APS: Adaptive Priority-aware Sketch with Low Overhead for Heterogeneous Traffic

Jin Wang , Houqiang Shen, Yusheng Deng , Jing Wang and Jinbin Hu

Changsha University of Science and Technology, Changsha 410114, China
jinwang@csuct.edu.cn

Abstract. With the emergence of software-defined networking, sketch-based network measurements have been widely used to balance the tradeoff between efficiency and reliability. Recently, a class of priority-aware sketches has been developed to provide differentiated measurement accuracy for flows with different priorities. However, as the number of priorities increases, the overhead of these priority-aware sketches also increases, resulting in poor performance in scenarios with a high number of priorities. In this paper, we propose an adaptive priority sketch called Adaptive Priority-aware Sketch (APS) with low overhead, which utilizes priority-aware hashing to dynamically allocate a sufficient number of hash functions based on the priority of different flows. Experimental results show that APS improves throughput by up to 110% under the scenarios with large number of high-priority flows, while maintaining similar overall measurement accuracy under medium and low priority flow scenarios.

Keywords: Sketch, Priority-aware, Network Measurement.

1 Introduction

With the advent of the digital age, network communication has become an indispensable part of our daily lives and work. To meet the growing demand for networks, Software-Defined Networking (SDN) has emerged as a network architecture that separates the network control plane from the data forwarding plane, making network management and control more flexible and programmable. The emergence of SDN has made network measurement and monitoring even more important, and network measurement is widely used in various fields such as load balancing, network slicing, congestion control, traffic scheduling, and network security [1-2]. Network measurement can help identify network bottlenecks and performance issues, evaluate network quality and reliability, and assist in network planning and resource allocation.

In network measurement, Sketch algorithms are commonly used. They calculate hashes for the five-tuple of traversed flows and record them in corresponding counters, achieving a good balance between accuracy, speed, and memory usage. However, most existing optimizations for Sketch algorithms focus on new storage structures or counter update strategies, without considering differentiated services for

flows with different priorities. Since high-priority flows are fewer in number but carry more important information, it is reasonable to prioritize the monitoring of these flows.

Recently, priority-based adaptive sketches have been proposed to address the issue of priority-aware traffic monitoring. These sketches provide differentiated detection accuracy for flows with different priorities while maintaining low memory usage, low computational overhead, and high throughput. Existing priority-aware sketches can be categorized into two types: MC-Sketch [3] and Cuckoo-Sketch [4], which prioritize high-priority flows by repeatedly evicting low-priority flows from buckets, and PA-Sketch [5], which linearly allocates the number of hash functions based on priorities. PA-Sketch achieves higher throughput at the cost of sacrificing some accuracy for low-priority flows compared to the former two.

To address this issue, we propose a new solution called APS(Adaptive Priority-aware Sketch), which achieves high performance in scenarios with a large number of high-priority flows. The core idea of APS is to allocate an appropriate number of hash functions to each flow based on the Zipf distribution when there are multiple priorities. This effectively avoids wasting hashing computation resources when the number of priorities is large. APS achieves significantly faster speeds while maintaining a certain level of accuracy compared to other priority-aware methods.

The rest of this paper is organized as follows: In Section 2, we describe the motivation. In Section 3, we describe our design. In Section 4, we show the experiment results. In Section 5, we conclude the paper.

2 Motivation

Existing priority-aware sketches can be categorized into two types: one represented by Cuckoo-Sketch, which focuses more on overall accuracy, and another represented by PA-Sketch, which emphasizes memory saving and speed. However, all previous priority-aware sketches have only considered scenarios with a small number of priorities, typically around 8. When the number of priorities increases, the performance of some priority-aware sketches can significantly change.

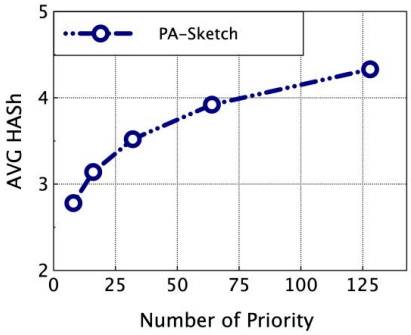

Fig. 1. The average number of hash calculations .

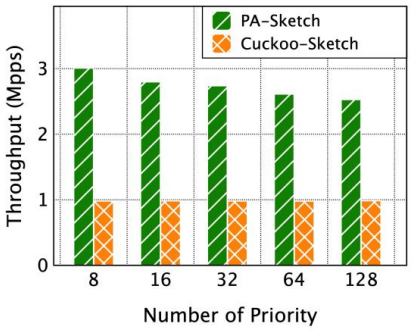

Fig. 2. Throughput

To validate the performance of PA-Sketch and Cuckoo-Sketch under high priority count conditions, we set the priority range to 8-128 and conducted experiments. 128 is the maximum number of priorities commonly found in servers in the market [6-7].

We conduct statistics on the average number of hash calculations for each packet in PA-Sketch, and the results are shown in Figure 1. The average computational overhead of PA-Sketch increases dramatically with the increase in the number of priority levels, confirming our previous hypothesis. Figure 2 shows the throughput performance of Cuckoo-Sketch and PA-Sketch under different numbers of priorities. The throughput of Cuckoo-Sketch remains consistently low regardless of the number of priority levels, while the throughput of PA-Sketch exhibits a noticeable decrease as the number of priority levels increases. This can be attributed to the fact that Cuckoo-Sketch consumes a significant amount of hash computation resources due to the repeated eviction process of a large portion of low-priority flows. In addition, as the number of priority levels increases, PA-Sketch allocates excessive unnecessary hash functions for medium and low-priority flows, resulting in more collisions. Therefore, we conclude that it is hard for existing priority-aware sketches to achieve sufficient throughput when the number of priority levels is high.

3 Experimental Design

3.1 Design Overview

APS consists of two parts: the PRIOR layer and the CM-Sketch, as shown in Figure 3. The PRIOR layer stores potential high-priority flows, while the CM-Sketch provides estimates for evicted low-priority flows. Each element in the PRIOR layer, represented by A(i), includes A(i).COU for cumulative count, A(i).PRI for priority, and A(i).ID for flow key. When an element is evicted, the ID is mapped to the CM-Sketch. APS also includes the hash allocation function fen(), which initializes based on the maximum priority (MAX_priority) to allocate hash functions for insertion and query processes.

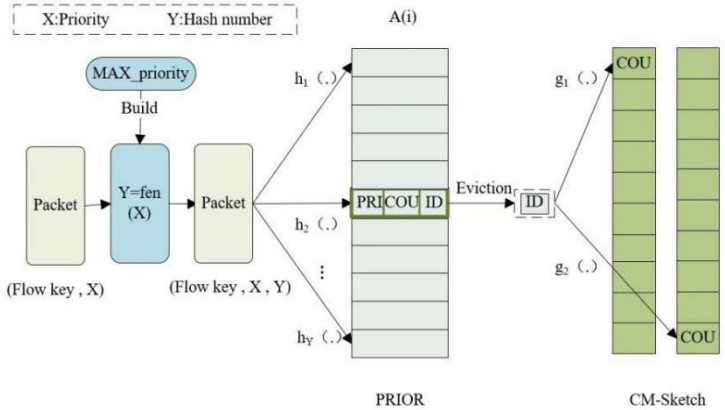

Fig. 3. Structure of the APS.

3.2 Design Details

Construction of Hash Allocation Functions. When APS starts running, it first reads the maximum number of priority levels, MAX_priority, for the flows being processed. APS then constructs an array of size MAX_priority and assigns a number to each element of the array using a series of calculations. This allocation of numbers follows a zipf distribution:$f(x) = \frac{1}{x^{\alpha} \sum_{i=1}^{\gamma} (1/i)^{\alpha}}$, where α is the skewness parameter of the zipf distribution and γ is the number of priority levels. In our case, we set α = 2.5 and γ = MAX_priority.

Subsequently, APS uses this array to construct the hash allocation function fen(). Given priority X as input, the fen() function will output the number of allocated hash functions Y. This means that the allocation of hash functions will be skewed towards high-priority flows.

Insertion. After the hash allocation function is constructed, all fields in APS are set to 0 or NULL. When APS receives a new incoming data packet P, it first reads the priority X of P and calculates the number of hash functions Y to be allocated to data packet P using the hash allocation function fen(). Then, APS maps data packet P to Y buckets in the PRIOR layer. The following three situations may occur:

Case 1: When at least one empty bucket exists among the Y positions to which P is mapped, the information of P will be directly stored in an empty bucket. The PRI of this bucket will directly record the priority of P, the ID will be recorded as the flow key of P, and the counter COU will become 1.

Case 2: When there are no empty buckets among the Y positions to which P is mapped, APS compares the flow key of P with the IDs recorded in the Y buckets. If there is a bucket with the same ID as the flow key of P, the counter COU of that bucket is incremented by 1.

Case 3: When there are no empty buckets or buckets with the same ID as P's flow key, APS compares priorities to find the bucket with the lowest priority. If P's priority is higher than the current bucket's priority, P replaces the information in that bucket, the elements in the original bucket will be evicted to the CM-Sketch. If P's priority is lower, P is evicted to the CM-Sketch.

Query. When querying a flow f, APS first uses the hash allocation function fen() to calculate the number of hash function allocations, Y. Then, it checks whether there is a bucket among the Y buckets in the PRIOR layer that records the flow key of f. If it exists, the counter COU recorded in this bucket is directly returned as the result. If it does not exist, APS queries the bucket to which f is mapped in the CM-Sketch and returns the minimum value in that bucket.

4 Experimental Results and Analysis

4.1 Experimental Setup

Datasets. We use an anonymous data set from CAIDA 2019 comprising 18 million packets and 1 million flows. Priorities are assigned to flows based on their frequencies, following a zipf distribution that reflects real-world scenarios. The majority of flows are assigned low priorities, while a small portion receives high priorities.

Platform. Our experiments are conducted on a server with a quad-core CPU (Intel(R) Core(TM) i5-8265U CPU @ 1.60GHz), which supports both 32-bit and 64-bit operating modes. Each core has three levels of cache: a 128KB L1d cache, a 128KB L1i cache, a 1MB L2 cache, and a 12MB L3 cache. The operating system used is ubuntu-22.04.3.

Evaluation Metrics. We use throughput as the evaluation metric for speed and Average Relative Error (ARE) as the evaluation metric for accuracy.The throughput is calculated as the total number of packets processed divided by the processing time. We measure throughput in Million packets per second (Mpps).The expression for Average Relative Error (ARE) is $\frac{1}{n}\sum_{i=1}^{n}\frac{|\hat{f}_i - f_i|}{f_i}$, where f_i represents the true size of each flow and $\hat{f}_i$ represents the estimated size of each flow.

4.2 Experimental Results

Average ARE for high priority flows: We evaluate the average ARE for high-priority flows in three sketches under different numbers of priority levels. Figure 4 shows that Cuckoo-Sketch consistently maintains a lower ARE for high-priority flows. APS initially has higher ARE for high-priority flows with a small number of priorities, but as the number of priorities increases, APS demonstrates a decreasing trend in ARE, surpassing PA-Sketch.

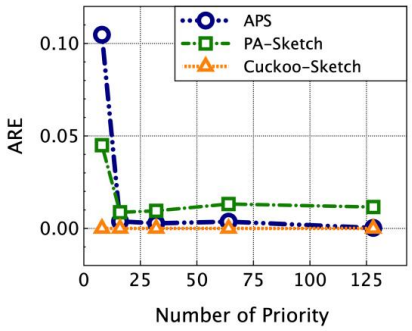

Fig. 4. Average ARE for high priority flows

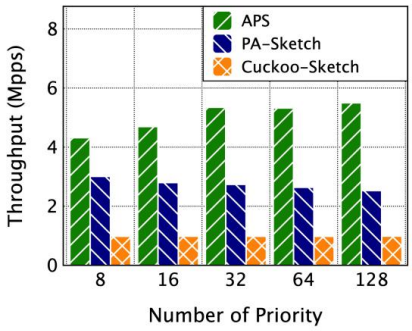

Fig. 5. Throughput

Throughput: We calculate the throughput for the three sketches under different number of priority levels. Figure 5 shows the throughput comparison of three sketches under different priority levels. Cuckoo-Sketch consistently has lower throughput, while PA-Sketch's throughput gradually decreases as the number of priority levels increases. In contrast, APS exhibits an increasing trend in throughput with more priority levels. Overall, the throughput pattern is APS > PA-Sketch > Cuckoo-Sketch.

5 Conclusion

In this article, we propose APS, a priority-aware Sketch optimized for scenarios with a large number of high-priority flows. APS achieves differentiated treatment of high and low-priority flows through differential hash function allocation and eviction strategies. Experimental results demonstrate that APS outperforms existing priority-aware sketches, achieving a throughput improvement of 90%-110% in high-priority scenarios while maintaining similar measurement accuracy. This validates APS as a superior solution for scenarios with a large number of priorities.

References

1. Y. Li, R. Miao, H. H. Liu, Y. Zhuang, F. Feng, L. Tang, Z. Cao,M. Zhang, F. Kelly, M. Alizadeh, M. Yu: HPCC: High precision congestion control. In: Proceedings of ACM conference on SIGCOMM, pp. 44-58 (2019).
2. Z. Liu, Z. Bai, Z. Liu, X. Li, C. Kim, V. Braverman, X. Jin, I. Stoica: DistCache: Provable Load Balancing for Large-Scale Storage Systems with Distributed Caching. In: Proceedings of USENIX conference on FAST, pp. 143-157 (2019).
3. K. C. -J. Lin, W. -L. Lai: MC-Sketch: Enabling Heterogeneous Network Monitoring Resolutions with Multi-Class Sketch. In:Proceedings of IEEE conference on INFOCOM, pp. 220-229 (2022).
4. Y. Yan, C. Chen, H. Lin, O. Ruas, T. Wang, T. Yang: Priority-Aware Per-flow Measurement using Cuckoo Sketch. In: Proceedings of IFIP conference on Networking, pp. 622-624 (2020).
5. S. Li, J. Huang, W. Zhang, J. Shao, "PA-Sketch: A Fast and Accurate Sketch for Differentiated Flow Estimation," In:Proceedings of the IEEE International Conference on Network Protocols (ICNP), 2023.
6. P. Goyal, P. Shah, K. Zhao, G. Nikolaidis, M. Alizadeh, T. E. Anderson: Backpressure flow control. In: Proceedings of 19th USENIX Symposium on Networked Systems Design and Implementation (2022), pp. 779–805.
7. W. Li, C. Zeng, J. Hu, K. Chen, "Towards Fine-grained and Practical Flow Control for Datacenter Networks," In:Proceedings of the IEEE International Conference on Network Protocols (ICNP), 2023.