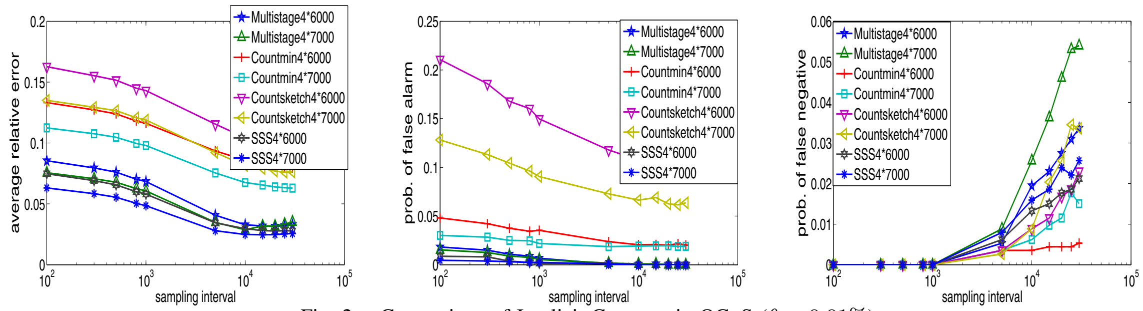


Fig. 2. Comparison of Implicit Counters in QC+S ($\theta = 0.01\%$)

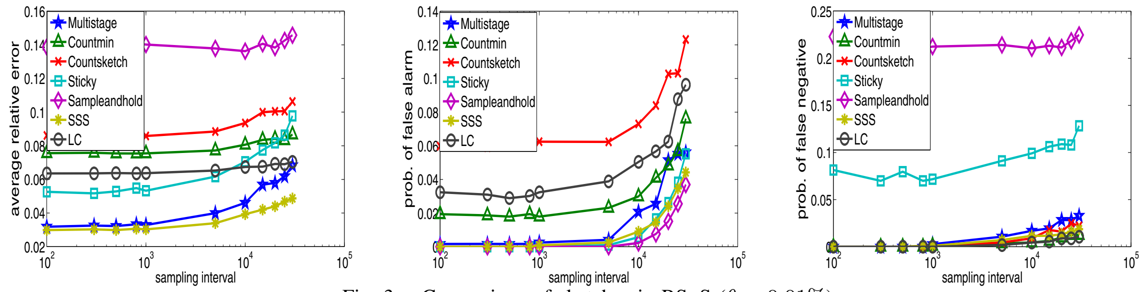


Fig. 3. Comparison of sketches in RS+S ($\theta = 0.01\%$)

found that the best counter size is 4×7000, which is very close to the comparative study in Section V-B. We use this counter size to compare these four sketches with sticky sampling, lossy counting and sample-and-hold. The comparison of different sketches are illustrated in Figure 3.

SSS still has the best results in almost all the metrics, with only a slightly larger probability of false positive than sample-and-hold. Meanwhile, since the sampled items are not used to query the counters (as in Section V-B), a larger sampling interval will always deteriorate the overall performance.

Proposition 1: Multistage is better than count-min in terms of probability of false alarms. This is because multistage uses hash table in the flow memory to keep the elephants. When a new packet comes in, it has the opportunity to directly update the flow memory, instead of increasing hash collisions in the counters [7]. A small or medium item is therefore less likely to be mis-reported as an elephant.

Proposition 2: Sticky sampling is a combination of sample-and-hold and the KSP algorithm. Sticky sampling acts stickily: an item's record will keep being updated once it is sampled. However, the sampling rate is decreasing as more packets are processed. This makes it similar with KSP, i.e, decrease count values to delete the smallest existing elephant. The deletion of smallest elephant is random, which is different from KSP. The combination of sample-and-hold and KSP makes it better than sample-and-hold.

Proposition 3: SSS is better than multistage since it is more selective in items with size around the threshold [10]. This is achieved by sampling an item before updating the counters. The sampling component helps to discard small flows that might be mis-reported as elephants, which reduces both the probability of false negative and probability of false positive.

Proposition 4: Lossy counting has large probability of false positive, due to the overestimation of the size of an item before it is inserted into the sketch.

D. Comparing QC+S and RS+S

In this section we compare QC+S and RS+S. We omit count-sketch and count-min since they are consistently worse than multistage and SSS in previous comparisons. We only present results for counter size 4×7000. Meanwhile, we also present the results for some sketches using the naive iceberg approach as well as the naive sampling approach. This helps us understand the limitations of the naive approaches. The results are illustrated in Figure 4.

The communication cost is the number of items that will be sent to the central server, which mostly depends on the sampling interval and remains similar for different sketches. For the intervals we examined, the total costs are around 25MB for sampling interval 100 and 580KB for interval 30000. The naive communication cost is roughly 64MB, in which case every local packet is dumped and every item is reported. For the naive iceberg approaches, the communication cost is irrelevant to the sampling interval, and remains constantly smaller than other combinations. The combinations generally utilize more memory to capture the split icebergs. There are a few points worth noting:

1) The naive sampling approach is accurate when the sampling interval is small. This does not conflict with [7], since we are measuring number of packets instead of number of bytes. For the latter case, the variation of individual packet size introduces more error. The inferior performance of our combination also suggests that there are large hash collisions in one measurement interval of 5 minutes. However, naive sampling is very sensitive to the sampling interval; even when it is as large as 100,

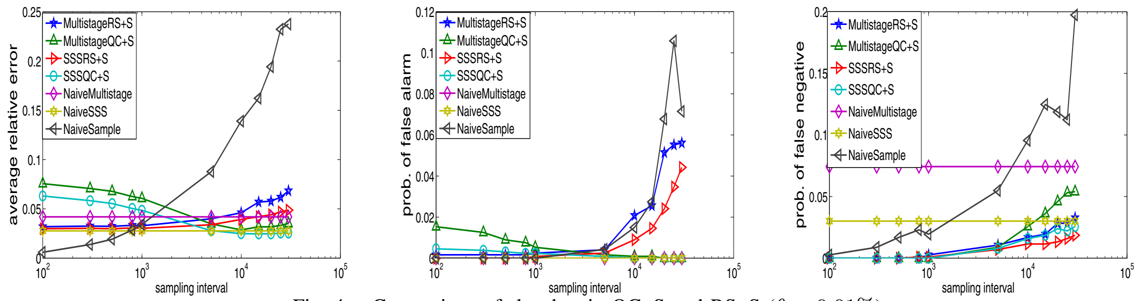


Fig. 4. Comparison of sketches in QC+S and RS+S ($\theta = 0.01\%$)

the total communication cost is still prohibitively large.

2) When sampling interval is small, RS+S is better. When the interval grows, QC+S becomes better. Note that only when the sampling interval is around 30000 can the communication cost be reduced to 18KB/2MB $\approx 1\%$ of the naive approach. QC+S is more feasible since we also aim to reduce the communication cost.
3) The naive iceberg approach has similar average relative error and probability of false alarm as RS+S and QC+S in large sampling intervals. However, it has larger probability of false negatives.

The results suggest that QC+S is the most efficient combination. For small communication cost, SSS, count-min and multistage perform better in QC+S rather than RS+S, in terms of average relative error and probability of false alarm. However, RS+S has a smaller probability of false negatives. This suggests that we can use the sampled identities to query the local counters for estimation. This can yield good performance even if the uniform sampling interval is large. The total communication cost can be reduced to $1\% \approx 0.58/64$ of the naive approach, i.e., two orders of magnitude improvement.

For the average relative error and probability of false alarm, SSS is the best. Count-min is the best in probability of false negatives. However, it is not sustainable by current SRAM since it requires large number of operations per packet.

E. Experiments on Synthetic Data

In this section we focus on the detection accuracy of one particular inserted iceberg. We also study how the detection accuracy will be influenced by different split patterns of this iceberg. In this experiment we only use combination RS+S since we want to compare all the sketches. The inserted iceberg size is $180,000$, while the iceberg threshold is about $170,000$ ($\theta = 0.01\%$). We split the inserted iceberg in different nodes ($n = 20$) according to uniform, gaussian, and zipfian distributions. Every experiment is repeated 20 times. The results are in Figure 5. Our findings are as follows.

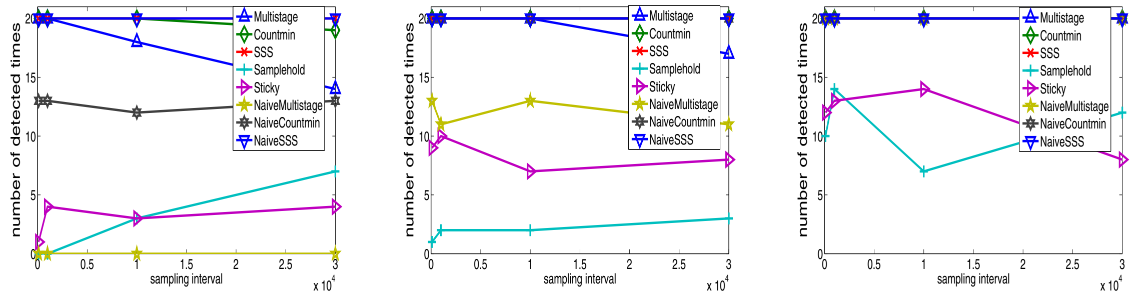
1) The gaussian and zipfian split both have better results than uniform split of the inserted iceberg. In non-uniform split, the global iceberg remains large in some monitors. In uniform split, the inserted iceberg has larger chance of being estimated by reverting sampling, which is inaccurate.
2) SSS in RS+S also performs well for capturing the inserted iceberg, however, its relative error is not the best in all the splits. For different split patterns, the best estimation algorithms are different.
3) For most cases, the naive iceberg approach is infeasible when the global iceberg is distributed finely across different streams. Although we find that naive SSS approach managed to detect the inserted iceberg every time, it might perform poorly in other split patterns. For example, the global iceberg can be slightly below the local elephant threshold at most monitors, and these count values may go undetected by naive SSS.

F. Improving Multistage

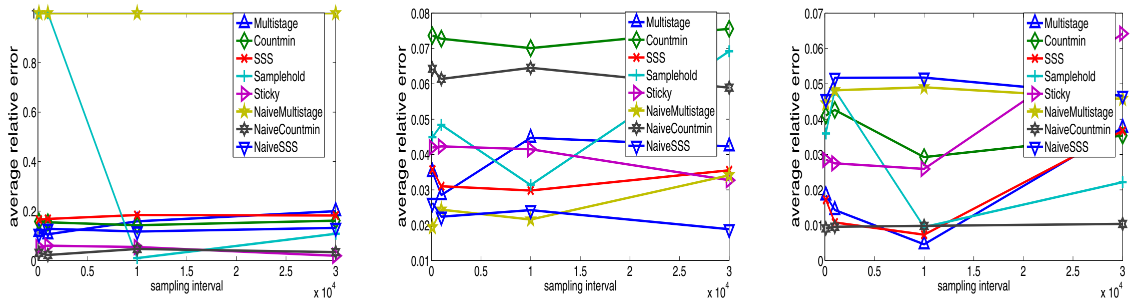
In this section we slightly modify the original multistage algorithm. In our modified multistage, once an elephant is reported from the counters to the flow memory, we decrease the counters by the estimated count value. This is different from the original multistage in [7], in which it only refreshes the counters at the beginning of each measurement interval. Intuitively, the flow inserted into the flow memory will not influence any other coming items. Note that the removal process will not influence the reported elephants, since the estimate remains the same as it is in original multistage algorithm. However, this process might underestimate other items, since it might decrease the counter by more than it should. Interestingly, we find the overall performance is better than the original multistage algorithm in every aspect.

The results of the overall performance for this modified algorithm are in Figure 6. "MMultistage" in the figures denotes our modified algorithm. In Figure 6 the modified multistage has smaller average relative error, probability of false alarm and probability of false negative than multistage for both QC+S and RS+S. It is even better than SSS for most metrics. For the synthetic data, we also find that the modified multistage algorithm can capture the inserted iceberg with more success and smaller estimation error. The results are similar with Figure 6 and are included in [23].

The modification introduces smaller probability of false positive, since the large flows have less impact on the small ones. It decreases the elephant threshold for the same memory size. For example, initially, a flow is inserted into the flow memory if its related counter size is larger than 10000; now, this value can be reduced to 8000. The earlier a flow is



(a) Number of detected times for uniform, gaussian, and zipfian split patterns



(b) Average relative error for uniform, gaussian and zipfian splits patterns

Fig. 5. Detection accuracy for one inserted iceberg in synthetic experiments

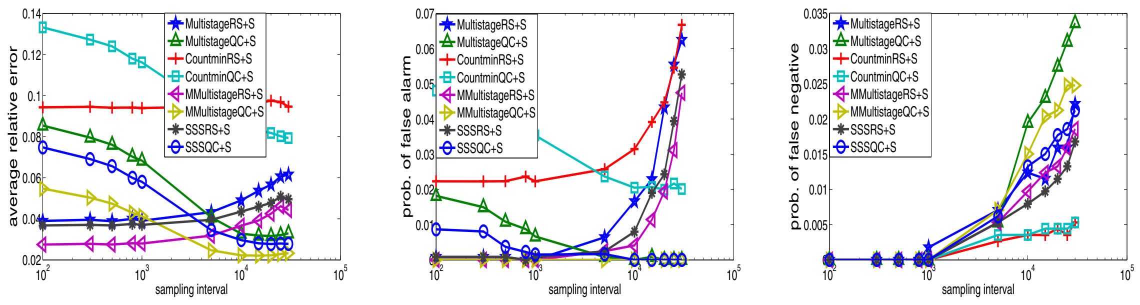


Fig. 6. Modified multistage in Combination QC+S and RS+S in real data($\theta = 0.01\%$)

inserted into flow memory, the more accurately it will be kept. To demonstrate how the local detection influences global detection, we especially look into two split pattern of global iceberg whose size is slightly above $0.01\%S$.

Case 1 (Uniform): The iceberg is split uniformly. It appears with frequency slightly above the threshold at every monitor. Since sampling can complement sketching, the overestimation by hash collision in counters and the underestimation introduced by the modified multistage have opportunities to cancel each other out.

Case 2 (Zipfian): The iceberg is split following a Zipfian distribution. It appears much larger than the threshold in some places and as small flows everywhere else. For the top local heavy-hitters, the estimation is more accurate since they are inserted into flow memory earlier. The small flows are estimated by reverting sampling; the estimation remains the same for different detection schemes.

Meanwhile, we find that this modification increases the probability of false negative when detecting local elephants [23]. Sampling does not necessarily complement sketching in local elephant detection, but does help capture the underestimated values in global iceberg detection.

VI. Discussion

Our combination of local sketches and sampling outperforms all the naive approaches for detecting global icebergs. The naive sampling approach is accurate when the sampling interval is small. However, the prohibitive communication cost precludes such a solution. The naive iceberg approach misses the distributed global icebergs which appear as non-elephants at some monitors.

We compared different sketches as well as different combinations of sampling and sketching. We conclude that SSS using QC+S with a large sampling interval is the best choice. Not only does this solution give a small probability of false alarm and average relative error, but also requires only a small number of operations per packet. Its communication cost is only 1% of the naive approach. This does not mean that SSS or QC+S are always best for all metrics. For instance, count-min often has a smaller false negative probability, with much higher estimation error and probability of false alarm. Also, RS+S is generally better than QC+S when uniform sampling interval is small.

We attain some interesting insights from the experiments on the combinations, which help understand global iceberg detection in distributed streams.

First, it is better to follow the design of multistage instead of using heaps, as in count-min or count-sketch. Since multistage can query whether an item is already contained in the sketch, it has the opportunity to directly update the count values of existing icebergs, which improves accuracy.

Second, comparing count-min and count-sketch reveals which method is better for estimating the count values. Although the estimate from count-sketch is unbiased [8], its overall performance is worse than count-min or multistage.

Finally, we presented a particular example in which global iceberg detection differs from local elephant detection in Section V-F. The slight modification of multistage can introduce underestimation, thus leading to increased probability of false negatives in detecting local elephants. However, both the overall detection accuracy and the accuracy for the small icebergs are improved for the global iceberg detection problem.

We believe our comparison study provides a first step towards global iceberg detection in distributed streams. The insights we gained will be useful in other experiments with different parameter settings.

Our solution to this problem is not yet optimal. There are some methods we need to explore. For instance, multistage [7] can be made to measure many shorter intervals, which might be more accurate and practical. Also, we did not optimize the process of reporting data sets to the central server, but only focused on combining sampling and sketching in local measurement. Results from [19] might further improve our solution. We will explore these in future work.

VII. Conclusion

In this paper we introduce and motivate the study of distributed algorithms for estimating icebergs across multiple streams. This is an important problem in the context where resources are limited at both the local collection points as well as the links connecting them to the central aggregator. This is a very useful application for problems such as worm detection, SLA measurements, and DDoS attack containment.

We studied the effect of combining several of the most widely used local heavy-hitter detection algorithms with sampling across the local points to estimate the global icebergs. We performed experiments on both real as well as synthetically constructed data to compare these different sketches and algorithms. Our experiments showed that the combination of uniform sampling with the multistage algorithm gives the best iceberg detection and estimation algorithm.

Acknowledgements

This work is supported in part by NSF grants CNS-0519745, CNS-0626979, CNS-0716423, and CT-ISG-0716831.

References

[1] P. Ayres, H. Sun, H. Chao, and W. Lau, "ALPi: A DDoS defense system for high-speed networks," *IEEE Journal on Selected Areas in Communications*, vol. 24(10), pp. 1864–1876, 2006.

[2] "Akamai technologies inc." http://www.akamai.com/.

[3] S. G. Cheetancheri, J. M. Agosta, D. H. Dash, K. N. Levitt, J. Rowe, and E. M. Schooler, "A distributed host-based worm detection system," in *Proceedings of the 2006 SIGCOMM workshop on Large-scale attack defense (LSAD)*, 2006.

[4] J. Sommers, P. Barford, N. Duffield, and A. Ron, "Accurate and efficient sla compliance monitoring," in *Proceedings of the 2007 conference on Applications, technologies, architectures, and protocols for computer communications (SIGCOMM)*, 2007.

[5] "Cisco netflow," http://www.cisco.com/warp/public/732/netflow/.

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

[7] C. Estan and G. Varghese, "New directions in traffic measurement and accounting," in *Proc. ACM SIGCOMM*, 2002.

[8] M. Charikar, K. Chen, and M. Farach-Colton, "Finding frequent items in data streams," *Theoretical Computer Science*, vol. 312(1), pp. 3–15, 2004.

[9] A. Arasu and G. S. Manku, "Approximate counts and quantiles over sliding windows," in *Proceedings of the ACM Symposium on Principles of Database Systems (PODS)*, 2004.

[10] F. Raspall, S. Sallent, and J. Yufera, "Shared-state sampling," in *IMC '06: Proceedings of the 6th ACM SIGCOMM conference on Internet measurement*. New York, NY, USA: ACM, 2006, pp. 1–14.

[11] R. M. Karp, S. Shenker, and C. H. Papadimitriou, "A simple algorithm for finding frequent elements in streams and bags," *ACM Transactional Database Systems*, vol. 28, no. 1, 2003.

[12] B. Babcock, S. Babu, M. Datar, R. Motwani, and D. Thomas, "Operator scheduling in data stream systems," *The VLDB Journal*, vol. 13, no. 4, pp. 333–353, 2004.

[13] J. Feigenbaum, S. Kannan, M. Strauss, and M. Viswanathan, "Streaming algorithms for distributed, massive data sets," in *Proc. IEEE FOCS*, 1999.

[14] M. Cherniack, H. Balakrishnan, M. Balazinska, D. Carney, U. Cetintemel, Y. Xing, and S. Zdonik, "Scalable distributed stream processing," in *Proceedings of the 2003 CIDR Conference*, 2003.

[15] G. Cormode, S. Muthukrishnan, and K. Yi, "Algorithms for distributed functional monitoring," in *Proceedings of the 19th annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, 2008.

[16] A. Manjhi, V. Shkapenyuk, K. Dhamdhere, and C. Olston, "Finding (recently) frequent items in distributed data streams," in *Proc. IEEE ICDE*, 2005.

[17] B. Babcock and C. Olston, "Distributed top-k monitoring," in *Proceedings of the ACM international conference on Management of data (SIGMOD)*, 2003.

[18] C. Olston, J. Jiang, and J. Widom, "Adaptive filters for continuous queries over distributed data streams," in *Proceedings of the ACM international conference on Management of data (SIGMOD)*, 2003.

[19] Q. Zhao, M. Ogihara, H. Wang, and J. Xu, "Finding global icebergs over distributed data sets," in *Proceedings of the ACM Symposium on Principles of Database Systems (PODS)*, 2006.

[20] E. Cohen, N. Duffield, H. Kaplan, C. Lund, and M. Thorup, "Sketching unaggregated data streams for subpopulation-size queries," in *PODS '07: Proceedings of the twenty-sixth ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems*. New York, NY, USA: ACM, 2007, pp. 253–262.

[21] G. Cormode and M. Hadjieleftheriou, "Finding frequent items in data streams," *PVLDB*, vol. 1, no. 2, pp. 1530–1541, 2008.

[22] A. Kumar and J. Xu, "Sketch guided sampling - using on-line estimates of flow size for adaptive data collection," in *Proc. IEEE INFOCOM*, 2006.

[23] G. Huang, A. Lall, C. Chuah, and J. Xu, "Uncovering global icebergs in distributed streams," *Technical Report ECE-CE-2009-1, UCDavis, 2009*.

[24] "Internet2 abilene network," http://abilene.internet2.edu/.