

Said Jawad Saidi, Damien Foucard, Georgios Smaragdakis, Anja Feldmann

Flowtree: Enabling Distributed Flow Summarization at Scale

Conference paper | Accepted manuscript (Postprint)

This version is available at <https://doi.org/10.14279/depositonnce-9376>



© ACM 2018. This is the author's version of the work. It is posted here for your personal use. Not for redistribution. The definitive Version of Record was published in ACM SIGCOMM 2018, <http://dx.doi.org/10.1145/3234200.3234225>.

Saidi, S. J., Foucard, D., Smaragdakis, G., & Feldmann, A. (2018). Flowtree. Proceedings of the ACM SIGCOMM 2018 Conference on Posters and Demos - SIGCOMM '18. <https://doi.org/10.1145/3234200.3234225>

Terms of Use

Copyright applies. A non-exclusive, non-transferable and limited right to use is granted. This document is intended solely for personal, non-commercial use.

WISSEN IM ZENTRUM
UNIVERSITÄTSBIBLIOTHEK

Technische
Universität
Berlin

Flowtree: Enabling Distributed Flow Summarization at Scale

Said Jawad Saidi
MPI-Informatics

Damien Foucard
TU Berlin

Georgios Smaragdakis
TU Berlin / MIT

Anja Feldmann
MPI-Informatics / UDS

ABSTRACT

NetFlow and IPFIX raw flow captures are insightful yet, due to their large volume, challenging to timely analyze and query. In particular, if these captures span long time periods or are collected at remote locations, storing or transferring them for analysis becomes increasingly expensive.

Enabling efficient execution of a large range of queries over flow captures while reducing storage and transfer volume requires working with mergeable succinct summaries that capture the most essential features of flows dynamically. However, the problem of building such structures is yet unmet. In this work, we introduce a self-adjusting data structure of generalized flows, called *Flowtree*, that (1) reduces the storage requirements by more than 95% while providing highly accurate answers for popular hierarchical flows, (2) minimizes transfer cost of flow summaries, and (3) supports several operators with distributed execution and summarization across time and multiple sites. The evaluation of our solution on different network traces confirms that Flowtree can accurately and promptly answer questions about flows using different feature sets.

KEYWORDS

Network Monitoring, Flow Summarization

1 INTRODUCTION

Network operators analyze IPFIX or NetFlow flow captures to glean insights about state, security, and performance of their networks. However, promptly analyzing and querying large amounts of flow captures becomes increasingly and prohibitively expensive in a wide area network. Consider Fig. 1: ISP operators want to know, in the last 24 hours, what is the total volume of traffic sent by one of its peers to all of five ISPs' sites. In the meantime, they notice that IP address range X/8 has received a lot of traffic, they want to know if it is due to a specific IP, a specific /24, or what is happening. Moreover, they want to know this for any of the flow features: src IP, dst IP, ports, and protocols. Answering such queries using raw captures can be expensive due to a large volume of flow captures. Moreover, transferring captures between sites can be forbidden due to different regulatory jurisdictions. All these underline the need for *online indexing* of flows on top of existing captures, a way to maintain *succinct (space-efficient)*

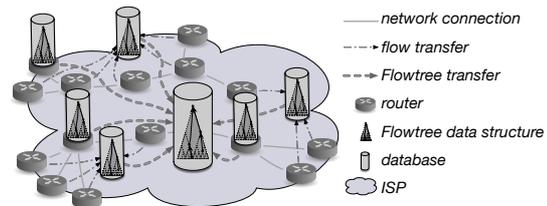


Figure 1: Flow summarization system with Flowtree.

flow summaries, and having the ability to drill down into specific events to perform in-depth analysis.

Working with succinct summaries of flows can reduce the storage and transfer cost. However, in order to allow efficient execution of a large range of queries, there is a need for a flow summary data structure that (a) captures the most essential features of network flows (b) efficiently allows for several operations, i.e., merge, diff with other summaries at different granularity levels. *Mergeable* flow summaries can reduce transfer and storage volume by allowing transfer of only summaries or even difference of consecutive summaries.

Nevertheless, the community has not agreed on any appropriate summary yet. Existing work in flow summarization is either relied on pre-installed rules [4] or concerned with capturing heavy hitters in tree-like structures [1–3, 5]. Keeping summaries of only the most popular flows misses information on less popular ones. Moreover, these approaches focus on computing summaries at a single router, which is not enough for a multi-site environment and working on existing captures. Therefore, our summaries need to go beyond capturing heavy hitters and capture non-popular flows as well while keeping the space usage under constraint.

Hence, we propose our novel self-adjusting data structure called Flowtree, for on-the-fly processing of flows and generating summaries that can be stored. Flowtree dynamically tracks the essential features of the input stream, by keeping the popular flows and summarizing the less-popular ones. It also allows efficient execution of wide range of queries in a distributed fashion, by providing several operators, i.e. query, merge, and diff.

2 FLOWTREE DATA STRUCTURE

Flowtree is a self-adjusting data structure that tracks the essential features of the input stream, i.e., capturing highly popular flows while summarizing less popular ones. Flows

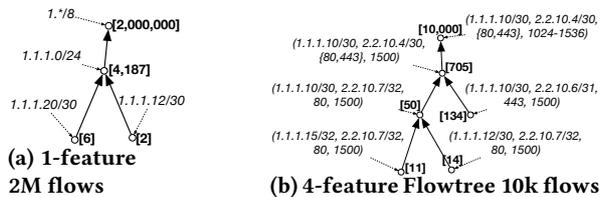


Figure 2: Examples of Flowtrees.

summarize related packets over time at a specific aggregation level. A flow can have 5-features, i.e., protocol, src/dst IP, src/dst port number. Other flow types are 2-feature flows, i.e., src/dst prefixes. Each flow type has a natural hierarchy using wildcards. For IP addresses and ports, hierarchies are expressed via network prefixes and port ranges, respectively.

Consider an example Flowtree in Fig. 2a. 1.1.1.0/24 is a parent of 1.1.1.20/30 as the latter is included in the former. We can map any trace of packets or flows to a corresponding flow graph by annotating each node with its popularity, i.e., packet count, flow count, and/or byte count. E.g., there are 4,187 packets with source IP in the prefix 1.1.1.0/24, so that the node corresponding to 1.1.1.0/24 has a popularity of 4,187. To save space, inspired by [5], Flowtree keeps the most popular nodes and summarizes the unpopular ones. Unlike hierarchical heavy hitter algorithms [1–3, 5], we offer a self-adjusting data structure and do not require prior memory allocation for different feature hierarchies. Moreover, we only keep complementary popularities in the nodes, namely the difference between a node’s popularity and the popularity of its popular children; the definition is recursive as popular nodes are taken to be nodes with high complementary popularity. Hence, in Fig. 2a, when considering nodes of complementary popularity above five to be popular, the complementary popularity of 1.1.1.0/24’s node equals $4187 - 6 = 4181$.

While updating the statistics of a flow in Flowtree, if the corresponding node exists, we simply increment the contribution of the node. If it does not exist, we find the longest matching parent for the node in the tree. The reason is that we optimize for constructing Flowtrees, as the flow arrival rate is expected to be way higher than the query rate. Therefore, we compute the statistics node but do not aggregate the statistics for nodes further in the tree. This leads to an amortized constant update time. Queries can still be answered in time proportional to the of tree nodes.

Flowtree Operators: Flowtree supports multiple operators, namely, query, merge, and diff. The simplest query is asking for the popularity of a flow. If the corresponding node is in the Flowtree, we can directly answer the query. If it is not in the subtree but a parent is, we can estimate its popularity by decomposing the query into a set of queries that can be answered by the given hierarchy. We can merge/diff two Flowtrees A and B by adding/subtracting the nodes of A

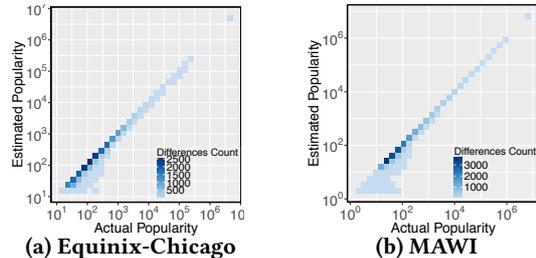


Figure 3: Accuracy of Flowtree (4-features, 40K nodes).

to/from B. This means that the update will only be done for the complementary popularities.

Evaluation of Accuracy: We evaluated the accuracy of Flowtree on different packet captures (6M packets each). Fig. 3 shows a two-dimensional histogram of the estimated vs. real popularities for flows in Flowtree. Each cell indicates how many flows have a specific combination of estimated and real popularities and the darker that cell, the higher the number of flows. More than 57% of entries are on the diagonal and off-diagonal entries are still very close to the diagonal and significantly decrease in number as the popularity rises. All flows which account for more than 1% of the packets are present in the tree. Hence, we not only capture high-popularity flows, but also capture medium/low-popularity flows with acceptable accuracy.

3 SUMMARY AND FUTURE WORK

In this paper, we introduce a novel data structure, called Flowtree, which can efficiently summarize flow captures and enables on-the-fly queries. In future, we plan to use Flowtree as a building block for a scalable trace management system which can be deployed network-wide. We envision that each router exports its data to a close-by Flowtree daemon using APIs such as NetFlow to continuously construct summaries of the active flows using Flowtree, see Fig. 1. This system extends Flowtree by adding two features, namely time and monitor location. Thus, it again enables drill down and quick exploration but also alarming when there are significant differences. We are about to deploy our prototype as test installation at a major IXP and a large ISP.

ACKNOWLEDGMENTS

Support for this work was provided by the European Research Council (ERC) grant ResolutioNet (ERC-StG-679158), by Leibniz Prize project funds of DFG - German Research Foundation: Gottfried Wilhelm Leibniz-Preis 2011 (FKZ FE 570/4-1), and by the German Federal Ministry of Education and Research (BMBF) under grant BBDC: Berlin Big Data Center (01IS14013A).

REFERENCES

- [1] R. B. Basat, G. Einziger, R. Friedman, M. C. Luizelli, and E. Waisbard. 2017. Constant time updates in hierarchical heavy hitters. In *SIGCOMM*.
- [2] G. Cormode, F. Korn, S. Muthukrishnan, and D. Srivastava. 2003. Finding Hierarchical Heavy Hitters in Data Streams. In *VLDB*.
- [3] G. Cormode, F. Korn, S. Muthukrishnan, and D. Srivastava. 2004. Diamond in the Rough: Finding Hierarchical Heavy Hitters in Multi-Dimensional Data. In *ACM SIGMOD*.
- [4] C. Cranor, T. Johnson, O. Spataschek, and V. Shkapenyuk. 2003. Gigascope: A Stream Database for Network Applications. In *ACM SIGMOD*.
- [5] N. Kammenhuber, D. M. Garching, and L. Kencl. 2005. Efficient Statistics Gathering from Tree-Search Methods in Packet Processing Systems. In *IEEE ICC*.