Network-Wide Routing-Oblivious Heavy Hitters

Ran Ben Basat
Technion
sran@cs.technion.ac.il

Gil Einziger
Nokia Bell Labs
gil.einziger@nokia.com

Shir Landau Feibish
Princeton University
sfeibish@cs.princeton.edu

Jalil Moraney
Technion
jalilm@cs.technion.ac.il

Danny Raz
Technion
danny@cs.technion.ac.il

ABSTRACT

The recent introduction of SDN allows deploying new centralized network algorithms that dramatically improve the network operation. Many of these solutions rely on the assumption that the centralized controller merges data from different Network Monitoring Points (NMP) to obtain a network-wide view. This is far from trivial when the same packet may traverse through several NMPs. Therefore, existing solutions either assume that each packet is measured at exactly one NMP or that the routing of each packet is known. Another approach is to mark the sampled packets so that other NMPs are aware that the packet was already considered.

We suggest the first network-wide and routing oblivious algorithms for three fundamental network monitoring problems. The suggested algorithms allow flexible NMP placement, require no control over edge routers, and are indifferent to network topology and routing. Moreover, they are based on passive measurements without modifying the traffic in any way. Formally, we provide a general, constant time framework that solves the distributed versions of the volume estimation, frequency estimation and heavy-hitters problems with provable guarantees. The evaluation of our scheme on real packet traces shows that we can achieve very accurate results using a very reasonable amount of memory. For example, using less than 60KB memory in each monitoring point we get a root square error of less than 0.01% of the packets for frequency estimation.

ANCS '18, July 23–24, 2018, Ithaca, NY, USA

ACM ISBN 978-1-4503-5902-3/18/07…$15.00
https://doi.org/10.1145/3230718.3230729

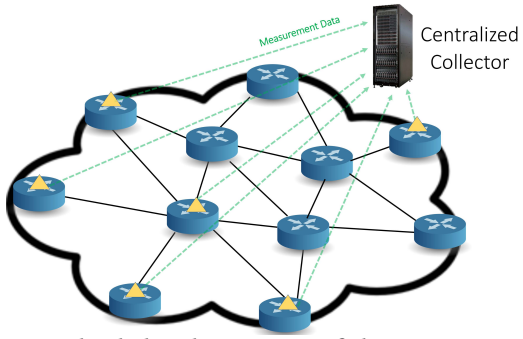


Figure 1: A high-level overview of the scenario considered in this work. Measurement points are placed in some of the network devices and deliver the measurement data to a centralized SDN controller. Our methods require no knowledge of routing and can handle packets that traverse more than a single site without double counting them.

ACM Reference Format:
Ran Ben Basat, Gil Einziger, Shir Landau Feibish, Jalil Moraney, and Danny Raz. 2018. Network-Wide Routing-Oblivious Heavy Hitters. In *ANCS '18: Symposium on Architectures for Networking and Communications Systems, July 23–24, 2018, Ithaca, NY, USA.* ACM, New York, NY, USA, 8 pages. https://doi.org/10.1145/3230718.3230729

1 INTRODUCTION

Many basic network functionalities such as routing, load balancing, QoS enforcement, anomaly detection and intrusion detection require accurate measurements of the network load and identification of the heavy flows [10, 14, 20, 26, 30, 37]. Network measurement is challenging due to the rapid line transmission rates, the massive traffic volume and the scarcity of SRAM memory in routers. Existing works are thus optimized for space and update complexity.

Gathering load and flow statistics at a single (virtual or physical) network device was extensively studied [8, 9, 11, 12, 15, 18, 22, 23, 34, 38]. The data collected at the device is used for local decisions that are made within the switch. However, several network applications such as traffic engineering [14], finding lossy links [33, 42] and identifying super-spreaders and port scans require a network-wide perspective [40].

Problem	Space in each NMP (Bits)	Update Time	Query Time	Theorem #
Distributed Volume Estimation	$O\left(\left(\varepsilon^{-2}+\log n\right)\log\delta^{-1}\right)$	$O(1)$ worst case	$O(1)$	3.1
Distributed Frequency Estimation	$O\left(\left(\varepsilon^{-2}\log\lvert\mathcal{U}\rvert+\log n\right)\log\delta^{-1}\right)$	$O(1)$ amortized	$O(1)$	3.3
Distributed Heavy Hitters	$O\left(\varepsilon^{-2}\log\lvert\mathcal{U}\rvert\log\left(\varepsilon^{-1}\delta^{-1}\right)\right)$	$O(1)$ amortized	$O(\theta^{-1})$	3.5

Table 1: Summary of our results. n is an upper bound on the number of packets ($n \geq |\mathcal{S}|$).

Indeed, network-wide measurement is extensively studied [4, 6, 7, 17, 19, 27, 29, 32, 41]. In this field, the existence of a centralized controller is assumed and the measurement data is collected in some of the network devices. The centralized controller merges the collected data from the *Network Measurement Points (NMPs)* and creates a network-wide view of the entire traffic as illustrated in Figure 1. This view is based on aggregate and should count each packet exactly once. This is a challenging task as packets may traverse multiple NMPs which may cause double counting.

Avoiding double counting is a fundamental challenge for network-wide measurements; yet, to date, a practical solution has not been proposed. Specifically, the majority of existing solutions avoid the problem by assuming that no packet traverses more than a single NMP. Clearly, this assumption requires an understanding of the routing and topology of the network. In the SDN era, routing is very flexible and even a single flow may be routed over multiple paths. Thus, this assumption complicates the measurement mechanism and limits the ability to enjoy the benefits of SDN technologies.

A possible solution to the double counting problem was recently proposed in [4], where the authors suggest to mark packets when they are first measured. Therefore, NMPs need to only count unmarked packets and avoid double counting. The method utilizes unused bits in the packet headers, which means that for correctness (and network friendliness), packets also need to be unmarked as they leave the network. While this solution tackles the fundamental problem it suffers from three major problems: (1) While marking packets is easy, unmarking is much more complex. It requires identifying for each packet the last NMP before it leaves the network (and unmark the packet there). (2) An attacker can easily exploit the mechanism to avoid detection by marking the attack packets. The NMPs would ignore these packets and the attack may go unmonitored. (3) Since the solution uses unused bits, it has a hidden assumption that these bits arrive cleared to the network. This assumption may not hold true for various reasons. Specifically, other entities may use the same bits for their own applications. Hence, marked packets may enter the network and interfere with the measurement. Thus, a solution that does not modify the traffic is preferable.

1.1 Our Contribution

In this work, we suggest routing oblivious algorithms for fundamental network-wide monitoring problems such as estimating the total number of packets or total payload size in the network, estimating per-flow size (packet or byte) and reporting the heavy hitter flows. For each of these problems, we suggest an algorithm that does not modify the packet content in any way and does not assume that packets only traverse a single NMP. Specifically, our algorithms only assume that each packet passes through one or more NMPs.

2 PRELIMINARIES

Our data consists of a *stream* of packets $\mathcal{S} \in (\mathcal{U} \times \mathbb{N})^*$, where each packet $\langle x, i\rangle$ is associated with a *flow identifier* $x \in \mathcal{U}$ and a *unique identifier* $i \in \mathbb{N}$. Here, $\mathcal{U}$ is the *universe* from which the identifiers are taken. For example, $\mathcal{U}$ may be all 32-bits source IPs, 64-bit source and destination IP pairs, 5-tuples, etc. Different packets may have the same unique identifiers, but each packet is associated with a distinct (flow identifier, unique identifier) pair. For the TCP protocol, we can use the sequence number as a unique identifier. For other protocols, the works of [21, 42] suggest various methods to distinctly identify packets according to their header fields. This paper assumes that such a unique identifier exists for each packet.

The network consists of k *measurement points* $R_1, \ldots R_k$. Each such network measurement point (or NMP) R_r observes a subsequence of the stream $\mathcal{S}_i \subseteq \mathcal{S}$ such that $\cup_{i=1}^{k}\mathcal{S}_i = \mathcal{S}$. That is, we assume that each packet traverses *at least one* NMP but may also traverse multiple NMPs. This model is considerably more general than the one studied in previous works [27, 29, 39]. The fundamental difference is that these works assumed that each packet traverses a **single** NMP and therefore no packet is counted twice.

The *frequency* of a flow $x \in \mathcal{U}$ is defined as the number of packets that belong to the flow: $f_x \triangleq |\{\langle x, i\rangle \in \mathcal{S}\}|$. We summarize the notations used in this paper in Table 2.

2.1 Count Distinct Algorithms

Our work utilizes a count distinct algorithm $\mathbb{A}$ as a black box. We assume that $\mathbb{A}$ allows processing of a stream and upon query provides a $(1+\varepsilon_{\mathbb{A}})$-approximation of the number of distinct items in the stream with probability $\geq 1-\delta_{\mathbb{A}}$. $\mathbb{A}$ supports three functions; $\mathbb{A}$.Add$(\cdot)$ processes elements, $\mathbb{A}$.Query returns an estimate to the number of distinct elements, and Merge$(\mathbb{A}_1, \mathbb{A}_2)$ returns a merge of the instances $\mathbb{A}_1$ and $\mathbb{A}_2$. That is, assume that the sets of distinct elements observed by $\mathbb{A}_1$ and $\mathbb{A}_2$ are S_1 and S_2, then Merge$(\mathbb{A}_1, \mathbb{A}_2)$

Symbol	Meaning
$\mathcal{S}$	The packet stream
n	An upper bound on the length of the stream ($n \geq \lvert\mathcal{S}\rvert$)
NMP	Network Measurement Point
$\langle x, i\rangle$	A packet from flow x with sequence number i
$\mathcal{U}$	The universe of flow identifiers
f_x	The frequency of flow $x \in \mathcal{U}$
k	The number of routers in the network
R_i	Router number i ($i \in \{1, \ldots, k\}$)
$\mathcal{S}_i$	The sub-stream seen by router R_i ($\cup_{i\in\{1,\ldots,k\}}\mathcal{S}_i = \mathcal{S}$)
$\widehat{V}$	an estimate for $\lvert\mathcal{S}\rvert$
ε	The goal error parameter
δ	The goal error probability
$\widehat{f_x}$	An estimate for the frequency of flow x
θ	Heavy hitter threshold
$\mathbb{A}$	A Count Distinct algorithm with an $(\varepsilon_{\mathbb{A}}, \delta_{\mathbb{A}})$ guarantee
$S_{\varepsilon_{\mathbb{A}},\delta_{\mathbb{A}}}$	The space that $\mathbb{A}$ requires
χ	The sample size required for an (ε, δ) approximation as in Lemma 3.2 ($\chi = 3\varepsilon^{-2}\log 2\delta^{-1}$)

Table 2: A list of symbols and notations

is an instance that monitors $S_1 \cup S_2$. These assumptions are satisfied by many count distinct algorithms [5, 24, 25, 28, 31].

Intuitively, count distinct algorithms calculate a hashed value of each item and then remember certain patterns about the observed values (e.g., the maximal or minimal values). Statistical reasoning is then applied to determine the number of distinct items most likely to yield the observed pattern.

We denote by $S_{\varepsilon_{\mathbb{A}},\delta_{\mathbb{A}}}$ the space required by S; we note that a $S_{\varepsilon_{\mathbb{A}},\delta_{\mathbb{A}}} = O\left(\left(\varepsilon_{\mathbb{A}}^{-2} + \log n\right)\log \delta_{\mathbb{A}}^{-1}\right)$ bits algorithm is known [31] for $\leq n$ distinct elements. Further, this algorithm processes new elements and answers queries in $O(1)$ time.

2.2 Problem Definitions

We consider the following network-wide problems:

- (ε, δ)-Distributed Volume Estimation: Return an estimator $\widehat{V}$ for the overall number of packets in the network. With probability $1-\delta$, $\widehat{V}$ is a $(1+\varepsilon)$ multiplicative approximation for $\lvert\mathcal{S}\rvert$.
- (ε, δ)-Distributed Frequency Estimation: given a query for flow $x \in \mathcal{U}$, return an estimator $\widehat{f_x}$ that satisfies: $\Pr\left[\left|\widehat{f_x} - f_x\right| > \lvert\mathcal{S}\rvert\varepsilon\right] \leq \delta$.
- (ε, δ)-Distributed Heavy Hitters: Upon a query with parameter θ, return a set $S \subseteq \mathcal{U}$ such that:
  (1) $\Pr[\exists x \mid f_x \geq \lvert\mathcal{S}\rvert\theta \wedge x \notin \mathcal{S}] \leq \delta$, and
  (2) for all $y \in \mathcal{U}$ such that $f_y < \lvert\mathcal{S}\rvert(\theta - \varepsilon)$ we have that $\Pr[y \in \mathcal{S}] \leq \delta\varepsilon$.

3 ALGORITHMS

We now provide algorithms for the problems presented in Section 2.2. Table 1 provides a summary of our algorithms.

3.1 Distributed Volume Estimation

In distributed settings, it is not trivial to measure the total traffic volume as packets may be counted multiple times along their route. In principle, existing approaches avoid this problem by assuming that each packet is only monitored at a single place. In this work, we provide a solution that tracks the overall number of packets without assumptions on the network topology, routing, or the order of packets. Thus, we explicitly allow packets to be routed through multiple NMPs.

We avoid double counting using a count distinct algorithm ($\mathbb{A}$) to measure the number of **distinct** packets. Attaining distinct ids for packet is addressed in [33, 42].

That is, each NMP maintains such an algorithm. To record a packet $\langle x, i\rangle$, we simply add x to the distinct count of $\mathbb{A}$ ($\mathbb{A}.Add(\langle x, i\rangle)$). The NMPs synchronize with the controller at the end of the measurement and each NMP transmits its state to the controller. Count distinct algorithms can be merged and the result is exactly the same as if the measurement was performed on a single NMP that observed the entire traffic. Therefore, the controller achieves a network-wide and provably accurate estimate of the total number of packets regardless of topology, routing and the order of the packets.

Theorem 3.1, whose proof follows immediately, shows the correctness of our approach

Theorem 3.1. *Let $\varepsilon_{\mathbb{A}} = \varepsilon, \delta_{\mathbb{A}} = \delta$ be the parameters for the algorithm $\mathbb{A}$ that runs on each NMP. Then the controller solves* (ε, δ)-Distributed Volume Estimation.

3.2 Distributed Sampling

Intuitively, we use samples to find the heavy hitters and estimate per-flow frequency. The following lemma explains how to estimate frequencies from a uniform sample.

Lemma 3.2. *Denote $\chi \triangleq 3\varepsilon^{-2}\log 2\delta^{-1}$. Let $\mathcal{S}' \in \binom{\mathcal{S}}{\chi}$ be a random subset of size χ of $\mathcal{S}$ and let $p \triangleq \frac{\chi}{\lvert\mathcal{S}\rvert}$ be the probability in which each packet is sampled. If we denote by f'_x the frequency of a flow $x \in \mathcal{U}$ in $\mathcal{S}'$, then*
$\Pr\left[\lvert p^{-1}f'_x - f_x\rvert > \lvert\mathcal{S}\rvert\varepsilon\right] \leq \delta$.

Proof. We use the Chernoff Bound [35] that states that for all $0 < t, p' \leq 1$ $n \in \mathbb{N}$ a binomial random variable $X \sim \mathrm{Bin}(n, p')$ satisfies $\Pr[\lvert X - np'\rvert \geq t \cdot np'] \leq 2e^{-np't^2/3}$. Let X denote the number of appearances of x in $\mathcal{S}'$. Then $X \sim \mathrm{Bin}(f_x, p)$ and according to the inequality

$$\Pr\left[\lvert p^{-1}f'_x - f_x\rvert > \lvert\mathcal{S}\rvert\varepsilon\right] \leq \Pr\left[\lvert f'_x - pf_x\rvert > pf_x\varepsilon\right]$$
$$\leq 2e^{-f_x p\varepsilon^2/3} = 2e^{-f_x\varepsilon^2\frac{\chi}{3\lvert\mathcal{S}\rvert}} = 2e^{-\frac{f_x\log 2\delta^{-1}}{\lvert\mathcal{S}\rvert}} \leq 2e^{-\log 2\delta^{-1}} = \delta.$$

□

In the distributed case, random sampling does not generate a uniform sample as each NMP samples a different substream. Thus, packets that traverse multiple NMPs are oversampled

and the result is not a uniform sample. This means, that the frequencies of packets cannot be directly extracted from the sample and limits its usability. Therefore, we briefly describe how to overcome this problem and provide the controller with a uniform sample that can be used for heavy hitters and frequency estimation.

Intuitively, we use a random hash function [1] $h : \{0,1\}^* \to [0,1]$ that returns a uniformly distributed random number for each hashed bit-string given as key. The hash function is known to the controller and all routers. Each router applies h on each arriving packet $\langle x, i\rangle$ and stores the χ packets with the highest h value. When the measurement ends, each NMP sends its state to the controller. The controller then merges all reports to obtain the χ packets that have the highest h values. As this is equivalent to finding the χ packets from $\mathcal{S}$ with the highest h value, and since h is random, we have that the controller obtains a uniform sample of χ packets [2]. The analysis in [16] shows that the expected number of updates each NMP R_i makes to obtain a χ-sized sample is $O(\chi \log |\mathcal{S}_i|)$. This implies, as each heap updates takes $O(\log \chi)$ time, that if the number of packets the NMP observes is $\Omega(\chi \log \chi)$ then its amortized update time becomes constant.

3.3 Distributed Frequency Estimation

The above sampling technique allows one to estimate the frequency of a flow if the effective sampling probability $p = \frac{\chi}{|\mathcal{S}|}$ is known (see Lemma 3.2). Unfortunately, while $\chi = 3\varepsilon^{-2}\log 2\delta^{-1}$ is known, $|\mathcal{S}|$ is not. We circumvent this problem by running a volume estimation algorithm as in Section 3.1 in parallel to a distributed sampling instance. Specifically, we show that by setting the error parameters of the volume estimation to $\varepsilon_V = \varepsilon/3$ and $\delta_V = \delta/2$ and the parameters of the distributed sampling to $\varepsilon_s = \varepsilon/2$ and $\delta_s = \delta/2$ (i.e., each router tracks $\chi = 12\varepsilon^{-2}\log 4\delta^{-1}$ packets), we can solve the problem by returning $\widehat{f_x} \triangleq f'_x\widehat{V}/\chi$. Theorem 3.3 is the main theoretical result for this section.

Theorem 3.3. *There exists an algorithm that requires: $O\left(\left(\varepsilon^{-2} + \log n + \log|\mathcal{U}|\right)\log\delta^{-1}\right)$ bits at each NMP, processes packets and answers queries in $O(1)$ amortized time, and solves (ε,δ)-*Distributed Frequency Estimation.

3.4 Finding the Heavy Hitters

In the non distributed case, heavy hitters are a by-product of frequency estimation. For example, we can find heavy hitters by returning all flows whose estimate is above the threshold. Surprisingly, we can solve the heavy hitter problem directly. That is, we use our distributed sampling technique to obtain a uniform sample and check which flows are heavy in the sample. Our algorithm is simpler than that of the previous section – we run a distributed sampling protocol and let the controller infer the heavy hitters upon query. The next lemma lays the theoretical foundation of our algorithm.

Lemma 3.4. *Let $\varepsilon,\delta,\theta > 0$ and $\chi' \triangleq \left\lceil 9\varepsilon^{-2}\log(2\delta^{-1}\varepsilon^{-1})\right\rceil$. Let $\mathcal{S}' \in \binom{\mathcal{S}}{\chi'}$ be a random subset of size χ' of $\mathcal{S}$. If we denote by f'_x the frequency of a flow $x \in \mathcal{U}$ in $\mathcal{S}'$, then $S \triangleq \left\{x \in \mathcal{S}' \mid f'_x \geq (\theta - \varepsilon/2)\chi'\right\}$ solves (ε,δ)-*Distributed Heavy Hitters.

Applying this lemma, we have the following theorem.

Theorem 3.5. *There exists an algorithm that requires $O\left(\varepsilon^{-2}\log\left(\delta^{-1}\varepsilon^{-1}\right)\right)$ bits at each NMP, processes packets in $O(1)$ amortized time, answers queries in $O(\theta^{-1})$ time, and solves (ε,δ)-*Distributed Heavy Hitters.

4 ANALYSIS

In this section, we provide the needed analysis and the proofs for the correctness of our algorithms.

4.1 Frequency Estimation Algorithm

We proceed by proving Theorem 3.3.

Theorem 3.3. *There exists an algorithm that requires: $O\left(\left(\varepsilon^{-2} + \log n + \log|\mathcal{U}|\right)\log\delta^{-1}\right)$ bits at each NMP, processes packets and answers queries in $O(1)$ worst case time, and solves (ε,δ)-*Distributed Frequency Estimation.

Proof. Recall that our algorithm sets the error parameters of the volume estimation to $\varepsilon_V = \varepsilon/3$ and $\delta_V = \delta/2$ and the parameters of the distributed sampling to $\varepsilon_s = \varepsilon/2$ and $\delta_s = \delta/2$. This means that by the correctness of the Distributed Volume Estimation algorithm, we have that $\widehat{V}$ is a $(1+\varepsilon_V)$ multiplicative approximation of $|\mathcal{S}|$, i.e., $\Pr\left[|\widehat{V} - |\mathcal{S}|| > |\mathcal{S}|\varepsilon_V\right] \leq \delta_V$. Recall that we observe the sampled frequency of the queried item f'_x, and that according to Lemma 3.2:
$\Pr\left[\left|\frac{|\mathcal{S}|}{\chi}f'_x - f_x\right| > |\mathcal{S}|\varepsilon_s\right] = \Pr\left[|p^{-1}f'_x - f_x| > |\mathcal{S}|\varepsilon_s\right] \leq \delta_s$.
Since $\delta_V + \delta_s = \delta$, we have that with probability $1-\delta$ $\left|\frac{|\mathcal{S}|}{\chi}f'_x - f_x\right| \leq |\mathcal{S}|\varepsilon_s$ *and* $|\widehat{V} - \mathcal{S}| \leq |\mathcal{S}|\varepsilon_V$. Recall that our estimator is $\widehat{f_x} = f'_x\widehat{V}/\chi$. Therefore, since $0 \leq f_x \leq |\mathcal{S}|$ and $\varepsilon < 1$, we get:

$$\begin{aligned}\widehat{f_x} - f_x &= f'_x\widehat{V}/\chi - f_x \leq (1+\varepsilon_V)f'_x|\mathcal{S}|/\chi - f_x\\ &\leq (1+\varepsilon_V)\left(f'_x|\mathcal{S}|/\chi - f_x\right) + |\mathcal{S}|\varepsilon_V\\ &\leq (1+\varepsilon_V)|\mathcal{S}|\varepsilon_s + |\mathcal{S}|\varepsilon_V = |\mathcal{S}|(\varepsilon_V + \varepsilon_s + \varepsilon_V\varepsilon_s) = |\mathcal{S}|\varepsilon.\end{aligned}$$

Similarly:

[1] Such hash functions do not exist in practice, but one can use a hash function that returns $(2\log|\mathcal{U}|\log\delta^{-1})$ random bits instead; for simplicity, we avoid the analysis of discretized image values.

[2] We also note that if $|\mathcal{S}| \leq \chi$ then no sampling takes place and the router sees the entire data and computes all frequencies with no error.

$$\widehat{f_x} - f_x = f'_x\widehat{V}/\chi - f_x \geq (1-\varepsilon_V)f'_x|\mathcal{S}|/\chi - f_x$$
$$\geq (1-\varepsilon_V)\left(f'_x|\mathcal{S}|/\chi - f_x\right) - |\mathcal{S}|\varepsilon_V$$
$$\geq -(1-\varepsilon_V)|\mathcal{S}|\varepsilon_s - |\mathcal{S}|\varepsilon_V = -|\mathcal{S}|(\varepsilon_V + \varepsilon_s - \varepsilon_V\varepsilon_s) \geq -|\mathcal{S}|\varepsilon.$$

We thus conclude that $\Pr\left[\left|\widehat{f_x} - f_x\right| \geq |\mathcal{S}|\varepsilon\right] \leq \delta_V + \delta_s = \delta$. Finally, since $\varepsilon_V = \varepsilon_s = \Theta(\varepsilon)$ and $\delta_V = \delta_s = \Theta(\delta)$, the memory per NMP is $O\left(\left(\varepsilon^{-2} + \log n + \log|\mathcal{U}|\right)\log\delta^{-1}\right)$. □

4.2 Heavy Hitters Algorithm

We proceed with the proof of Lemma 3.4.

Lemma 3.4. *Let $\varepsilon, \delta, \theta > 0$ and $\chi' \triangleq \lceil 9\varepsilon^{-2}\log(2\delta^{-1}\varepsilon^{-1})\rceil$. Let $\mathcal{S}' \in \binom{\mathcal{S}}{\chi'}$ be a random subset of size χ' of $\mathcal{S}$. If we denote by f'_x the frequency of a flow $x \in \mathcal{U}$ in $\mathcal{S}'$, then $S \triangleq \left\{x \in \mathcal{S}' \mid f'_x \geq (\theta - \varepsilon/2)\chi'\right\}$ solves (ε, δ)-*Distributed Heavy Hitters.

Proof. Denote by $H \triangleq \{x \in \mathcal{U} \mid f_x \geq |\mathcal{S}|\theta\}$ be the set of true heavy hitters, and let $x \in H$. We apply Lemma 3.2 with $\delta' = \delta\varepsilon$ and $\varepsilon' = \varepsilon/2$ to get

$$\Pr\left[f'_x < (\theta - \varepsilon/2)\chi'\right] = \Pr\left[f'_x < (\theta - \varepsilon')\chi'\right]$$
$$= \Pr\left[\frac{|\mathcal{S}|}{\chi'}f'_x < |\mathcal{S}|(\theta - \varepsilon')\right] = \Pr\left[\frac{|\mathcal{S}|}{\chi'}f'_x - f_x < |\mathcal{S}|(\theta - \varepsilon') - f_x\right]$$
$$\leq \Pr\left[\frac{|\mathcal{S}|}{\chi'}f'_x - f_x < |\mathcal{S}|(\theta - \varepsilon') - |\mathcal{S}|\theta\right]$$
$$= \Pr\left[\frac{|\mathcal{S}|}{\chi'}f'_x - f_x < -|\mathcal{S}|\varepsilon'\right] \leq \Pr\left[\left|\frac{|\mathcal{S}|}{\chi'}f'_x - f_x\right| > |\mathcal{S}|\varepsilon'\right] \leq \delta'.$$

Thus, the probability of each $x \in H$ to not be reported is at most δ'. We then use the Union bound to conclude that the probability *all* of H is successfully reported in H is at least $1 - |H|\delta' \geq 1 - \theta^{-1}\delta' \geq 1 - \varepsilon^{-1}\delta' \geq 1 - \delta$. Next, let $NH \triangleq \left\{y \in \mathcal{U} \mid f_y < |\mathcal{S}|(\theta - \varepsilon)\right\}$ be the set of non-heavy flows and let $y \in NH$. Then, we use Lemma 3.2 for y to obtain

$$\Pr\left[f'_y > (\theta - \varepsilon/2)\chi'\right] = \Pr\left[\frac{|\mathcal{S}|}{\chi'}f'_y > |\mathcal{S}|(\theta - \varepsilon')\right]$$
$$= \Pr\left[\frac{|\mathcal{S}|}{\chi'}f'_y - f_y > |\mathcal{S}|(\theta - \varepsilon') - f_y\right]$$
$$\leq \Pr\left[\frac{|\mathcal{S}|}{\chi'}f'_y - f_y > |\mathcal{S}|(\theta - \varepsilon') - |\mathcal{S}|(\theta - \varepsilon)\right]$$
$$= \Pr\left[\frac{|\mathcal{S}|}{\chi'}f'_y - f_y > |\mathcal{S}|\varepsilon'\right] \leq \Pr\left[\left|\frac{|\mathcal{S}|}{\chi'}f'_y - f_y\right| > |\mathcal{S}|\varepsilon'\right] \leq \delta' = \delta\varepsilon.$$

We thus conclude that our method of examining just the sampled set $\mathcal{S}'$ solves Distributed Heavy Hitters. □

Based on the correctness of Lemma 3.4, the space requirement stated in Theorem 3.5 follows immediately. For answering queries in $O(\theta^{-1})$ time, the controller keeps the histogram of $\mathcal{S}'$ in an array sorted by the observed frequencies $\left\{f'_x\right\}$.

5 EVALUATION

In order to study the actual tradeoff between the amount of memory used and the accuracy of the relevant estimations, we evaluate our algorithms on real traffic traces. Our C++ prototypes are online [36] and we used an open source implementation HyperLogLog [2].

Datasets: We used the following datasets:

(1) The CAIDA Anonymized Internet Trace 2016 [3]. From the "Equinix-Chicago" high-speed monitor.
(2) Border router trace from the CS department at the University of California, Los Angeles, denoted by UCLA [1].
(3) A data center trace [13], denoted by UNIV.

Metrics: We consider the following performance metrics:

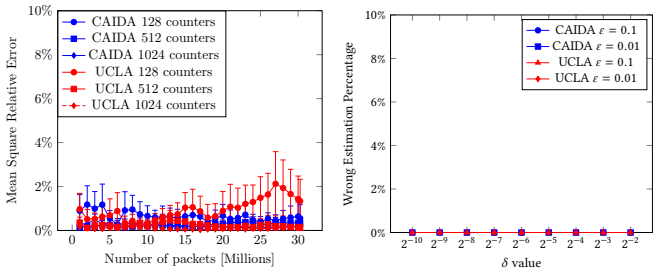
(1) Mean Square Relative Error (MSRE): Measures the average of the squares of the relative errors, i.e., given the estimations $(a_1, a_2, ..., a_n)$ and true values $t_1, t_2, ..., t_n$, the MSRE is: $\frac{1}{n}\sum_{i=1}^{i=n}(\frac{a_i - t_i}{t_i})^2$.
(2) Root Mean Square Error (RMSE): Measures the differences between predicted values of an estimator to actual values. Formally, for each flow x the estimated frequency is $\widehat{f_x}$ and real frequency is f_x. RMSE is calculated as: $\sqrt{\frac{1}{N}\sum_x(\widehat{f_x} - f_x)^2}$.
(3) Wrong Estimation Percentage (WEP): The percentage of flows in which the absolute difference between their estimation and their real frequency is larger than $|S|\varepsilon$.
(4) False Positive and False Negative Ratio (FPR) and (FNR).

5.1 Evaluation Results

As we proved in the previous sections, the accuracy of our method depends only on the set of packets in the network during the monitoring period and does not depend on the placement of the NMPs, the routing of the flows, or the actual network topology. Thus we evaluated a single instance of our algorithm. The accuracy is identical, to the one achieved when any number of instances are merged together.

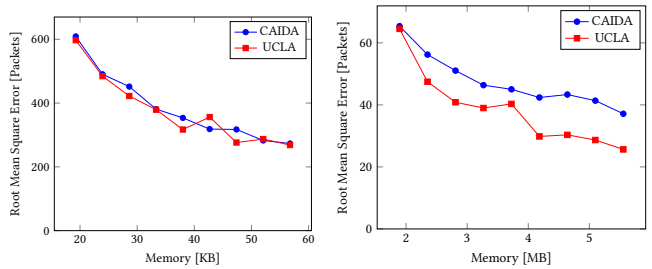
5.1.1 Volume Estimation. Figure 2a shows results for estimating the total number of unique packets in the network. As can be observed, the results are accurate with a small number of counters and accuracy improves as the trace prolongs.

5.1.2 Frequency Estimation. Figure 2b shows results for the Frequency Estimation problem. In this experiment, we vary δ and measure how many wrong estimations we see. That is estimations that vary by more than $|S|\varepsilon$. As can be observed, the accuracy is much better in practice than indicated by our analysis. The difference is because our analysis is a worse case analysis whereas the traffic is taken from real packet traces. Note that in this figure, the error is always 0. That is, we did not observe any wrong estimations regardless of the value of δ. This can be explained by our



(a) (b)

Figure 2: (a) MSRE of the volume estimation algorithm. (b) Wrong estimation percentage of the frequency estimation algorithm for varying δ.



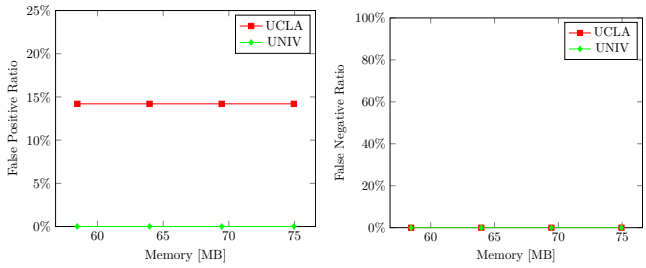
(a) $\varepsilon = 0.1$ **(b)** $\varepsilon = 0.01$

Figure 3: RMSE in frequency estimation and varying δ.

analysis; Lemma 3.2 discusses the sample size χ required for adversarial streams. Namely, we use the fact that $f_x \leq |\mathcal{S}|$ and use $e^{f_x/|\mathcal{S}|} \leq e^1$. If we know that $f_x \ll |\mathcal{S}|$, we can get a far better bound. Namely, in CAIDA the largest flow accounts for just 0.6% and in UCLA the largest takes about 5% of the stream. This means that the error probability is over 400 million smaller than the guaranteed δ.

Figure 3a quantifies the root mean square error (RMSE) for $\varepsilon = 0.1$ and Figure 3b for $\varepsilon = 0.01$ with varying delta values. We evaluate the required space including all overheads of our algorithms and compare the space to accuracy tradeoff of our approach. First, notice that both configurations are very accurate in practice on both the CAIDA and the UCLA trace. Indeed, allocating more memory improves the empirical error. Specifically, notice that increasing the memory from $60KB$ to 6 MB (x100) reduces the error by a factor of ≈ 10 as is indicated by our squared dependence on ε. We selected these parameters as they are within the operating range of network devices. We note that the RMSE of 300 packets corresponds to less than 0.01% of $|\mathcal{S}|$ which shows that the empirical performance on real traces is even better than guaranteed by the analysis.

5.1.3 Heavy Hitters. Figure 4a, shows the false positive ratio of the algorithm as a function of the memory. For the given $\theta = 0.01$, there were no heavy hitters in the CAIDA trace and our algorithm indeed did not identify any flow as



(a) False Positive Ratio **(b) False Negative Ratio**

Figure 4: False Positive/Negative Ratio for heavy hitter detection with $\theta = 0.01, \varepsilon = 0.005$ and varying δ.

a heavy hitter. We can see that for UCLA there where seven heavy hitter flows for the given threshold, and the algorithm had identified mistakingly a single flow as a heavy hitter. In the UNIV trace, the mean FPR is in the range 0.4%-0.5%.

Figure 4b, shows the false negative ratio of the algorithm's a function of the memory. For both traces, the actual FNR is 0. Similarly to Figure 2b, the low error is explained by the fact that our analysis reflects an adversarial setting in which the stream contains just a single flow; as the largest flow is much smaller than the overall traffic volume, the effective error probability is much lower than δ.

6 DISCUSSION

Our work shows the feasibility of attaining a network-wide measurement in the most general model, without traffic manipulations or prior knowledge of the routing protocols, the network topology or the order of the packets. We allow the network monitoring points (NMPs) to be placed anywhere in the network, as long as each packet is covered by at least one of them. This is in contrast to the majority of prior works that assume that each packet is counted exactly once, which implies a severe limitation on their deployment.

We studied three fundamental network problems under this model; (i) total volume within a network, (ii) providing per-flow frequency estimations and (iii) finding the heavy hitters. We introduced an efficient algorithm for each problem and formally proved accuracy guarantees as well as analyzed the runtime complexity. Our analysis is accompanied by an evaluation of the schemes on real Internet packet traces. Our evaluation shows that the proposed approach works well in practice and within the performance parameters of network devices. Our code is released as open source [36].

Acknowledgements. We would like to thank the anonymous reviewers, as well as our shepherd Paolo Costa. This work is partially sponsored by the Technion HPI research school and by the Eric and Wendy Schmidt Fund for Strategic Innovation.

REFERENCES

[1] http://www.lasr.cs.ucla.edu/ddos/traces/.

[2] C++ implementation of hyperloglog - available: https://github.com/hideo55/cpp-hyperloglog.

[3] The caida ucsd anonymized internet traces 2016 - january. 21st.

[4] Yehuda Afek, Anat Bremler-Barr, Shir Landau Feibish, and Liron Schiff. Detecting heavy flows in the SDN match and action model. *Computer Networks*, 136:1 – 12, 2018.

[5] Pankaj K. Agarwal, Graham Cormode, Zengfeng Huang, Jeff Phillips, Zhewei Wei, and Ke Yi. Mergeable summaries. In *ACM PODS*, 2012.

[6] Daniel Anderson, Pryce Bevan, Kevin Lang, Edo Liberty, Lee Rhodes, and Justin Thaler. A high-performance algorithm for identifying frequent items in data streams. In *Proceedings of the 2017 Internet Measurement Conference*, IMC '17, pages 268–282, New York, NY, USA, 2017. ACM.

[7] Mina Tahmasbi Arashloo, Yaron Koral, Michael Greenberg, Jennifer Rexford, and David Walker. Snap: Stateful network-wide abstractions for packet processing. In *Proceedings of the 2016 ACM SIGCOMM Conference*, pages 29–43. ACM, 2016.

[8] Ran Ben Basat, Gil Einziger, Roy Friedman, and Yaron Kassner. Heavy hitters in streams and sliding windows. In *IEEE INFOCOM*, 2016.

[9] Ran Ben Basat, Gil Einziger, Roy Friedman, and Yaron Kassner. Optimal elephant flow detection. In *IEEE INFOCOM*, 2017.

[10] Ran Ben Basat, Gil Einziger, Roy Friedman, and Yaron Kassner. Randomized admission policy for efficient top-k and frequency estimation. In *IEEE INFOCOM*, 2017.

[11] Ran Ben Basat, Gil Einziger, Roy Friedman, Marcelo Caggiani Luizelli, and Erez Waisbard. Constant time updates in hierarchical heavy hitters. *ACM SIGCOMM*, 2017.

[12] Ran Ben-Basat, Gil Einziger, and Roy Friedman. Fast flow volume estimation. In *ACM ICDCN*, 2018.

[13] Theophilus Benson, Aditya Akella, and David A. Maltz. Network traffic characteristics of data centers in the wild. In Mark Allman, editor, *Proceedings of the 10th ACM SIGCOMM Internet Measurement Conference, IMC 2010, Melbourne, Australia - November 1-3, 2010*, pages 267–280. ACM, 2010.

[14] Theophilus Benson, Ashok Anand, Aditya Akella, and Ming Zhang. Microte: Fine grained traffic engineering for data centers. In *ACM CoNEXT*, page 8, 2011.

[15] Min Chen, Shigang Chen, and Zhiping Cai. Counter tree: A scalable counter architecture for per-flow traffic measurement. *IEEE/ACM Transactions on Networking (TON)*, 2017.

[16] Edith Cohen and Haim Kaplan. Summarizing data using bottom-k sketches. In *Proceedings of the twenty-sixth annual ACM symposium on Principles of distributed computing*, pages 225–234. ACM, 2007.

[17] Graham Cormode. Continuous distributed monitoring: A short survey. In *Proceedings of the First International Workshop on Algorithms and Models for Distributed Event Processing*, AlMoDEP '11, pages 1–10, New York, NY, USA, 2011. ACM.

[18] Graham Cormode and Marios Hadjieleftheriou. Finding frequent items in data streams. *Proc. VLDB Endow.*, 1(2):1530–1541, August 2008. Code: www.research.att.com/ marioh/frequent-items.html.

[19] Xenofontas Dimitropoulos, Paul Hurley, and Andreas Kind. Probabilistic lossy counting: An efficient algorithm for finding heavy hitters. *SIGCOMM Comput. Commun. Rev.*, 38(1), January 2008.

[20] Gero Dittmann and Andreas Herkersdorf. Network processor load balancing for high-speed links. In *Proc. of the 2002 Int. Symp. on Performance Evaluation of Computer and Telecommunication Systems*, volume 735.

[21] N. G. Duffield and Matthias Grossglauser. Trajectory sampling for direct traffic observation. *IEEE/ACM Trans. Netw.*, 2001.

[22] Gil Einziger, Marcelo Caggiani Luizelli, and Erez Waisbard. Constant time weighted frequency estimation for virtual network functionalities. In *IEEE ICCCN*, 2017.

[23] Cristian Estan and George Varghese. New directions in traffic measurement and accounting. *SIGCOMM Comput. Commun. Rev.*, 32(4), August 2002.

[24] Philippe Flajolet and G. Nigel Martin. Probabilistic counting algorithms for data base applications. *J. Comput. Syst. Sci.*, 1985.

[25] Philippe Flajolet, ï£¡ric Fusy, Olivier Gandouet, and et al. Hyperloglog: The analysis of a near-optimal cardinality estimation algorithm. In *AOFA*, 2007.

[26] Pedro Garcia-Teodoro, JesÃžs E. DÃŋaz-Verdejo, Gabriel MaciÃą-FernÃąndez, and E. VÃązquez. Anomaly-based network intrusion detection: Techniques, systems and challenges. *Computers and Security*, pages 18–28, 2009.

[27] Rob Harrison, Qizhe Cai, Arpit Gupta, and Jennifer Rexford. Network-wide heavy hitter detection with commodity switches. In *SOSR*. ACM, 2018.

[28] Stefan Heule, Marc Nunkesser, and Alexander Hall. Hyperloglog in practice: Algorithmic engineering of a state of the art cardinality estimation algorithm. In *ACM EDBT*, 2013.

[29] Qun Huang, Xin Jin, Patrick P. C. Lee, Runhui Li, Lu Tang, Yi-Chao Chen, and Gong Zhang. Sketchvisor: Robust network measurement for software packet processing. In *Proceedings of the Conference of the ACM Special Interest Group on Data Communication*, SIGCOMM '17, pages 113–126, New York, NY, USA, 2017. ACM.

[30] Abdul Kabbani, Mohammad Alizadeh, Masato Yasuda, Rong Pan, and Balaji Prabhakar. Af-qcn: Approximate fairness with quantized congestion notification for multi-tenanted data centers. In *Proc. of the 18th IEEE Symposium on High Performance Interconnects*, HOTI, 2010.

[31] Daniel M. Kane, Jelani Nelson, and David P. Woodruff. An optimal algorithm for the distinct elements problem. In *ACM PODS*, 2010.

[32] Yuliang Li, Rui Miao, Changhoon Kim, and Minlan Yu. Flowradar: A better netflow for data centers. In *13th USENIX Symposium on Networked Systems Design and Implementation (NSDI 16)*, pages 311–324, Santa Clara, CA, 2016. USENIX Association.

[33] Yuliang Li, Rui Miao, Changhoon Kim, and Minlan Yu. Lossradar: Fast detection of lost packets in data center networks. In *Proceedings of the 12th International on Conference on Emerging Networking EXperiments and Technologies*, CoNEXT '16, pages 481–495, New York, NY, USA, 2016. ACM.

[34] Ahmed Metwally, Divyakant Agrawal, and Amr El Abbadi. Efficient computation of frequent and top-k elements in data streams. In *IN ICDT*, 2005.

[35] Michael Mitzenmacher and Eli Upfal. *Probability and Computing: Randomized Algorithms and Probabilistic Analysis*. Cambridge University Press, 2005.

[36] Jalil Moraney. Network-wide routing-oblivious heavy hitters - available: https://github.com/jalilm/tdhh.

[37] B. Mukherjee, L.T. Heberlein, and K.N. Levitt. Network intrusion detection. *Network, IEEE*, 8(3), May 1994.

[38] Vibhaalakshmi Sivaraman, Srinivas Narayana, Ori Rottenstreich, S. Muthukrishnan, and Jennifer Rexford. Heavy-hitter detection entirely in the data plane. In *Proceedings of the Symposium on SDN Research*, ACM SOSR, pages 164–176, 2017.

[39] Ke Yi and Qin Zhang. Optimal tracking of distributed heavy hitters and quantiles. *Algorithmica*, 65(1):206–223, Jan 2013.

[40] Minlan Yu, Lavanya Jose, and Rui Miao. Software defined traffic measurement with opensketch. In *Presented as part of the 10th USENIX Symposium on Networked Systems Design and Implementation (NSDI 13)*, pages 29–42, Lombard, IL, 2013. USENIX.

[41] Haiquan Zhao, Ashwin Lall, Mitsunori Ogihara, and Jun Xu. Global iceberg detection over distributed data streams. In *IEEE ICDE*, 2010.

[42] Yibo Zhu, Nanxi Kang, Jiaxin Cao, Albert Greenberg, Guohan Lu, Ratul Mahajan, Dave Maltz, Lihua Yuan, Ming Zhang, Ben Y. Zhao, and Haitao Zheng. Packet-level telemetry in large datacenter networks. In *Proceedings of the 2015 ACM Conference on Special Interest Group on Data Communication*, SIGCOMM '15, pages 479–491, New York, NY, USA, 2015. ACM.