I Know What You Did Last Summer: Network Monitoring using Interval Queries

Nikita Ivkin
Amazon*

Ran Ben Basat
Harvard University

Zaoxing Liu
Carnegie Mellon University

Gil Einziger
Ben Gurion University

Roy Friedman
Technion

Vladimir Braverman
Johns Hopkins University

ABSTRACT

Modern telemetry systems require advanced analytic capabilities such as drill down queries. These queries can be used to detect the beginning and end of a network anomaly by efficiently refining the search space. We present the first integral solution that (i) enables multiple measurement tasks inside the same data structure, (ii) supports specifying the time frame of interest as part of its queries, and (iii) is sketch-based and thus space efficient. Namely, our approach allows the user to define both the measurement task (e.g., heavy hitters, entropy estimation, cardinality estimation) and the time frame of relevance (e.g., 5PM-6PM) at query time. Our approach provides accuracy guarantees and is the only space-efficient solution that offers such capabilities. Finally, we demonstrate how the algorithm can be used to accurately pinpoint the beginning of a realistic DDoS attack.

CCS CONCEPTS

• **Networks** → **Data path algorithms**; **Network algorithms**;

ACM Reference Format:

Nikita Ivkin, Ran Ben Basat, Zaoxing Liu, Gil Einziger, Roy Friedman, and Vladimir Braverman. 2020. I Know What You Did Last Summer: Network Monitoring using Interval Queries. In *ACM SIGMETRICS / International Conference on Measurement and Modeling of Computer Systems (SIGMETRICS '20 Abstracts)*, June 8–12, 2020, Boston, MA, USA. ACM, New York, NY, USA, 2 pages. <https://doi.org/10.1145/3393691.3394193>

1 INTRODUCTION

Network monitoring is at the heart of many networking protocols and network functions, such as traffic engineering, attack and anomaly detection, and forensic analysis. Over the years, a large number of traffic metrics have been defined, such as *heavy hitter detection* [3], *cardinality estimation*, *quantiles* [7], and *entropy estimation* [2]. Given limited memory and computing resources on the network devices, network telemetry systems usually provide approximate results with sufficient accuracy guarantee.

Existing algorithms are designed to estimate the metrics in a predefined, fixed size window with no ability to zoom into any sub-region of the window. This lack of visibility inside the window can fail to capture important events that we want to monitor.

*This work was done while the author was at Johns Hopkins University.

Permission to make digital or hard copies of part or all of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for third-party components of this work must be honored. For all other uses, contact the owner/author(s).

SIGMETRICS '20 Abstracts, June 8–12, 2020, Boston, MA, USA

© 2020 Copyright held by the owner/author(s).

ACM ISBN 978-1-4503-7985-4/20/06.

<https://doi.org/10.1145/3393691.3394193>

Figure 1: The impact of a DDoS attack on the Distinct Count (Green), and the Entropy (Blue) metrics. On the right, the result of our attack pinpointing algorithm for given space allocations.

For instance, an attack detector may wish to identify the beginning and the end of a DDoS attack. To accurately detect such an event, as illustrated in Figure 1, we need efficient algorithms to monitor significant changes in the entropy or number of distinct flows in some small intervals of the measurement window.

The aforementioned attack monitoring problem can be solved in the *Interval Query* (IQ) [4] model, which generalizes *sliding window* to provide visibility to any interval inside the window. In the IQ model, user provides the interval of reference at query time, and the metric is estimated over that interval. It is suitable to accurately pinpoint the beginning of an attack (as in Figure 1) or for identifying short and important patterns. For example, [4] proposes efficient solutions to track L_1 -Heavy Hitters in the IQ model.

In this abstract¹, we present a measurement framework that supports queries with L_2 -guarantee in the IQ model. Pushing the limit to L_2 allows us to estimate important metrics which are useful in attack detection including L_2 -heavy hitters, distinct flows, and entropy. In addition, it enables us to extend general sketches (e.g., UnivMon [9]) to the IQ model. To the best of our knowledge, our work is the first to study the IQ model with L_2 guarantee.

1.1 Interval Queries with L_2 -Guarantee

We seek to identify approximate L_2 heavy hitter flows in the IQ model. That is, at time t , the user issues a query $(t_1, t_2) \subset (t - n, t)$ that returns **all** flows whose packets appear in the interval (t_1, t_2) at least $\epsilon \cdot L_2(t_1, t)$, and **no** flow with frequency less than $\frac{\epsilon}{2} L_2(t_1, t)$. The notation $L_2(t_1, t)$ is the L_2 norm of the interval (t_1, t) .

To design the algorithm, we use Smooth Histogram (SH) technique and adapt state-of-the-art L_2 -heavy hitters sketches [5, 8]. To be concise, we only present the high-level intuition and bounds.

By definition, any flow that is heavy on the interval (t_1, t_2) is also heavy on the suffix (t_1, t) ; however, the reverse logic does not follow. We show that sliding window frameworks (such as SH) can be used to identify heavy hitters on any prefix (t_1, t) , and we use it as a set of *potentially heavy flows* on any interval (t_1, t_2) . For each potentially heavy flow, we run an IQ counting algorithm, in parallel we run IQ

¹For full manuscript we refer reader to [6].

Figure 2: An illustration of the first steps of the attack pinpointing algorithm (with 63MB). In steps 1, 3, and 4, the first unexpected entropy appeared in the last two sub-intervals, allowing the algorithm to prune the first sub-interval. In steps 0 and 2, the algorithm already encounters abnormally high entropy in the second sub-interval and therefore prunes the latest sub-interval. For reference, we annotate the actual starting time of the attack (which is not known to the algorithm).

L_2 norm estimator for the entire stream. The approximate candidate count and L_2 norm on a given interval let us verify if it is heavy on the interval. Our algorithm requires $O(\epsilon^{-3} \log^2 n)$ memory, and works with $O(\log n)$ update time and $O(\epsilon^{-2} \log n)$ query time.

L_2 -guarantee in IQ model paves the route to interval Univmon Sketch [9]. We theoretically extend Univmon to the IQ model and use our L_2 algorithm as a building block towards maintaining a variety of metrics: entropy, cardinality, and others, with only one sketch.

1.2 Attack localization algorithm

We now explore the attack localization capability, which is facilitated by L_2 heavy hitter algorithms in the IQ model. We use the entropy signal, which is useful for detecting a variety of attacks. In addition to attack detection, we also attempt to estimate when the attack started. Our attack localization algorithm periodically estimates the entropy within a sliding window, e.g., once per 1M packets, and maintains the average and standard deviation of the entropy over previous windows. We identify a possible attack when the current entropy estimation is not within five standard deviations from the past entropy. At that point, we begin the attack localization process. It is important to consider the case when attack starts prior the window that raised the alarm, i.e. that it did not build enough impact to trigger the alarm earlier. Therefore, our *Baseline* interval for the start of the attack is the last two windows. Note that any sliding window entropy algorithm (e.g., [2]) reaches the *Baseline* localization.

Next, we attempt to narrow the suspected interval recursively. We partition the current interval (e.g., 4M-6M packets) into four equal parts (e.g., [4M,4.5M], [4.5M,5M], [5M,5.5M], and [5.5M,6M]) and estimate the entropy on each of them. We note that entropy is not linear, and we cannot compare the entropy of a short interval to that of a long interval. Instead, we use the entropy of equal sized subintervals that happened before *Baseline* to define the ‘normal’ entropy. That is, we calculate the mean and standard deviation of such intervals. We call these *safe* intervals as they are before the attack. In this example, we estimate the entropy of [3.5M,4M], [3M,3.5M], etc. We favor the most recent safe intervals as the normal behavior may gradually change, and for algorithmic considerations as the estimation error for these intervals is smaller (see Figure 3).

We set the new suspected interval as a concatenation of all abnormal intervals, and recursively repeat the above process until we can no longer shorten the suspected interval. The operation of the algorithm is illustrated in Figure 2. Specifically, if one of the first two subintervals had unexpectedly high entropy (as in steps 0 and 2), we prune the latest subinterval. Otherwise (as in steps 1,3, and 4), we can safely prune the first subinterval as we expect the entropy to increase after at least one complete attack subinterval.

Figure 3: (a) L_2 -Heavy Hitters and (b) Distribution Entropy Estimation.

2 EVALUATION

We implemented a prototype in C and evaluated the accuracy for various measurements using the CAIDA Internet Trace NY2018 [1].

Entropy and Heavy Hitters: As illustrated in Figure 3, our solution improves with more space and is more accurate on entropy estimation and L_2 -Heavy Hitters for recent intervals.

DDoS attack generation: We simulate an attack containing numerous attackers from 50 randomly picked subnets and multiple victims sharing a single 16-bit subnet, similar to the one considered in [3]. The high number of attackers and the low per-attacker traffic makes attack localization impossible using existing IQ techniques.

Localization results: Figure 1 depicts the actual entropy and distinct count of the stream and the attack start time. In addition, it presents the localization output for different memory allocations. Our results show that (i) the *Baseline* solution using sliding windows is inaccurate for the task; that is, while existing methods may identify the attack they cannot localize it. (ii) Our attack localization successfully localizes the beginning of the attack; and (iii) there is a trade-off between the localization’s accuracy and the allocated memory. More memory allows for better localization.

REFERENCES

- [1] The CAIDA anonymized Internet Traces equinix-nyc 2018-03-15, Dir. A.
- [2] ASSAF, E., BEN-BASAT, R., EINZIGER, G., AND FRIEDMAN, R. Pay for a sliding bloom filter and get counting, distinct elements, and entropy for free. In *INFOCOM 2018*.
- [3] BASAT, R. B., EINZIGER, G., KESLASSY, I., ORDA, A., VARGAFTIK, S., AND WAISBARD, E. Memento: Making sliding windows efficient for heavy hitters. In *CoNEXT 2018*.
- [4] BASAT, R. B., FRIEDMAN, R., AND SHAHOUT, R. Heavy hitters over interval queries. In *PVLDB* (2019). Also available on arXiv:1804.10740.
- [5] BRAVERMAN, V., CHESTNUT, S. R., IVKIN, N., NELSON, J., WANG, Z., AND WOODRUFF, D. P. Bptree: an ℓ_2 heavy hitters algorithm using constant memory. *arXiv:1603.00759*.
- [6] IVKIN, N., BASAT, R. B., LIU, Z., EINZIGER, G., FRIEDMAN, R., AND BRAVERMAN, V. I know what you did last summer: Network monitoring using interval queries. *Proceedings of the ACM on Measurement and Analysis of Computing Systems* (2019).
- [7] IVKIN, N., YU, Z., BRAVERMAN, V., AND JIN, X. Qpipe: Quantiles sketch fully in the data plane. In *ACM CoNEXT* (2019).
- [8] LIU, Z., BEN-BASAT, R., EINZIGER, G., KASSNER, Y., BRAVERMAN, V., FRIEDMAN, R., AND SEKAR, V. Nitrosketch: Robust and general sketch-based monitoring in software switches. In *ACM SIGCOMM* (2019).
- [9] LIU, Z., MANOUSIS, A., VORSANGER, G., SEKAR, V., AND BRAVERMAN, V. One sketch to rule them all: Rethinking network flow monitoring with univmon. In *ACM SIGCOMM* (2016).