Fourth Colloquium on Mathematics and Computer Science DMTCS proc. **AG**, 2006, 119–128

LOGLOG *counting for the estimation of IP traffic*

Olivier Gandouet[†] and Alain Jean-Marie

INRIA and LIRMM, CNRS/University of Montpellier 2
161 rue Ada, F-34392 Montpellier
gandouet,ajm@lirmm.fr

In this paper, we discuss the problem of estimating the number of "elephants" in a stream of IP packets. First, the problem is formulated in the context of multisets. Next, we explore some of the theoretical space complexity of this problem, and it is shown that it cannot be solved with less than $\Omega(n)$ units of memory in general, n being the number of different elements in the multiset. Finally, we describe an algorithm, based on Durand & Flajolet's LOGLOG algorithm coupled with a thinning of the packet stream, which returns an estimator of the number of elephants using a small amount of memory. This algorithm allows a good estimation for particular families of random multiset. The mean and variance of this estimator are computed. The algorithm is then tested on synthetic data.

Keywords: Probabilistic counting, Communication Complexity, IP traffic

1 Introduction

The initial motivation for this paper is the advance of *traffic engineering* in data networks. Modern architectures and protocols are being standardized, which aims at guaranteeing the quality of service delivered to users. The proper functioning of these protocols requires an increasingly detailed knowledge of statistical characteristics of the stream of data packets (usually, IP packets) passing through a given point of the network. On the other hand, the amount of information flowing through the network also increases, and the challenge is to obtain in real time, with little computational time and small memory consumption, information from a huge set of data packets.

Among the characteristics attached to a data flow, the question of *mice and elephants* has recently attracted some attention: see for instance [2]. Roughly speaking, elephants are flows of information which stay active for a long time in the network, while mice are the flows with a small lifetime. Since the dynamical response of the network has been found to be different for both categories, it is interesting to have an idea of the amount of flows of each type. This paper is devoted to algorithms which estimate the number of elephants present in a flow of packets (with a precise definition of what exactly is an elephant), using an amount of memory which is small relative to the total amount of data.

Two types of results are presented. First, we explore the theoretical limitations of the problem. Using results from the theory of *communication complexity*, we show that no algorithm can solve the problem in general with less than $\Omega(n)$ units of memory, where n is the number of distinct flows, elephants and mice together. Some related work has recently been done in [7], on the estimation of the entropy of a multiset. This paper also features a simple estimator of the number of mice and elephants; our present proposal is much more precise and uses less memory. We also improve a result of [8] about the closely connected problem of knowing if a given Internet user scans more than a certain number of communication "ports" on a target machine.

Next, we restrict our attention to a specific class of data flows, for which we exhibit an algorithm that produces a statistical estimator for the number of elephants, and uses $O(\log\log\ n)$ units of memory. This algorithm is based on Durand and Flajolet's LOGLOG algorithm [3]. We use the properties of LOGLOG to derive the mean and variance of our estimator. Finally, we test the estimator and its parameterization on synthetic data, and show that the results are satisfactory.

The paper is organized as follows. The theoretical memory complexity of the problem is analyzed in Section 3. Our algorithm and its analysis are described in Section 4. The experiments are reported in Section 5. We conclude in Section 6.

[†]This research is funded by the ACI "Masses of Data" program of the French Ministry of Research (Contract # 04 5 58).

2 Preliminaries

A *multiset* is a structure similar to a set, but where repetitions of the same element are allowed. The *cardinal* of the multiset $\mathcal{M}$, denoted by $\#\mathcal{M}$, is the number of elements it contains, including repetitions. The set composed by the different elements of a multiset $\mathcal{M}$ is named the *support* of the multiset and denoted by $Supp(\mathcal{M})$. Union and intersections are naturally defined on multisets. A *representative sequence* of a multiset is a sequence with length equal to the cardinal of the multiset, which is such that each element of the multiset appears once in the sequence.

Example 1 *Let $\mathcal{M}$ be the multiset {0,0,7,7,4,7}, which is also equal to {0,0,4,7,7,7} (and any other permutation). Then the support $Supp(\mathcal{M})$ of this multiset is $\{0, 4, 7\}$. The cardinal of $\mathcal{M}$ is 6, whereas the cardinal of $Supp(\mathcal{M})$ is 3. The sequences $u = (7, 7, 0, 0, 7, 4)$ and $v = (7, 7, 7, 0, 4, 0)$ are two different representative sequences of $\mathcal{M}$.*

For practical reasons, we introduce here a more formal (but equivalent) definition for multisets. Let E be a finite set; a multiset with support E can be seen as a function $\mathcal{M} : E \longrightarrow \mathbb{N}^*$. With this notation, we have for the cardinal: $\#\mathcal{M} := \sum_{x \in Supp(\mathcal{M})} \mathcal{M}(x)$. This is not to be confused with the cardinal of the support of $\mathcal{M}$ which is $\#E$. In the example above, the equalities $\mathcal{M}(0) = 2$, $\mathcal{M}(4) = 1$ and $\mathcal{M}(7) = 3$ characterize the multiset $\mathcal{M}$.

2.1 The LogLog algorithm

We shall make use of the LogLog algorithm, proposed by Durand and Flajolet [3]. LogLog was initially designed for approximate counting in multisets. It produces an estimator for the cardinal of the support of a multiset, given any of its representative sequences. The principal statistical properties of this estimator are recalled in Theorem 1. We summarize LogLog in Algorithm 1, where the function $s(x)$ returns the position of the first bit equal to 1 in the binary expansion of x. The sequence α_h is known and computable.

Algorithm 1 LogLogAlgorithm

Let m,k,h,n,d be integer numbers, with $h := 2^m$.
Let $(x_i)_{i \in [1,d]}$ be a representative sequence having cardinal of support n such that each $x_i \in \{0, 1\}^\infty$.
Let j be an integer and M an array integer of size 2^k. Initialize j to 1 and each element of M to 0.
while $j \leq d$ **do**
  $x_j = (x_{j,1}, x_{j,2}, ...)$
  $l := (x_{j,1}, x_{j,2}, ..., x_{j,k})$ the k first bits of the binary expansion of x_i
  $M[l] := \max(M[l], s(x_{j,k+1}, x_{j,k+2}, ...))$
  $j \leftarrow j + 1$
end while
RETURN $E_n := \alpha_h h 2^{\sum_{i=1}^{h} M[i]/h}$

Let n be the size of the support of the multiset corresponding to the representative sequence. LogLog uses $O(\log\log\ n)$ memory bits. It turns out that the distribution of the (random) value returned by the LogLog algorithm depends only on the cardinal of the support of the multiset, and not on the nature of its elements, nor on the particular representative sequence. Let therefore E_n be the number returned by the LogLog algorithm, applied on a multiset with support of cardinal n.

Theorem 1 (Durand and Flajolet [3]) *The mean and variance of E_n are given by:*

$$\begin{aligned} \mathbb{E}(E_n) &= n + n\theta_{1,n} + o_m(n)\,, \\ \sqrt{\mathbb{V}(E_n)} &= \frac{n\beta_m}{\sqrt{m}} + n\theta_{2,n} + o_m(n)\,, \end{aligned}$$

where $\theta_{1,n}$ and $\theta_{2,n}$ are bounded by 10^{-6} for all n, $o_m(n)$ is $o(n)$ for any m, and where β_m denotes a known, computable sequence, bounded by 2.

Our idea is to use the LogLog algorithm to estimate the number of elements being repeated "many times", because we have in mind that our algorithm will inherit the properties of LogLog: low memory consumption, independence of the representative sequence presented as data. This second property is desirable in the sense that in real streams of packets, the different flows are mixed randomly in a way that may be difficult to capture statistically. It is therefore a good thing that the estimator we propose does not depend on the exact way this mixing occurs.

2.2 Elephants and Multisets

In our analysis, we shall consider that elephants are IP flows that contain more than a certain number λ of packets. The remaining flows will be termed as mice. We do not consider other categories of IP flows, nor do we take time or packet dispersion into account. Indeed, the estimator we build is independent of the order in which packets arrive to the counter. While introducing more flow types is a rather straightforward extension of the present work, the temporal aspect of flows will be the topic of future research.

In terms of multisets, we partition the support E of the multiset $\mathcal{M}$ as: $E = E_e \cup E_m$ with

$$E_e \;=\; \{x \in E \mid \mathcal{M}(x) \geq \lambda\}\,, \qquad E_m \;=\; \{x \in E \mid \mathcal{M}(x) < \lambda\}\,,$$

for some $\lambda \in \mathbb{N}^*$. The problem we are interested in (the "Elephant problem") is to estimate the number of elephants:

$$N_{\lambda,\mathcal{M}} \;:=\; \#E_e\,.$$

3 Space Complexity Bounds related to the Elephant Problem

In this section, we explore some theoretical limits of the problem. We are particularly interested in the amount of memory necessary to an algorithm which estimates *in one pass* the number of elephants in a data flow. Among all possible algorithms for solving the problem of counting elephants, we are particularly interested in those which a) work in a single pass, b) are allowed to use probabilities. We will model algorithms as functions of their input and internal memory. Accordingly, we define *online* algorithms as functions $A(\cdot)$ having the property that $A(X^{(0)}; b_1, ..., b_m) = A(X^{(k)}; b_{k+1}, ..., b_m)$, where $\{b_1, b_2, \ldots\}$ is a binary input sequence, and $X^{(i)}$ is the contents of the memory of A just before reading the $(i+1)$st bit. A *probabilistic* algorithm will be such that $A(\cdot)$ also depends on a sequence of *internal* random variables $\{\omega_1, \omega_2, ...\}$.

We introduce the notion of online algorithms because it includes the class of one-pass algorithm, and because their definition suits to the proofs of our results. Algorithms that can be used in traffic monitoring must work in one pass because it is usually not desirable, or simply not feasible, to store the whole data. We deal with both deterministic algorithms and randomized algorithms. Most of our attention has been put on randomized ones because our hope was to obtain lower memory complexity bounds by relaxing the accuracy of the result. We obtain below the result that the computation of the number of elephants in a multiset with n different flows has a space complexity which is $\Omega(n)$ in general, for deterministic as well as for probabilistic algorithms. In that sense, our hope has been too optimistic.

3.1 Communication complexity issue

In order to obtain some *lower bounds* on memory complexity for the problem we have introduced previously, we use results from communication complexity theory (see [6] and [5, 9, 10]). The idea of using communication complexity was already used in [1] for the computation in one pass of the frequency moments of multiset. The paper [8] also applies this methodology to the identification of properties of Internet flows. A recent application, also connected to the problem of Elephants, appears in [7]. The present introduction to this theory will be limited here to the basic concepts and examples, and to results we shall need for our purpose.

Communication complexity theory is related to a game where k players try to compute cooperatively a binary function. Each player knows a part of entry of the function that the other players do not know. Furthermore, they each possess an infinite calculation potential and a perfect knowledge of the function. They can send messages to each other under a previously defined protocol, until one of the player has a sufficient knowledge about the entry of other players to compute the value of the function. Finding a protocol of discussion between players that allows one player to compute the function, while sending a minimum number of messages (or of bits) is the goal of the game. Accordingly, the *communication complexity* of a function f is defined as the minimum total number of bits that players must exchange in order to compute the function under the best protocol.

Depending on the application context, several families of admissible protocols have been defined. The one which is the closest to our problem is the "one-way communication complexity". For these protocols, the ith player can only transmit information to the $(i+1)$st player. The number of bits exchanged in the protocol is then simply the sum of the size of the messages sent from i to $i+1$, for i varying from 1 to k. The one-way communication complexity of a function f is noted $C^{\rightarrow}_f$.

We now illustrate the notion of communication complexity through classical examples.

Example 2 *Let $(x, y) \in \{0, 1, ..., n\}^2$. Consider the following function f: $f(x, y) := (x + y) \mod 2$, to be computed by $k = 2$ players. The first player knows x and the second one knows y. Consider the protocol is: 1) the first player sends to the second player $x \mod 2$; 2) the second player can then compute exactly f. One bit is transmitted, and it is impossible to do better than that.*

Example 3 *Let g be the following function of $k = 2$ arguments, defined for $x, y \in (0, 1)^n$: $g(x, y) := \mathbb{I}_{x.y>0}(x, y)$, where $x.y$ denotes the standard scalar product. Less formally, if the set coded by x and y share at least one element, then $g(x, y) = 1$. In the other case, $g(x, y) = 0$. This function g is called "Dis_n" because it is computing the answer to the question: "are the sets coded by x and y disjoint?".*

For this function, it has been proved in [5] that the complexity is $\Omega(n)$. Since n is precisely the number of bits in x, this means that asymptotically, the best protocol for g is not better than the one by which the first player just sends x to the second player.

We will also consider the randomized version of the communication complexity called tritely "randomized communication complexity". In that variant, a player is allowed to return a result when the probability that his result is correct is more that $1 - \varepsilon$ for a certain $\varepsilon > 0$. The one-way randomized communication complexity of a function f is noted $C^{\rightarrow}_{f,\varepsilon}$. The probability space used for defining the probability of correctness is based on the internal random variables of the protocol. This implies that when the algorithm is run twice with the same entry, it may return two different results. But the probability of getting the true result is at least $1 - \varepsilon$. Let us consider a concrete case.

Example 4 *Let h be the binary function defined on $\{1, ..., 2^n\} \times \{1, ..., 2^n\}$ such that $h(x, y) = 1$ if $x = y$ and 0 otherwise. This function has a one-way communication complexity $C^{\rightarrow}_h = \Omega(n)$. On the other hand, it has a $C^{\rightarrow}_{h,\varepsilon} = \Omega(\log\ n)$ one-way randomized communication complexity.*

We see through this example that using a randomized protocol can result in a lower space bound. This is not always the case since it is possible to exhibit functions for which allowing a certain degree of error does not change the memory complexity bound in asymptotic terms. For example, let reconsider the function Dis_n (Example 3). For a tolerance error less than $1/8$ ($\varepsilon < 1/8$) the one-way randomized communication complexity is $\Omega(n)$, as in the non-randomized case (see [5] for a proof).

We will see that a similar phenomenon occurs for our problem. In order to prove it, we shall make use of a function which we introduce now. Let n be an integer, $X = \{0, 1\}^n$ and $Y = \{1, 2, ..., n\}$. Define the function $D : X \times Y \mapsto \{0, 1\}$ as: $D(x, y) := x_y$. This function computes the projection of the binary vector x on its yth dimension. We prove in Appendix A the complexity result for D:

Theorem 2 *For any $\varepsilon < 1/8$, the one-way randomized communication complexity of D is $\Omega(n)$.*

3.2 Space Bounds for the Elephant Problem

As previously said, we present here memory complexity bounds related to our problem, both for the deterministic and probabilistic case. The central link between *online* algorithms and communication complexity is the following: the internal memory of online algorithms plays the part of messages sent in a communication complexity protocol. It is therefore possible to use space complexity bounds for some function to deduce space bounds for online algorithms, and as a consequence for one pass algorithms.

We obtain a first theorem about the exact computation of the number of elephants with an online algorithm. Observe that [8] obtains bounds for the problem of counting flows which "scan" more than λ ports on some machine. These bounds can be used to the problem of Elephants, but would yield a lower memory complexity bound of $\Omega(n/\lambda^5)$. Our results are stronger, in that they do not depend on λ.

Theorem 3 *Let $\lambda \in \mathbb{N}$ with $\lambda \geq 2$, and let $\mathcal{M}$ be a multiset with support included in $\{0, 1, \ldots, n\}$ and with cardinal m at least $n + 2\lambda$. Consider an online algorithm A which, for every representative sequence u of the multiset $\mathcal{M}$, returns the value $N_{\lambda,\mathcal{M}}$. Then A needs $\Omega(n)$ memory bits.*

Before proving this result, it shall prove convenient to introduce an auxiliary construction. Let $x \in \{0, 1\}^n$. Define: $m_1 := \sum_{i=1}^{n} x_i$. For every x different of the null vector (equivalently: for which $m_1 > 0$), let $u(x) := (u_k)_{k\in[1,m_1]}$ be the sequence of indices k where x_k equals 1. Next, for each $y \in \{1, 2, \ldots, n\}$ and $m_1 \in \mathbb{N}^*$, let $v(y, m_1) = (v_k)_{k\in[1,m-m_1]}$ by the sequence formed of $\lambda-1$ times the element y followed by $(m - m_1 - \lambda + 1)$ times the value 0. Finally, let $w(x, y) = (w_k)_{k\in[1,m]}$ be the concatenation of $u(x)$ and $v(y, m_1)$.

Example 5 *If $n = 5$, $\lambda = 4$, $m = 11$, $x = (0, 1, 1, 0, 1)$ and $y = 3$ then: $m_1 = 3$, $u(x) = (2, 3, 5)$, $v(y, 3) = (3, 3, 3, 0, 0, 0, 0, 0)$ and $w(x, y) = (2, 3, 5, 3, 3, 3, 0, 0, 0, 0, 0)$.*

We have the property:

Lemma 1 *Let $w(x,y)$ be a sequence constructed as above, for some $m \geq 2\lambda + n$. Consider it as a representative sequence of a multiset $\mathcal{M}(w)$. Then $N_{\lambda,\mathcal{M}(w)} = 1$ if and only if $D(x,y) = 0$ and $N_{\lambda,\mathcal{M}(w)} = 2$ if and only if $D(x,y) = 1$.*

Proof: Observe that the element 0 is repeated more than λ times in w because $m - m_1 - \lambda + 1 \geq \lambda$ since, by assumption, $m \geq 2\lambda + n$ and $m_1 \leq n$. If $N_{\lambda,\mathcal{M}(w)} = 2$, it must be that some element other than 0 is repeated λ times. The elements of $u(x)$ being all distinct, this repeated element can only be y, and it is repeated λ times if and only if one of the u_k is equal to y. In other words, if and only if $x_y = 1$ or $D(x,y) = 1$. Conversely, if $D(x,y) = 0$, y is repeated only $\lambda - 1$ times, and only the element 0 is repeated more than λ times; in that case, $N_{\lambda,\mathcal{M}(w)} = 1$. □

We can not return to the **Proof of Theorem 3:** The proof is by contradiction.

Suppose that there exists an online algorithm A which returns $N_{\lambda,\mathcal{M}}$ ($\lambda \geq 2$) when given a representative sequence of a multiset $\mathcal{M}$ with cardinal equal to m, and with support having less than $n+1$ elements. Suppose further that the memory of this algorithm at step k, $X^{(k)}$, is $o(n)$ for each k.

We will exhibit a communication protocol A' with $k = 2$ parties (P_1 and P_2), that allows to compute $D(x,y)$ using less than $o(n)$ memory bits. This is a contradiction with Theorem 2 applied with $\varepsilon = 0$.

The protocol A' is the following. 1) party P_1 converts his vector x into the sequence $u(x)$; 2) P_1 runs algorithm A with $u(x)$ as parameter; 3) P_1 communicates m_1 and $X^{(m_1)}$ to P_2; 4) P_2 creates the sequence $v(y, m_1)$; 5) P_2 runs algorithm A with initial memory $X^{(m_1)}$ and with parameter $v(y, m_1)$; 6) P_2 returns 1 if the result is equal to 2, 0 otherwise.

Since A is online, we have: $A(X^{(0)}; b_1, ..., b_m) = A(X^{(m_1)}; b_{m_1+1}, ..., b_m)$, and therefore the result computed by P_2 at step 5) is indeed $N_{\lambda,\mathcal{M}(w)}$. According to Lemma 1, step 6) returns $D(x,y)$.

The amount of information passed from P_1 to P_2 in step 3) is $\log\ n$ (the maximal size of m_1) plus the number of bits in $X^{(m_1)}$, which is $o(n)$ by assumption. Consequently if A exists, the protocol A' allows to compute $D(x,y)$ using less than $o(n)$ communication bits. This concludes the proof. □

We now proceed with a similar result for probabilistic algorithms.

Theorem 4 *Let $\varepsilon \leq 1/8$, let $\lambda \in \mathbb{N}$ with $\lambda \geq 2$, and $\delta \in [0, 1/3)$. Let further $\mathcal{M}$ be a multiset with support included in $\{1, 2, \ldots, n\}$ and with cardinal m at least $2\lambda + n$. Consider an online probabilistic algorithm A which, to every representative sequence u of the multiset $\mathcal{M}$, associates an estimator $\widetilde{N}_{\lambda,\mathcal{M}}$ such that:*

$$\mathbb{P}(|\widetilde{N}_{\lambda,\mathcal{M}} - N_{\lambda,\mathcal{M}}| \ \leq\ \delta N_{\lambda,\mathcal{M}}) \ \geq\ 1 - \varepsilon\,. \tag{1}$$

Then A needs $\Omega(n)$ memory bits.

Proof: This proof has the same scheme as the proof of Theorem 3. It starts with an algorithm A which uses $o(n)$ memory bits, and returns an estimator $\widetilde{N}_{\lambda,\mathcal{M}}$ with the property (1). From this online algorithm, we construct a protocol A' as in the said proof, with a modified step 6): protocol A' returns 1 if the result of A belongs to the open interval $(\frac{4}{3}, \frac{8}{3})$ and 0 otherwise.

The probability that A' returns the correct answer for $D(x,y)$ is larger than $1 - \varepsilon$. This can be seen by considering the two possible values for $N_{\lambda,\mathcal{M}}$: 1 or 2. If $N_{\lambda,\mathcal{M}} = 1$ then with probability less than ε the result of A belongs to $(\frac{4}{3}, \frac{8}{3})$ because:

$$\mathbb{P}(|\widetilde{N}_{\lambda,\mathcal{M}} - 1| \ >\ \delta) \ =\ \mathbb{P}(1 - \delta < \widetilde{N}_{\lambda,\mathcal{M}} < 1 + \delta) \ <\ \varepsilon\,,$$

and because $\delta < 1/3$. If the true result is $N_{\lambda,\mathcal{M}} = 2$, then with probability less than ε the result of A does not belong to $(\frac{4}{3}, \frac{8}{3})$ because:

$$\mathbb{P}(|\widetilde{N}_{\lambda,\mathcal{M}} - 2| \ >\ 2\delta) \ =\ \mathbb{P}(2 - 2\delta < \widetilde{N}_{\lambda,\mathcal{M}} < 2 + 2\delta) \ <\ \varepsilon\,.$$

So in any case the probability of getting wrong is less than ε. Finally: the number of bits communicated from P_1 to P_2 equals the number of bits in $X^{(m_1)} = o(n)$ plus $O(log\ n)$. Consequently if algorithm A exists, the protocol A' allows to compute $D(x,y)$ using less than $o(n)$ communication bits with probability error less than ε. This is impossible according to Theorem 2. □

4 The B-LogLog-EC Algorithm

In the previous section we have seen some theoretical memory bounds about the problem to compute exactly, or estimate statistically, the numbers of elephants. These bounds in $\Omega(n)$ are not satisfying because for applications we have in mind, n can be very large: typically 10^9 and more. This motivates us to investigate further the problem with the following starting idea: by restricting the attention to particular (yet reasonably practical) classes of multisets, may it be possible to devise an algorithm that allows a good estimation of the number of elephants with a small amount of memory?

In this section, we describe an algorithm, based on Algorithm 1, which computes an estimator for $N_{\lambda,\mathcal{M}}$. We call it the B-LogLog-EC algorithm. We then analyze the performance of this algorithm, under the specific assumption that the multiset $\mathcal{M}$ contains exactly two types of elements: mice, which have a multiplicity of exactly α, and elephants, which have a multiplicity of exactly $\lambda\alpha$ (we will see later that the algorithm still works with more general multisets). Here, $(\alpha, \lambda) \in \mathbb{N}^* \times \mathbb{N}^*$. Formally, $\mathcal{M}$ is restricted to the set of multisets (in the notation of Section 2): $\{\mathcal{F} : S \to \mathbb{N}^* \mid \forall s \in S, \mathcal{F}(s) = \alpha \text{ or } \mathcal{F}(s) = \lambda\alpha\}$. Accordingly, $E_m = \mathcal{M}^{-1}(\alpha)$, and $E_e = \mathcal{M}^{-1}(\lambda\alpha)$. We shall denote $n_m = \#E_m$ and $n_e = \#E_e$, and $n = n_m + n_e$. The analysis will emphasize the importance of the *density* of elephants, defined as $\mu = n_e/n$. Observe that even in this simplified case the theoretical space bound remains in $\Omega(n)$. Indeed, in the proof of Theorem 3, the multisets that were used have only two kinds of elements, those which have one occurence and the ones which have more than λ occurences. So we cannot expect even in this case a lower space bound better than $\Omega(n)$. Since the B-LogLog-EC algorithm uses $o(n)$ memory bits, it cannot have strong properties such as (1). Yet, it turns out that for multisets with reasonable proportions of mice/elephant, it allows an accurate estimation of the number of elephants. This proportion appears to play a crucial role in the difficulty of the problem.

In addition, the algorithm B-LogLog-EC enjoys an interesting property inherited from the LogLog algorithm: it returns the same result independently of the representative sequence chosen.

4.1 The Algorithm

The principle of the algorithm is the following. The original stream is analyzed using the LogLog algorithm. This results in an estimator E_n of the total number of flows n, elephants and mice together. In parallel, a sub-stream is selected according to a Bernoulli sampling with some probability $1 - p$. In other words, a packet is *deleted* from the original stream with probability p, independently of previous choices. This second stream is also analyzed using the LogLog algorithm. This results in an estimator $\widetilde{E}_n$ of the total number of surviving flows. Since the probability of surviving the sampling is larger for an elephant than for a mouse, the proportion elephants/mice should increase in the sampled stream. Accordingly, it should be possible to combine the two estimators to recover the estimated values of n_e and n_m.

More precisely, let n'_m and n'_e be the number of mice and elephants in the sub-stream resulting from the Bernoulli sampling of the original stream. The probabilities that an elephant or a mouse "passes" the sampling are $g := 1 - p^{\lambda\alpha}$ and $f := 1 - p^{\alpha}$, respectively. Consequently, the average number of flows in the sampled stream is $\mathbb{E}(n') = fn_m + gn_e$, while the original number is $n = n_m + n_e$. A linear combination of (unbiased) estimators for n and n' should give access to n_m and n_e. This principle results in Algorithm 2.

Algorithm 2 Algorithm B-LogLog-EC

Let m,k,h,n,d be integers, $h := 2^m$, p a real in $(0, 1)$.
Let $B_{p,j}$ be a family of i.i.d. Bernoulli variables of parameter p
Let $(x_i)_{i\in[1,d]}$ be a representative sequence having cardinal of support n such that each $x_i \in \{0, 1\}^\infty$.
Let M and T be tables of j integers, initialized to 0. Set j to 1.
while $j \leq d$ **do**
  $x_j = (x_{j,1}, x_{j,2}, ...)$ is the binary expansion of x_j
  $l := (x_{j,1}, x_{j,2}, ..., x_{j,k})$
  if $B_{j,p} = 0$ **then**
    $T[l] := \max(T[l], s(x_{j,k+1}, x_{j,k+2}, ...))$
  end if
  $M[l] := \max(M[l], s(x_{j,k+1}, x_{j,k+2}, ...))$
  $j \leftarrow j + 1$
end while
RETURN $E_{n,e} := \alpha_h h \frac{1}{p^\alpha - p^{\lambda\alpha}} \left(2^{\sum_{i=1}^h T[i]/h} - 2^{\sum_{i=1}^h M[i]/h}\right)$

4.2 Analysis of the algorithm

Denote by $\widetilde{E}_{n,e}$ the random variable returned by the algorithm B-LOGLOG-EC. The following results provides the asymptotic mean and variance of this estimator. See [4] for detailed proofs.

Theorem 5 *The mean of $\widetilde{E}_{n,e}$ is:*

$$\mathbb{E}(\widetilde{E}_{n,e}) \;=\; n_e + \theta''_{1,n}\frac{n}{g-f} + \frac{1}{g-f}\,o_m(n)\;,$$

where $\theta''_{1,n}$ is bounded above by 2.10^{-6}, uniformly in n, and $o_m(n)$ is $o(n)$ for all m .

Proof: We have:

$$\begin{aligned}\mathbb{E}(\widetilde{E}_n - fE_n) &= fn_s + gn_e + \theta'_{1,n}n + o(n) - fn_e - fn_s - f\theta_{1,n}n \\ &= (g-f)n_e + n(\theta'_{1,n} - f\theta_{1,n}) + o(n) - fo(n)\;.\end{aligned}$$

Dividing by $g-f$, we get:

$$\mathbb{E}(\widetilde{E}_{n,e}) = \mathbb{E}\left(\frac{\widetilde{E}_n - fE_n}{g-f}\right) = n_e + \theta''_{1,n}\frac{n}{g-f} + o(n)\;,$$

where $\theta''_{1,n}$ is bounded by $(2-p^\alpha)10^{-6}$, and therefore by 2.10^{-6}, for all $n \in \mathbb{N}$. □

Theorem 6 *The variance of $\widetilde{E}_{n,e}$ is:*

$$\mathbb{V}(\widetilde{E}_{n,e}) = \frac{1}{m}\left(\frac{\beta_m n_e}{(g-f)}\right)^2\left(\left(\frac{1-\mu}{\mu}f+g\right)^2 + \frac{f^2}{\mu^2}\right) + \frac{n_e^2\widetilde{\Theta}_{1,n}}{(g-f)^2} + \frac{n_e^2(1-f)^2\widetilde{\Theta}_{2,n}}{\mu^2(g-f)^2} + \frac{o(n_e^2)}{(g-f)^2}\;, \quad (2)$$

where for all $i \in \{1,2\}$ $\widetilde{\Theta}_{i,n}$ is bounded above by 10^{-6} uniformly in n, and β_m is the value in Theorem 1.

The result above can be extended to a more general setting, where mice are assumed to have a random number X of elements, and elephants are assumed to have a random number Y of elements. The formulas above apply, with $f := 1-\mathbb{E}(p^X)$ and $g := 1-\mathbb{E}(p^Y)$. In particular, the order of magnitude of the variance remains the same. Also, the right-hand side in (2) becomes an upper bound if the distribution of the size of an elephant has support $[\lambda\alpha..\infty)$ instead of being concentrated on $\lambda\alpha$. Such an extension could be useful for the classification of flows described by an *a priori* size distribution (in addition to other features), which would be for instance determined by off-line statistics.

The asymptotic expansion in (2) reveals several facts. First, it is possible to reduce the variance at the expense of more memory, by increasing m, just as in the LOGLOG algorithm. On the other hand, the standard deviation is roughly inversely proportional to the elephant density factor μ. The effect of μ can be compensated by increasing m in the first term of the expansion, but not in the second. However, since the term $\Theta_{1,n}$ is of very small magnitude, this second term can become important only for very small values of μ. We have performed some work (not reported here) on the "size" of the value p, and we conclude that defining $p = (1/\lambda)^{1/(\lambda-1)\alpha}$ guarantees that our estimator has good asymptotic properties when λ is large. We have used this setting in the experiments reported below.

Another remark can be made regarding the approximation of the standard deviation: we see that when λ goes to infinity we have nothing else that the same standard deviation of a LOGLOG running only on elephants. This is quite reasonable because when λ become large, elephants possess much more elements than mice and the multiset can easily be filtered in order to make mice almost disappear.

When the size of elephants and mice is random, and not fixed anymore, the same dimensioning work can be done. The estimation is "robust" in the sense that if the laws of the respective random variables are relatively concentrated around their mean, choosing p as if they were deterministic remains a good choice. Preliminary experiments validating this assertion have been performed, and will be reported elsewhere.

5 Experiments

In this section we present some simulations we have made in order to have a practical idea of results obtained by B-LOGLOG-EC on synthetic data.

Experimental setup. We have made several experiments in order to test the theoretical predictions for the mean and variance of the estimator. Only results about the variance are reported here. Result on the mean are as expected. A first concern is about the influence of second order terms on the variance of the estimator. A second question is on the way the variance is decreasing with respect to the λ parameter (the size ratio between elephants and mice). Also we want to see the impact of the choice of the parameter p.

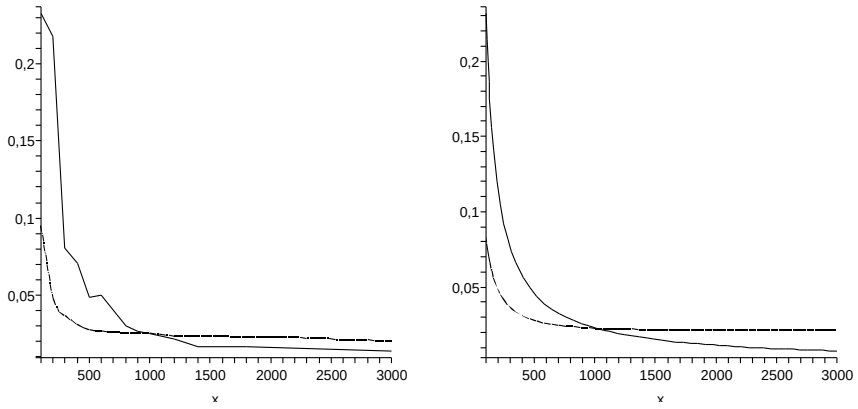

Fig. 1: Left-hand side: Empirical variance coefficient Right: Theoretical variance coefficient

Figure 1 represents the coefficient of variation $\mathbb{V}(\widetilde{E}_{n,e})/\mathbb{E}(\widetilde{E}_{n,e})^2$, as a function of λ. The value was empirically collected (left-hand side) or theoretically obtained using the main terms in (2) (right-hand side). On both plots, the dashed curve is obtained for $p = 0.99$ and the continuous one for $p = p_\lambda := (1/\lambda)^{1/(\lambda-1)\alpha}$. For all experiments, we choose $m = 2^{16}$, and $\mu = 1/1000$. The figure shows that the coefficient of variation is generally small, and gets smaller as λ grows larger. Furthermore, we see that taking p constant gives a better result than with $p = p_\lambda$ for small λ values; when λ is larger than 1000, using $p = p_\lambda$ gives better results. This can be explained by the fact that the objective of our heuristic is asymptotic, and it is indeed expected to give better results when λ is large. Secondly, comparing both empirical and theoretical variances (the latter being restricted to its leading factor), we observe that the influence of the "Θ" and "$o(n_e)$" terms appears to be small in the effective value of the variances. This is especially the case when $p = p_\lambda$.

To conclude this analysis, we report findings on the shape of the distribution of the estimator E_{n_e}. We have compared in Figure 2 the distribution of a version of E_{n_e} centered and reduced with its empirical mean and variance, with a standard Normal distribution. The parameters used were: $n_e = 10^5$ elephants, $n_s = 10^8$ mice, $m = 2^{15}$, $\lambda = 1000$ and $\alpha = 3$. One thousand i.i.d. copies of the random variable have been averaged to obtain the figure. The curves are almost indistinguishable, which lead us to believe that the law of our estimator weakly converges to a Normal distribution when m goes to infinity. Proving this conjecture does not appear to be easy, since the random variables involved are not independent. Assuming the result true, this suggests the following heuristic: for m sufficiently large, $P(E_{n_e} \in [0.7n_e, 1.3n_e]) \geq 0.99$. The estimate thus obtained can be sufficient for practical applications.

6 Conclusion

As a conclusion, B-LOGLOG-EC seems to work well when we have a good knowledge of the size distribution of mice and elephants. It is our current goal to relax this restriction using an estimation of the law of mice and elephants, even if we know that such results can not be obtained using less than linear memory, according to Theorem 4. It is however possible that with multisets which are not completely general, but are typical of the Internet, this algorithm gives good statistics for the number of elephants. Finally, observe that what we have done here for two types of flows can be quite easily extended to more types. We expect that the results will be similar.

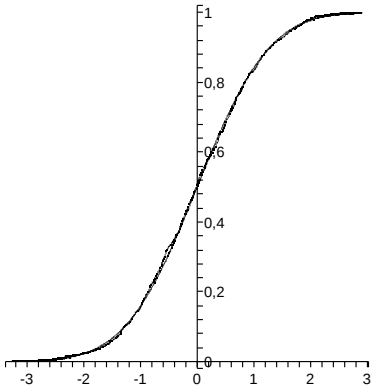

Fig. 2: Repartition function of a $\mathcal{N}(0,1)$ and normalized empirical distribution of E_{n_e}.

Acknowledgements

The authors would like to thanks particularly the reviewers for their extremely constructive comments.

References

[1] Noga Alon, Yossi Matias, and Mario Szegedy. The space complexity of approximating the frequency moments. In *28th ACM Symp. on Theory of Computing*, pages 20–29, 1996.

[2] N. Ben Azzouna, C. Fricker, and F. Guillemin. Modeling ADSL traffic on an IP backbone link. *Annals of Telecommunications*, 59(11-12), nov-dec 2004.

[3] Marianne Durand and Philippe Flajolet. Loglog counting of large cardinalities. In G. Di Battista and U. Zwick, editors, *Annual European Symposium on Algorithms (ESA03)*, volume 2832 of *Lecture Notes in Computer Science*, pages 605–617, September 2003.

[4] O. Gandouet and A. Jean-Marie. LogLog counting for the estimation of IP traffic. Technical Report forthcoming, INRIA, 2006.

[5] Ilan Kremer, Noam Nissan, and Dana Ron. On randomized one-round communication complexity. In *Proc. STOCS'95*, pages 596–605, 1995.

[6] E. Kushilevitz and N. Nisan. *Communication Complexity*. Cambridge University Press, 1996.

[7] A. Lall, V. Sekar, M. Ogihara, J. Xu, and H. Zhang. Data streaming algorithms for estimating entropy of network traffic. In *Proc. SIGMETRICS/PERFORMANCE'06*, pages 145–156, St Malo, France, june 2006.

[8] Kirill Levchensko, Ramamohan Paturi, and George Varghese. On the difficulty of scalably detecting network attacks. *Computers and Communication Security*, pages 25,29, October 2004.

[9] Michael Saks and Xiaodong Sun. Space lower bounds for distance approximation in the data stream model. In *Proc. STOCS'02*, pages 360–369, 2002.

[10] A. C-C. Yao. Some complexity questions related to distributive computing. In *11th ACM STOC*, pages 209–213, 1979.

A Proof of Theorem 2

In order to prove this result we need the following definition, taken from [5].

Definition 1 (VC-dimension) *Let H be class of boolean functions over a domain Y. We say that a set $S \subset Y$ is* shattered *by H if for every subset $R \subset S$ there exists a function $h_R \in H$ such that $\forall y \in S,\ h_R(y) = 1$ iff $y \in R$. The largest value d for which there exists a set S of size d that is shattered by H is the VC-dimension of H, denoted by VC-dim(H).*

For a binary function $f : X \times Y \to \{0,1\}$, we note f_X the set $\{f(x,.), x \in X\}$ of boolean functions defined on Y. The following theorem has been proved in [5]:

Theorem 7 *For every function $f : X \times Y \to \{0,1\}$, and for every constant error $\varepsilon \leq 1/8$, the one-way randomized communication complexity of f is Θ(VC-dim(f_X)).*

Let us now prove Theorem 2. Since for any $\varepsilon \leq 1/8$ the one-way randomized communication complexity of D is bounded below by the VC-dim of D_X, it is sufficient to prove that VC-dim(D_X) is n in order to prove Theorem 2.

For any subset R of $Y = \{1, \ldots, n\}$, if we choose $x \in X = \{0,1\}^n$ such as $\forall y \in R$, $x_y = 1$ and for any $y \in \{1, \ldots, n\} \setminus R$, $x_y = 0$ then we obtain for this x: $D(x,y) = 1 \Leftrightarrow y \in R$. Indeed it is easy to see that by construction, both directions of the equivalence are straightforward. Therefore, the VC-dim of D_X is $\|\{1, \ldots, n\}\| = n$. This concludes the proof.