Enabling Fast Hierarchical Heavy Hitter Detection using Programmable Data Planes

Diana Andreea Popescu
University of Cambridge
diana.popescu@cl.cam.ac.uk

Gianni Antichi
University of Cambridge
gianni.antichi@cl.cam.ac.uk

Andrew W. Moore
University of Cambridge
andrew.moore@cl.cam.ac.uk

ABSTRACT

Measuring and monitoring network traffic is a fundamental aspect in network management. This poster is a first step towards an SDN solution using an event triggered approach to support advanced monitoring dataplane capabilities. Leveraging P4 programmability, we built a solution to inform a remote controller about the detected hierarchical heavy hitters, thus minimizing control plane overheads.

CCS Concepts

•**Networks** → **Programmable networks; Network management; Network monitoring;**

Keywords

Hierarchical Heavy Hitters, SDN, P4

1. INTRODUCTION

Network monitoring is the fine art of providing the necessary information for network management. Despite the importance to detect network trends to quickly locate potential issues is well recognized, dataplane observability has not considerably improved over the time. In the past, it relied on packet sampling techniques to lower overheads and data collection bandwidth, thus impacting estimation accuracy. OF does not currently improve the dataplane visibility: the main mechanism exposes the per-port and per-flow counters available in the switches. An application running on top of the controller can periodically poll each counter using the standard OF APIs, and then perform any software-based algorithm to get insights into the network behavior. However, this approach limits significantly the original flexibility intended by SDN. While increasing the gap between two consecutive counters requests reduces the controller capabilities to react in a timely fashion to network events, continuously requesting counters from switches leads to non-scalable solutions by overloading the control plane. For this reason, lately a number of proposal rely on P4 programmability to extend dataplane functionality for more advanced monitoring applications. Recent solutions focused on enhancing network visibility by exporting counters at fixed periods. FlowRadar [4] exports the dataplane flow counters kept for all flows at short time scales, e.g., 10ms. HashPipe [6] determines the heavy hitters in the dataplane, exporting the flow counters every 20 seconds, corresponding to a table flush period, while UnivMon [5] relies on sketches computed in the data plane which are exported to the control plane. Although these architectures prove to be flexible and accurate, the collector, i.e., the network controller, receives at a fixed time interval the generated counters/sketches from the data plane, and estimates the various application-level metrics of interest. However, such an architecture might suffer from the same problems as the "legacy" OF protocol: the ability to detect a network event and react accordingly depends on the controller's capabilities to collect statistics at short time ranges [2]. Our approach takes a different direction. We recognize the need of monitoring the traffic from multiple sources to have a global view of the network and we acknowledge that reducing controller overheads is crucial to avoiding (potential) scalability problems. To this end, in this poster we present a first step towards smarter dataplanes. By leveraging dataplane programmability, we seek to transform the switch from a passive monitoring infrastructure to an "active" one. We envision enhancing the switch functionality with precise monitoring tasks, where the controller gets insights into the network only when a specific event occurs, without facing controller scalability problems. As a first example, in this poster we show how we can leverage P4 programmability to detect Hierarchical Heavy Hitters into the dataplane and inform the control plane only when the detection is completed.

SOSR '17, April 3–4, 2017, Santa Clara, CA, USA

 ISBN 978-1-4503-4947-5/17/04…$15.00

DOI: http://dx.doi.org/10.1145/3050220.3060606

2. SOLUTION

We propose a solution to identify the heavy hitters in a network in a hierarchical manner, reducing the controller overheads, by leveraging P4 to move the monitoring logic in the dataplane. An algorithm for hierarchical heavy hitter detection for OF-enabled switches was presented in [3].

To detect the heavy hitters, the controller inserts rules in the TCAM to match source IP prefixes, starting with the rules that match the 0* and the 1* prefixes. In each time interval, the controller polls the switches for the counts of the rules that have already been inserted, and determines whether the threshold for heavy hitters has been surpassed. If this happens for a specific prefix, the controller will report that IP prefix as a heavy hitter for that time interval, and it will insert additional rules in the corresponding switch to monitor the child prefixes of that IP prefix. In the case that the count for a prefix is smaller that the threshold, the count is added to the parent prefix, and the rule corresponding to that IP prefix will not be used in the next time interval. This algorithm relies on the controller to process the counters, determine the hierarchical heavy hitters for that interval, and decide which prefixes should be monitored in the future. However, using P4, the controller could poll periodically the switch to obtain the current sizes for the monitored IP prefixes, in order to determine the hierarchical heavy hitters for that interval, but it does not need to indicate to the switch which IP prefixes should be monitored during each time interval, leaving this operation to the dataplane. Going further, we eliminate the periodic polling done by the controller by having the dataplane generate a packet to signal the detection of a hierarchical heavy hitter.Thus, when a packet arrives, we need to find the longest IP prefix that matches the source IP address. Following the idea from [7], we use a hash table for each prefix length, thus for IPv4 addresses, we need 32 hash tables. The key to index the hash table is the source IP prefix. When a packet arrives, we check each table, starting with the table whose prefix length is the longest (32 bits, the full IP address), to see if there is a non-zero entry in the hash table. If the sum of the hash table entry and the packet size is larger than a predefined threshold size for hierarchical heavy hitters, then we inform the controller about the detected hierarchical heavy hitter. Additionally, we form the child IP prefix by extending the IP prefix with one additional bit and we store the packet size in the corresponding hash table. If the sum is not larger than the threshold, we store the new sum in the hash table corresponding to the original IP prefix.

P4 Prototype. We built a prototype of our algorithm in P4 version 1.1. Although P4 offers the `lpm` option for longest-prefix matching, we found that we are not be able to construct the child IP prefix from the matched IP prefix, in the case that the IP prefix's count indicates a hierarchical heavy hitter. Thus, our implementation uses a match+action table for each hash table and additional match+action tables for the other operations. A match+action table specifies a single action that applies to every packet. Each table uses a register array for the hash table that holds the IP prefixes counters. The action of each match+action table looks up the packet's source IP prefix in its corresponding hash table. The key used for hashing is the source IP prefix, which is obtained from the source IP address of the packet using a mask to keep only the relevant bits. The control flow specifies a packet's flow through the pipeline of match+action tables. We use 8 such lookup tables, each corresponding to an IP prefix length of 32, 28, 24, 20, 16, 12, 8 and 4 bits, respectively. We used IP prefixes multiples of 4 due to the fact that the number of stages in the pipeline is small [1]. If the location in the hash table is empty, the packet moves to the next table in the pipeline, which corresponds to a smaller IP prefix length. If the location is not empty, the current counter associated with that prefix is read into metadata using another match+action table, and the remaining lookup tables are not applied. Further, we check whether the sum between the current packet's size and the counter read is larger than a predefined heavy hitter threshold size. If it is, then we have found a hierarchical heavy hitter, and we thus need to alert the controller about it, by generating a packet digest (`generate_digest` action in P4) which contains the current IP prefix of the hierarchical heavy hitter, and also to insert the current packet's size in the appropriate hash table, using the child IP prefix, represented by the current IP prefix extended with 4 additional bits of the source IP address. Thus the packet will move through match+action tables whose purpose is to construct the IP prefix to be used for hashing into the hash table corresponding to the child IP prefix. There are 8 update tables, corresponding to IP prefixes of length 32, 28, 24, 20, 16, 12, 8 and 4 bits, respectively. Depending on the value of the child IP prefix, the corresponding table is applied, and the remaining update tables will not be applied. The last match+action table in our pipeline either writes the new size for current IP prefix in the hash table, or, in the case that a hierarchical heavy hitter was detected, just the packet size in the hash table corresponding to the child IP prefix.

In future work we will extend our prototype with other network monitoring applications, and we will evaluate it and compare its performance and accuracy to other existing solutions.

Acknowledgements. This work is supported by the EU FP7 ITN METRICS (grant no. 607728), the EU Horizon 2020 SSICLOPS (grant no. 644866) and EU Horizon 2020 ENDEAVOUR (grant no. 644960) projects.

3. REFERENCES

[1] Bosshart et al. Forwarding Metamorphosis: Fast Programmable Match-action Processing in Hardware for SDN. In *SIGCOMM*. ACM, 2013.

[2] Curtis et al. DevoFlow: Scaling Flow Management for High-performance Networks. In *SIGCOMM*. ACM, 2011.

[3] Jose et al. Online Measurement of Large Traffic Aggregates on Commodity Switches. In *Hot-ICE*. USENIX, 2011.

[4] Li et al. FlowRadar: A Better NetFlow for Data Centers. In *NSDI*. USENIX, 2016.

[5] Liu et al. One Sketch to Rule Them All: Rethinking Network Flow Monitoring with UnivMon. In *SIGCOMM*. ACM, 2016.

[6] Sivaraman et al. Heavy-Hitter Detection Entirely in the Data Plane. In *SOSR*. ACM, 2017.

[7] Waldvogel et al. Scalable High Speed IP Routing Lookups. In *SIGCOMM*. ACM, 1997.