A Traffic-Aware Perspective on Network Disaggregated Sketches

Alessandro Cornacchia†, German Sviridov*, Paolo Giaccone*, Andrea Bianco*
Department of Electronics and Telecommunications - Politecnico di Torino, Italy
{†firstname_lastname@polito.it, *firstname.lastname@polito.it}

***Abstract*—Sketches have emerged as a powerful tool for network traffic monitoring due to the good trade-off between accuracy and memory footprint offered by such techniques. Yet, implementing sketches on commercial switches raises numerous challenges related to availability of memory and its access frequency. Recently, disaggregated sketches, i.e., fragments of single network-wide sketches distributed across multiple switches, were introduced to cope with these limitations. However, none of the current approaches exploit any knowledge about the network traffic patterns when deploying such schemes.**

In this paper, we investigate the impact of traffic patterns on the performance of disaggregated sketches. Our findings show that blindly updating all fragments of a sketch might degrade the monitoring accuracy. Instead, taking into account the spatial distribution of the traffic may lead to globally better monitoring accuracy. Finally, we provide hints on the existence of an optimal solution for such a problem which opens new opportunities for the design of traffic-aware update policies for sketches.

I. Introduction

Modern telecommunications networks are characterized by a high degree of dynamicity in traffic patterns. A continuous rolling out of new network-based applications and sudden changes to existing ones have been shown to have a catastrophic impact on the performance of seemingly unrelated services, and on the whole network infrastructure [1]. Due to this unforeseeable behavior, network monitoring has become one of the most important aspects of modern network management. Complex network monitoring mechanisms have been developed to try to predict and counteract those unexpected behaviors. Among those mechanisms, sophisticated centralized network monitoring schemes enabled by breakthrough technologies such as Software Defined Networking (SDN) have found large popularity and applicability. Yet, centralized solutions lack in scalability, as continuously conveying monitoring information from all switches becomes unbearable for large networks. As a consequence, traffic monitoring distributed across the switches has become the dominant best practice for network monitoring.

Still, measurements are challenging to implement due to the huge number of concurrent flows and the ever increasing link rates, which force network devices to complete per-packet operations at nanosecond time scales. This implies resorting to expensive SRAM as the only viable solution to store measurement data. Due to the scarce amount of such dedicated memory, it is prohibitively expensive to keep exact per-flow information locally at each switch. Sampling-based techniques such as NetFlow [2] were traditionally employed to limit resource overhead. However, they are not capable of providing sufficient accuracy and flow coverage, if not adopting extremely high sampling rates [3], [4], [5].

Due to the aforementioned constraints, sketch-based algorithms have found vast applicability in the field of network monitoring. They permit to condense the target flow metrics in compact probabilistic data structures stored inside the switch. This information is then periodically fetched by a central entity and aggregated into a unique, network-wide approximate result. The accuracy of measurements is proportional to the amount of dedicated memory and inversely proportional to the amount of traffic that traverses the switches. While in modern networks the volume of concurrent flows per second keeps growing, the amount of SRAM inside single switches remains constant and it has to be shared among concurrent measurement services and network functions. Under such a scenario, to increase the monitoring accuracy it is necessary to reduce the number of flows stored inside single sketches. A natural option in this case is to increase the sketch fetching frequency, in order to have less flows being condensed in a single sketch within each measuerement interval. Yet such an approach has its own limits which are dictated by the resulting communication overhead and, most importantly, by the underlying hardware capabilities of single switches [6], [7].

Recently, in [8] the authors have proposed DISCO, a system of disaggregated sketches able to address the aforementioned issues. At its heart, DISCO employs multiple sketch fragments scattered around the network which are updated by the flows that traverse them. This enables the possibility of reducing the sketch fetching frequency of each switch without loosing accuracy — or to provide higher accuracy with the same amount of memory — with respect to traditional approaches. Yet DISCO does not consider aspects related to traffic distribution and locality that are critical for improving the overall system performance.

In this paper we analyze the impact of traffic distribution on the performance of disaggregated sketches. We show that, under certain conditions, blindly updating all of the fragments crossed by a flow on its path may lead to measurement performance degradation. We show that, by just selecting a subset of fragments to update it is possible to improve the aggregate monitoring accuracy and we provide a hint on the existence of such an optimal subset.

The rest of the paper is organized as follows. In Sec. II

and Sec. III we provide the background related to probabilistic data structures for network monitoring and discuss the relevant related work. In Sec. IV we discuss disaggregated network-wide sketches and motivate our contribution, which is numerically evaluated in Sec. V. Finally, in Sec. VI, we draw our conclusion.

II. Atomic sketches

Sketches are probabilistic data structures for stream processing, able to estimate flow statistics using a fixed and small number of entries, relatively to the number of processed flows. To distinguish them from the recently proposed disaggregated sketches [8], we refer to them as *atomic sketches*, since not distributed across the network. While a myriad of sketch-based algorithms has been proposed, they all are based on primary components: one or more array of counters, a procedure for their *update* upon new packet arrival and a method to read counters and answer *queries*.

A. Sketch dimensioning

We take the popular Count-Min Sketch (CMS) as an example [9]. When measuring the network traffic, its counters keep track of flows' occurrences in a packet stream. The available memory budget is organized into d rows of w counters each. The *update* procedure consists on computing d pairwise independent hash functions over the flow identifier (e.g., 5-tuple) in order to associate a flow to one counter per row. Then, the selected counters are incremented by one, so as to add the contribution of the new measured packet to the current flow size. Since multiple flows may collide onto the same counter, the *query* operation returns the minimum among d values as its estimate. Generally, sketch-based algorithms come with provable theoretical guarantees and allow us to tune the parameters to trade between estimation accuracy and memory consumption. Indeed, in a CMS it is possible to derive the required number of rows d and of counters w as function of the error magnitude ϵ and the error probability δ. Indeed, if $w = \lceil e/\epsilon \rceil$ and $d = \lceil -\log \delta \rceil$, the estimated size $\hat{x}$ of any flow size x is proven [9] to be within the bound:

$$\hat{x} \leq x + \epsilon \|\mathbf{x}\|_1 \tag{1}$$

with probability greater than $1 - \delta$. Here, x is the true value of the counter, while $\|\mathbf{x}\|_1$ is the $\ell 1$-norm of the flow size vector and it amounts to the total number of packets counted in the whole sketch. A higher number of independent hash functions d are needed in order to reduce the error probability and statistically corresponds to relying upon more estimators, whereas increasing w reduces the error magnitude.

B. Network monitoring with sketches and their limits

Atomic sketch algorithms have been widely used to cope with memory scarcity while employing switch-local traffic monitoring. Traditionally, sketches are deployed on multiple distributed monitors, orchestrated by some controller. Individual monitors inside the network periodically convey their local information to the controller that, together with information from other switches, aggregates and processes data with arbitrary sophisticated statistical methods and retrieve the final measurement [5], [4], [10], [11]. A noteworthy application of sketches is the so called *heavy-hitter detection* which involves the detection of network flows utilizing an eccessive amount of bandwidth. As it is highly impractical, and most often impossible, to gather fine-grained per-flow statistics, CMS found excellent applicability for such kind of task as they permit to discriminate mice and elephant flows up to a given error, while still utilizing small amount of resources.

Yet, as the amount of fast SRAM memory in commodity switches remains constant, the ever-growing number of concurrent flows pushes this kind of approach to its limits, thus making it difficult to maintain an acceptable level of accuracy. In addition, multiple measurement tasks usually execute concurrently and share the available memory. For example, operators may run an heavy-hitter detection task to make intelligent routing decisions and, at the same time, run also anomaly detection tasks to discover the presence of port-scanners or Distributed Deny of Service (DDoS) attacks. These tasks run in parallel and need a dedicated sketch instance [5], [6]. While increasing the reporting frequency may counteract the previously discussed limitations, it has its own drawbacks as it is constrained by the underlying hardware and the generated network overhead.

III. Related work

Sketch algorithms have been employed for various measurement tasks, such as top-k flow identification, traffic entropy, flow size distribution and cardinality estimation, as well as heavy-changer and heavy-hitter detection [7]. The diversity among these tasks required generic data structures to support them concurrently at low complexity, together with efficient communication with the central orchestrator. OpenSketch [5] proposed a software-defined measurement architecture, where a single unified hardware pipeline can be programmed to support a wide range of measurement tasks based on sketches. However, it is too complex to be implemented on recent PISA switch architectures [12]. SCREAM [6] improves the accuracy of distributed monitoring with sketches, by optimizing resource allocation across multiple tasks and multiple switches, based on user requirements. ElasticSketch [10] combines a hash table with a CMS to separate elephant from mice flows and mitigate their collisions, thereby adapting to skewed flow size distributions. Additionally, they first propose a sketch compression technique to reduce communication overhead with the central aggregator. FCMSketch [13] is an elegant tree-like multi-stage counter scheme, which enables the implementation of sketches that support generic measurement tasks directly on PISA switches.

A radically different approach was recently proposed in DISCO [8]. The authors suggested to *disaggregate* a large sketch into multiple smaller sketch *fragments* (i.e., a subset of rows and columns) and distribute them across monitor points in the network. Then, a single sketch is logically rebuilt by assembling fragments encountered along network paths.

Hence, different paths correspond to different logical sketches. For the case of CMS this approach leads to a very simple implementation by which the current minimum across the fragments is piggybacked in the packet headers and the final estimate is obtained by analyzing the packet header at the last-hop. This approach leads to an efficient, yet accurate, heavy-hitter detection scheme while keeping the approach realistic enough to be implemented in real scenarios.

IV. Traffic-aware disaggregated sketches

While DISCO is capable of outperforming atomic sketches, it still adopts a static fragment update policy, meaning that flows are always counted at all fragments along their path.

We argue that, for a given flow, updating all available fragments is often unnecessary and, in some cases, even harmful. Indeed, DISCO update policy has the side effect of generating, otherwise avoidable, collisions (which we will refer to as *counter pollution*) inside single fragments, ultimately degrading heavy-hitter detection accuracy. On the other hand, counting each flow only in one fragment would obviously reduce collisions, but would also impair the accuracy whenever a collision occurs. We show that there exists a trade-off on the number of fragment updates, which balance tolerance to collisions with pollution on sketch counters.

The effects of our observations are exacerbated by non-uniform traffic patterns in the network. In the specific scenario of data center networks, the traffic workload typically presents a non-homogeneous distribution across different network devices [14], with the majority of flows being rack-local. This implies that different switches observe a different load in terms of packets and flows per second. Now blindly updating all of the fragments of a sketch may lead to "overload" fragments (e.g., in top-of-rack switches) and "underutilized" fragments (e.g., in spine switches) and this fact degrades the accuracy of the overall network-wide sketch scheme.

To formalize the considered monitoring scenario, we assume to have a set F of active flows and a network-wide sketch denoted by a set S of fragments distributed in the network. For simplicity, coherently with DISCO, we assume one sketch row (i.e., one hash function) in each switch, but our results can be qualitatively extended to multiple rows per switches. Consider now a flow $f_i \in F$ that traverses a subset $S_i \subseteq S$ fragments along its path. Let k_i^{opt} be the optimal number of fragments to update for f_i, which maximizes the average monitoring accuracy for all flows in F. We will show that it may hold that $k_i^{opt} < |S_i|$ for some $f_i \in F$, i.e., the flow should be not counted in all the fragments. Our goal is to highlight the presence of this phenomenon and to quantify to which extent it may affect the network measurement performance. Indeed, devising a policy capable of selecting the optimal amount of fragments to update and their location across individual flows' path is still to be investigated.

V. Numerical evaluation

In this section, we discuss preliminary results obtained through numerical simulations on the testbed topology

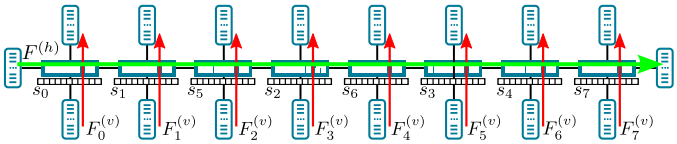


Fig. 1: Bus topology employed for the simulations.

of Fig. 1. Our goal is to show that the accuracy of disaggregated sketches under different workloads depends on the fragment update strategy and traffic patterns.

A. Simulation scenario

For our analysis, we employ a discrete-event packet-based simulator built on top of OMNeT++ [15]. For the sake of clarity, we employ a simple bus topology depicted in Fig. 1. This choice allows us to effortlessly create a scenario highlighting the previously discussed phenomenon of traffic interference. We assume a set of traffic flows equal to $F = \cup_{k=0}^{7} F_k^{(v)} \cup F^{(h)}$, i.e., comprising eight sets of "vertical" flows and one set of "horizontal" flows, as shown in Fig. 1. The horizontal flows in $F^{(h)}$, depicted in blue, updates all or a subset of all the fragments $S = \{s_0, .., s_7\}$ available in the topology. All vertical flows in $F_k^{(v)}$, depicted in red, can exploit only one single monitoring point s_k (i.e., the only switch traversed by them). For all flows in F, we use a heavy-tailed Pareto flow length distribution with shape parameter $\alpha = 1.2$ and mean value 10 packets, to approximate a realistic data center workload [16]. Horizontal and vertical flows are generated according to Poisson processes with $\lambda_h = 100 \times \lambda_v$, respectively. Note that the results are completely invariant with respect to the absolute value of λ_v and on the link capacity and propagation delay, which are assumed homogenous in the whole network. All switches are equipped with sketch fragments consisting of a single row (i.e., $d = 1$) and $w = 2000$ counters.

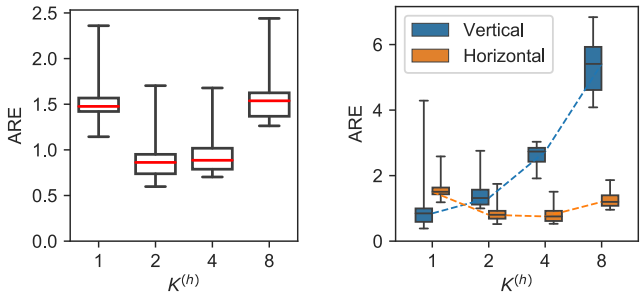
As in prior work [10], [17], we evaluate the performances on the basis of the Average Relative Error (ARE) metric, which is defined as the relative error of the estimated flow size $\hat{x}(f_i)$ with respect to the real flow size $x(f_i)$, averaged over all measured flows:

$$\text{ARE} = \frac{1}{|F|} \sum_{i=1}^{|F|} \frac{\hat{x}(f_i) - x(f_i)}{x(f_i)} \tag{2}$$

Note that in the Count-Min sketch, by construction, $\hat{x}(f_i) \geq x(f_i)$. The relative errors are always computed at flow termination. The choice of ARE with respect to more application-specific metrics, (e.g., false positive rate for heavy-hitter detection), is due to its versatility. Indeed, ARE is agnostic to the definition of an elephant flow and general enough to be amenable for several applications.

B. Simulation results

To highlight the impact of interfering traffic on the monitoring performance of sketch fragments we consider to vary the number of fragments to update $K^{(h)}$ for the horizontal flow. This implies that, out of 8 fragments present along the path,



(a) Overall (b) Horizontal and vertical flows

Fig. 2: Average Relative Error evaluated for different number of fragment updates.

only $K^{(h)}$ will be chosen *randomly* for each $f_i \in F^{(h)}$. For vertical flows instead, we fix $K^{(v)} = 1$ as those flows traverse only one fragment.

Fig. 2 show the ARE in terms of minimum, maximum, 25/50/75-th percentiles, and summarizes the main findings of our simulations. The bars depicted in Fig. 2a represent the overall ARE of aggregate flows, measured for varying $K^{(h)}$, while Fig. 2b presents a more detailed breakdown to distinguish between horizontal and vertical flows. While transitioning from $K^{(h)} = 1$ to $K^{(h)} = 2$, the monitoring accuracy significantly improves, further increasing $K^{(h)}$ to 4 only leads to a slight reduction in the ARE for horizontal flows. Interestingly enough, setting $K^{(h)} = 8$ increases the error for both groups of flows. Thus, in contrast with intuition, the behavior of ARE in function of $K^{(h)}$ does not appear to be monotonic, suggesting us that there exists an optimal value for $K^{(h)}$ in function of the topology and traffic patterns, as discussed in Sec. IV. Indeed, this behavior is due to the fact that, for $K^{(h)} = 8$, all horizontal flows are counted on all fragments present in the topology, thus increasing the pollution on all fragments, ultimately leading to reduced accuracy. On the contrary, the monitoring performance of vertical flows drops considerably while increasing $K^{(h)}$, with their ARE increasing by a factor of 4.85 when moving from $K^{(h)} = 1$ to $K^{(h)} = 8$. Yet, when taking into account the overall monitoring accuracy that includes both vertical and horizontal flows, it can be seen that employing $K^{(h)} = 2$ leads to better aggregate monitoring accuracy across all flows.

VI. Conclusion

In this paper, we highlight the effect of traffic patterns on the performance of disaggregated sketches. In particular, we show that blind fragment update policies which force a flow to update all of the fragments along its path may not necessary lead to the best overall monitoring performance. Through numerical simulation and under a simplistic testbed scenario, we show that there should exist an optimal fragment update policy that is capable of selecting a subset of fragments to update among those available on the path of individual flows. Such policy must operate on the network traffic pattern and take into account the total traffic traversing individual fragments. Further investigation of this behavior, alongside more complex experimental scenarios, is left as future work.

References

[1] M. Trevisan, D. Giordano, I. Drago, M. M. Munafò, and M. Mellia, "Five years at the edge: Watching Internet from the ISP network," *IEEE/ACM Transactions on Networking*, vol. 28, no. 2, 2020.

[2] "NetFlow," https://tools.ietf.org/html/rfc3954, accessed: 2021-03-14.

[3] C. Estan and G. Varghese, "New directions in traffic measurement and accounting," in *Proceedings of the Conference on Applications, technologies, architectures, and protocols for computer communications*, 2002.

[4] Y. Li, R. Miao, C. Kim, and M. Yu, "Flowradar: A better netflow for data centers," in *USENIX NSDI*, 2016.

[5] M. Yu, L. Jose, and R. Miao, "Software defined traffic measurement with OpenSketch," in *USENIX NSDI*, 2013.

[6] M. Moshref, M. Yu, R. Govindan, and A. Vahdat, "Scream: Sketch resource allocation for software-defined measurement," in *ACM Conference on Emerging Networking Experiments and Technologies*, 2015.

[7] S. Li, L. Luo, and D. Guo, "Sketch for traffic measurement: design, optimization, application and implementation," *arXiv preprint arXiv:2012.07214*, 2020.

[8] V. Bruschi, R. B. Basat, Z. Liu, G. Antichi, G. Bianchi, and M. Mitzenmacher, "DISCOvering the heavy hitters with disaggregated sketches," in *International Conference on emerging Networking EXperiments and Technologies*, 2020.

[9] G. Cormode and S. Muthukrishnan, "An improved data stream summary: the count-min sketch and its applications," *Journal of Algorithms*, vol. 55, no. 1, 2005.

[10] T. Yang, J. Jiang, P. Liu, Q. Huang, J. Gong, Y. Zhou, R. Miao, X. Li, and S. Uhlig, "Elastic sketch: Adaptive and fast network-wide measurements," in *Conference of the ACM Special Interest Group on Data Communication*, 2018.

[11] Q. Huang, P. P. Lee, and Y. Bao, "Sketchlearn: relieving user burdens in approximate measurement with automated statistical inference," in *Conference of the ACM Special Interest Group on Data Communication*, 2018.

[12] "Portable Switch Architecture," https://p4.org/p4-spec/docs/PSA-v1.0.0.pdf, accessed: 2021-03-15.

[13] C. H. Song, P. G. Kannan, B. K. H. Low, and M. C. Chan, "FCM-sketch: generic network measurements with data plane support," in *International Conference on emerging Networking EXperiments and Technologies*, 2020.

[14] T. Benson, A. Akella, and D. A. Maltz, "Network traffic characteristics of data centers in the wild," in *ACM SIGCOMM Conference on Internet Measurement*, 2010.

[15] A. Varga, "OMNeT++," in *Modeling and tools for network simulation*. Springer, 2010.

[16] A. Roy, H. Zeng, J. Bagga, G. Porter, and A. C. Snoeren, "Inside the social network's (datacenter) network," in *ACM Conference on Special Interest Group on Data Communication*, 2015.

[17] T. Yang, L. Wang, Y. Shen, M. Shahzad, Q. Huang, X. Jiang, K. Tan, and X. Li, "Empowering sketches with machine learning for network measurements," in *Workshop on Network Meets AI & ML*, 2018.