BatchSketch: A "Network–server" Aligned Solution for Efficient Mobile Edge Network Sketching

Wendi Feng
BISTU
China
wendifeng@bistu.edu.cn

Chuanchang Liu
BUPT
China
lcc3265@bupt.edu.cn

Junliang Chen
BUPT
China
chjl@bupt.edu.cn

Abstract

Heavy flow identification is essential for discovering potential adversarial activities in mobile edge networks. However, state-of-the-art falls short in accuracy with approximation algorithms and unbounded memory usages with precise measurement. To this end, we introduce BatchSketch, a "network – server" aligned solution to achieve both high accuracy and low memory usage. The intelligence behind is that BatchSketch first conducts *coarse-grained* filtering from the switch with bounded memory and computation resources, and it then sends the filtered flows to the RDMA-link attached server with plenty of memory for accurate measurement. Our primary experimental results indicate that the *filter* on the switch can filter 99% of non-heavy flows, remarkably reducing the memory usage for the measurement.

Keywords

Mobile edge network, network measurement, sketching

ACM Reference Format:

Wendi Feng, Chuanchang Liu, and Junliang Chen. 2022. BatchSketch: A "Network–server" Aligned Solution for Efficient Mobile Edge Network Sketching. In *The 28th Annual International Conference on Mobile Computing and Networking (ACM MobiCom '22), October 17–21, 2022, Sydney, NSW, Australia.* ACM, New York, NY, USA, 3 pages. https://doi.org/10.1145/3495243.3558246

1 Introduction

Mobile edge computing (MEC), bringing services close to the users, is a promising architecture for the next-generation low-latency mobile communication networks like 6G [7]. However, closing to users means limited real estate and power resources, resulting in weaker network and computation capabilities [4, 8] and making it more vulnerable to malicious behaviors. Thereby, identifying adversary behavior is critical to MEC, in which network traffic measurement is an important yet essential technique to identify potential malicious behaviors from certain traffic patterns, *e.g.*, *top-flows* [10] (*i.e.*, elephant flows), *heavy hitters* [9] (*i.e.*, flows generating huge amounts of data in a short period of time), and *spreaders* [6] (*i.e.*, hosts initiating a large number of distinctive connections in a small duration), without disrupting network services.

However, detecting heavy flows (*e.g.*, top-flows, heavy hitters, and spreaders) in practice is challenging because **i)** the traffic volume is gigantic and can continuously increase over time; and **ii)** the detection should tolerate the worst-case (*e.g.*, when bursts and attacks arrive). For example, detecting on a fully used modern 100 Gbps link with 64 B[1] Ethernet packets indicates a 148 Mpps packet processing throughput. Besides, the available memory resources on in-network devices (*e.g.*, switches) are limited. Hence, simply using a flow table to record/count packets for each flow results in unbounded memory usage (proportional to the number of flows) and is considered infeasible for real-world million-flow detection.

Under such rigid constraints, *sketches* summarize network measurement metrics with a *summary data structure* that tracks the metric value in a fixed number of *Buckets*. Take CM-Sketch [3] as an example. It employs d rows of ω buckets, and each row uses a dedicated hash function to record one element (*e.g.*, a packet) in a bucket that has a counter initialized as 0. When a packet arrives, its flow identification fields[2] are extracted and calculated using hash functions as the keys. A key is mapped to a bucket in each row, and the bucket increases its counter by 1. Since multiple flows can hit the same bucket making the counters may be larger than the flow's actual size, CM-sketch uses the minimum count across all rows as the final estimated measurement result.

State-of-the-art sketches run merely on a programmable switch [9] or general-purpose commodity server [5, 6, 11], failing to achieve optimal performance (*i.e.*, throughput) due to the limited memory on the switch and impaired packet processing capability on the server, respectively. Our observation is that both platforms have unreplaceable benefits:

ACM MobiCom '22, October 17–21, 2022, Sydney, NSW, Australia

ACM ISBN 978-1-4503-9181-8/22/10.
https://doi.org/10.1145/3495243.3558246

[1] The minimum size of Ethernet packets

[2] For example, the identification fields of a 5-tuple flow is $\langle srcIP, dstIP, srcPort, dstPort, proto \rangle$.

the switch can attain peak packet processing performance, whereas the server has plenty of memory resources to utilize. Besides, buffering a fixed number of packets in a *batch* and processing (*e.g.*, received, transmitted, updated) them simultaneously is a widely used technique for achieving higher throughputs. Hence, the question is: *can we achieve fast yet precise network measurements that benefit from the ultimate performance of programmable switch platforms and the large memory resources of the general-purpose commodity server along with the batch mechanism?*

Motivated by the batching technique and the status-quo of sketching systems, we present BatchSketch. It leverages the batching mechanism commonly utilized in packet processing along with the fast packet processing capabilities introduced by switches and large memory resources delivered by commodity servers. The intelligence behind BatchSketch is that **i)** it first pre-counts the flow numbers in each received packet batch and finds potential heavy flows[3] from within on a programmable switch[4]. **ii)** It then sends the potential heavy flows to the attached general-purpose commodity server capable of using more memory footprints. Moreover, **iii)** Remote Direct Memory Access (RDMA) is employed when transmitting packets between the programmable switch and the server to mitigate processing latencies. Hence, BatchSketch is a "network – server" aligned solution, in which we leverage the benefits of both commodity servers (plenty of memory resources) and programmable hardware (high-performance). To summarize, the contribution of this paper is three-fold as follows.

- We employ a "network – server" aligned solution called BatchSketch to measure network traffic efficiently.
- We introduce a batch-based pre-filtering technique to reduce memory usage and improve performance using programmable switches.
- Experimental results show that the programmable switch can pre-filter over 99% of non-heavy flows.

2 BatchSketch Design

2.1 BatchSketch from a Bird's Eyes View

To simultaneously achieve fast and precise network measurement with bounded memory usage, we present a "network – server" aligned approach called BatchSketch. It utilizes a greedy strategy that first employs the high-performance programmable switch to find "potential" heavy flows in each received batch of packets, and it then sends the selected packets to the server to "enjoy" plenty of memory resources for finer-grained measurements.

As depicted in Figure 1, BatchSketch consists of two parts. The *pre-filter* and the *sketcher*. The pre-filter runs on a

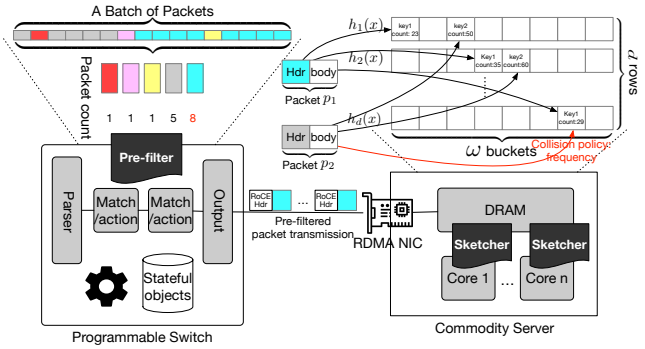


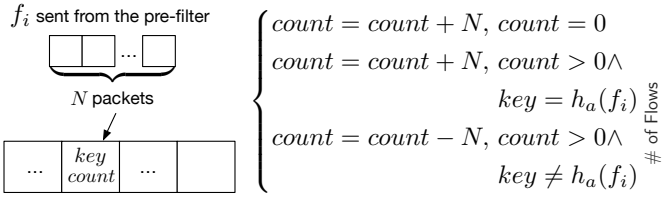
Figure 1: Overview of BatchSketch.

programmable switch, and the sketcher runs on a general-purpose commodity server. When the programmable switch receives packets in each batch, the pre-filter identifies different flows and counts their packet number with a match-action table. The pre-filter uses the exact match strategy and flushes the match-action table after receiving n *batch epochs* (depending on the match-action table size) to avoid collisions. Then, the heaviest flow (*i.e.*, the maximum count flow) is filtered out as a "potential" heavy flow. The filtered flow is sent to the commodity server for sketching via the RDMA connection between the switch and server to avoid transmission overhead introduced by the server operating system [1]. Besides, the pre-filter only sends flow identifiers retrieved from packet headers and the counter of packet numbers recorded by the pre-filter to further reduce the transmission overhead. RDMA connections rely on high-performance RDMA NICs deployed on each side of the communication entities. The commodity server is easy to facilitate, but installing an RDMA NIC on a programmable switch is impossible. To this end, we generate RoCEv2 packets directly[5] from the programmable switch, thanks to its *programmability*. Therefore, the "potential" heavy flow packets can be directly DMA'ed to the sketcher server's user space memory.

2.2 Data Structure

The data structure used for the pre-filter on the programmable switch is a match-action table with a fix-sized length len. len is capable of recording the number of flows of n batch epochs of packets mentioned above. This match-action table can match flows and count their packet number precisely. To avoid consuming too much memory or overflowing the match-action table, the pre-filter flushes table entries after n batch epochs. Since the batch size is usually a constant, the time complexity for this process is $O(n)$. On the server-side, the sketcher maintains a data structure similar to BurstSketch [11], and each bucket contains a *key* and a *counter* that is slightly different from the classical CM-Sketch [3]. The

[3]For example, top flows, heavy hitters, spreaders.
[4]This procedure can run on a SmartNIC, such that **Step iii)** is not needed.
[5]Programmable switches are very good at modifying packet headers, which allows us to build RoCEv2 packets directly.



$$\begin{cases} count = count + N,\ count = 0 \\ count = count + N,\ count > 0 \wedge \\ \qquad\qquad key = h_a(f_i) \\ count = count - N,\ count > 0 \wedge \\ \qquad\qquad key \neq h_a(f_i) \end{cases}$$

Figure 2: Heavy flow detection on the server.

sketcher employs the *frequency* [11] strategy to deal with collisions on each row detailed in Section 2.3.

2.3 Detection Example: Heavy Hitter

The pre-filter process is straightforward, and it can be proved that heavy flows must be in the filtered flows. We omit the proof due to space limitations. Assume a flow f_i is pre-filtered by the programmable switch and transmitted to the server. As depicted in Figure 2, the sketcher first passes f_i's identifier to the hash function of each row and mods the hash value with ω to index the bucket. If the bucket is empty, the sketcher directly stores the key (hash value) and the counted packet number from the pre-filter. If not, the sketcher determines whether the current key in the bucket is the same as f_i's key. If yes, the counter increments by N and decreases by N if otherwise. When retrieving the estimated value of flow f_i, the maximum count across rows is selected. Then if the estimated count exceeds a predefined threshold value T, flow f_i is considered a heavy flow. This process can report heavy flow with a probability of $1 - e^{-d}$ with proof, where e is the Euler number, and d is the number of rows of the sketcher.

3 Primary Experimental Results

This section shows our primary experimental results of the number of flows filtered by the pre-filter. We use a clip of the CAIDA anonymized Internet traces dataset [2] to conduct the experiments. As illustrated in Figure 3, the pre-filtering process can significantly reduce the number of flows sent to the server for further measurement (as low as 0.93% of all flows), which decreases the input scale of the sketcher. The two figures show same experimental results but in different metrics. In the left figure, the number of flows filtered by the pre-filter (*i.e.*, sent to the server for sketching) is significantly reduced as the batch size increases. Similarly, when n (detailed in Section 2) increases, the number of filtered flows also reduces in the right figure. Our experimental results demonstrate that the pre-filter is efficient, and we believe the sketcher can minimize the memory usage on the commodity server that benefits from the pre-filtering.

4 Conclusion and Future Work

We present BatchSketch, a "network – server" aligned solution for mobile edge network measurement. It employs a two-stage mechanism that first leverages a greedy strategy to pre-filter potential heavy flows from each received packet

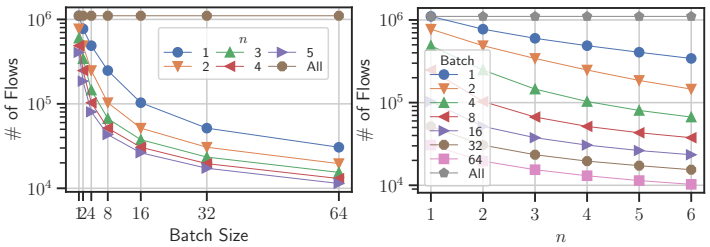


Figure 3: Number of filtered flows as the increase of batch size and the number of epochs n.

batch directly at the programmable switch. It then transmits the filtered packets to the commodity server, which allows sketches to "enjoy" plenty of memory resources for finer-grained measurement. To reduce data transmission overheads between the switch and the server, an RDMA link is created by directly generating RoCEv2 packets from the programmable switch. Our primary experimental results show that the pre-filter can filter over 99% of non-heavy flows, remarkably reducing the memory usage on the server.

This work is currently in an early stage. Our next step is to systematically evaluate memory reduction achieved by the pre-filter and accuracy affected by the batch size and number of epochs. We will conduct rigorous theoretical analyses for the pre-filter, sketcher, and overall result with proven tighter bounds. We present BatchSketch to call for broad comments and to inspire the network measurement community to solve the problem from a systematic approach.

Acknowledgments

This work is supported by BISTU Research Fund Under Grant 2022XJJ19. We thank the anonymous reviewers for their valuable comments.

References

[1] Qizhe Cai, et al. 2021. Understanding host network stack overheads. In *SIGCOMM*. 65–77.
[2] CAIDA. 2022. The CAIDA UCSD Anonymized Internet Traces. https://www.caida.org/catalog/datasets/passive_dataset
[3] Graham Cormode et al. 2005. An improved data stream summary: the count-min sketch and its applications. *JALGO* 55, 1 (2005), 58–75.
[4] Wendi Feng, et al. 2022. STATOEUVER: State-aware Load Balancing for Network Function Virtualization. In *INFOCOM WKSHPS*. 1–2.
[5] Lu Tang, et al. 2019. Mv-sketch: A fast and compact invertible sketch for heavy flow detection in network data streams. In *INFOCOM*.
[6] Lu Tang, et al. 2020. Spreadsketch: Toward invertible and network-wide detection of superspreaders. In *INFOCOM*. 1608–1617.
[7] Harsh Tataria, et al. 2021. 6G wireless systems: Vision, requirements, challenges, insights, and opportunities. *Proc. of the IEEE* 109, 7 (2021).
[8] Ziyan Wu, et al. 2022. NFlow and MVT Abstractions for NFV Scaling. In *INFOCOM*. 180–189.
[9] Tong Yang, et al. 2018. Elastic sketch: Adaptive and fast network-wide measurements. In *SIGCOMM*. 561–575.
[10] Tong Yang, et al. 2019. HeavyKeeper: An Accurate Algorithm for Finding Top-k Elephant Flows. *TON* 27, 5 (2019), 1845–1858.
[11] Zheng Zhong, et al. 2021. Burstsketch: Finding bursts in data streams. In *SIGMOD*. 2375–2383.