Sampling and Large Flow Detection in SDN

Yehuda Afek*
afek@cs.tau.ac.il

Anat Bremler-Barr†
bremler@idc.ac.il

Shir Landau Feibish*
shirl11@post.tau.ac.il

Liron Schiff*
schiffli@post.tau.ac.il

*Blavatnik School of Computer Science
Tel-Aviv University
Tel-Aviv, Israel

†Computer Science Department
Interdisciplinary Center
Herzliya, Israel

CCS Concepts

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

Keywords

Network monitoring; Software Defined Networks; Heavy Hitters

1. INTRODUCTION

We present techniques for traffic sampling and large flows detection in SDN with Openflow. In many cases, in order to efficiently compute high speed traffic statistics, sampling is needed. While SDN switches are very efficient and considerably simpler to manage than existing routers and switches, they don't offer direct means for sampling and detection of large flows. Both of these capabilities are important for various basic network applications. For example, traffic monitoring is such an application, which is a key ability in providing QoS, capacity planning and efficient traffic engineering. Additional applications which make use of sampling and large flow detection are applications that depend on network visibility, such as security (DDoS and others), anomaly detection, DPI and billing.

Traffic visibility, and specifically measurements and monitoring in IP networks has become a very difficult task due to the overwhelming amounts of traffic and flows. One of the earliest network measurements tools was Cisco Netflow [1], which allowed IP flow level measurement. Netflow provided a variety of monitoring capabilities yet suffered from high processing and collection overheads, which were partially decreased using sampling in the variant *Sampled* Netflow, yet this variant provided reduced accuracy caused by the straightforward use of sampling. In [2] Estan and Varghese significantly improve the accuracy of the sampling process by introducing the *Sample and Hold* algorithm which provides better accuracy while reducing the processing and collection overhead.

In order to increase the availability of monitoring, and following the SDN trend, we explore ways to implement this with the widespread OpenFlow standard for SDN switches. OpenFlow switches provide counters of the number of bytes and packets per flow entry, yet traffic measurement remains a difficult task in SDN for two reasons. The first is the hardware (usually Ternary Content Addressable Memories (TCAMs)) constraints which limit the number of flows which the switch can maintain and follow. The second is the limited number of updates which the switch can process per second, which can therefore limit the amount of updates that the controller will make to the flow table. The algorithms provided herein overcome these limitations by providing efficient building blocks for sampling and large flow detection which can be used by various monitoring applications.

1.1 Our Contribution

First, we present various OpenFlow based methods to sample packets that traverse an SDN switch. These methods are immune to various cyber attacks and are based on Open-Flow 1.3 capabilities. Second, we make use of the sampling mechanisms for the development of an efficient method to detect large flows. The techniques presented are both flow-table size and switch-controller communication efficient.

2. TRAFFIC SAMPLING

2.1 Packet Sampling

In *Packet Sampling* we select each packet in a stream of packets traversing the switch with probability p, $0 \leq p \leq 1$, and send them to a *receiver* that can be the controller or some middlebox (monitoring box). We present three basic approaches for packet sampling, each using different OpenFlow 1.3 optional features which are supported by existing software and hardware switches.

2.2 Pseudo Byte Sampling

In *Pseudo Byte Sampling* we select each byte in a stream of packets traversing the switch with probability p, $0 \leq p \leq 1$, and send the packet containing the sampled byte to a receiver. We present optimized techniques for pseudo-byte sampling, in which we assume that packet size is accessible and can be matched in the OpenFlow pipeline, using the Experimenter extension or any other way. It is expected that with upcoming new OpenFlow and alike SDN archi-

This research was supported by European Research Council (ERC) Starting Grant no. 259085, and the Neptune Consortium, administered by the Office of the Chief Scientist of the Israeli ministry of Industry, Trade, and Labor, and the Ministry of Science and Technology, Israel.

SIGCOMM '15 August 17-21, 2015, London, United Kingdom

ACM ISBN 978-1-4503-3542-3/15/08.

DOI: http://dx.doi.org/10.1145/2785956.2790009

tectures, such as [3] more header fields could be natively matched by the flow tables.

2.3 Evaluation

We evaluate the performance of our sampling schemes by considering the resulting sampled flow size distributions compared to real flow size distribution. Figure 1 shows the three packet sampling methods achieve similar distributions, and closely approximate the real (exact) distribution.

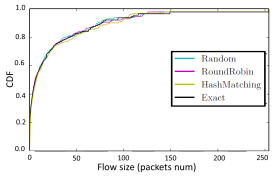


Figure 1: Flow size CDF under three sampling schemes and the exact (not sampled) traffic CDF.

3. HEAVY FLOWS DETECTION

A Heavy flow in a stream of packets S, is a flow which takes up more than T percent of the traffic (i.e., packets) in the stream. Fundamental counter based algorithms for finding Heavy Hitters (or flows) such as that of Metwally et. al. [4], cannot be directly implemented in the SDN framework since in the worst case they would require rule changes for every packet that traverses the switch. A different approach is therefore needed.

First we consider a naive solution which we name Sample&HH, that samples packets in the switch and then sends all sampled packets to the controller. The controller computes the heavy flows using a heavy hitters algorithm. However, as can be seen in Figure 3a (and other works [2]), relying solely on the samples is not accurate enough. Next we consider a solution based on the *Sample&Hold* paradigm of [2] which was devised for identifying elephant flows in traffic of classic IP networks. Sample&Hold achieves very accurate results by using sampling together with accurate in-band counters for sampled flows, yet the high amount of counters and the rate of installing them make Sample&Hold incompatible with SDN switch architecture. Therefore we only consider it as a reference point to evaluate our algorithm.

To deal with the problems of the above solutions, we present our Sample&Pick algorithm. Sample&Pick uses sampling to identify flows that are suspicious of being heavy. For these *suspicious* flows a special rule is placed in the switch flow table providing exact counters for the suspicious flows. The Sample&Pick algorithm considers both the bounded rule space in the switch as well as the time it takes for the controller to install a rule in the switch. Therefore we use two separate thresholds, one for determining which flows are heavy and a second lower threshold for detecting potentially large flows. This lower threshold allows us to install a rule in the switch early enough to get an accurate count of the large flows, yet we do not install rules for too many flows that will remain small.

Our algorithm operates as follows: in the first step we sample the flows going through the switch using one of the sampling techniques mentioned. As can be seen in Fig. 2, these samples are sent to the controller, that feeds them as input to a heavy hitters computation module in order to identify the suspicious heavy flows (steps 2 and 3). Once a flow's counter in the heavy hitters module has passed some predefined threshold t, a rule is inserted in the switch to maintain an exact packet counter for that flow (steps 4 and 5). This counter is polled by the controller at fixed intervals and stored in the controller (steps 6 and 7). Finally the last step increments the counters that are processed by the Heavy Hitters module to maintain correct counters of non-sampled flows.

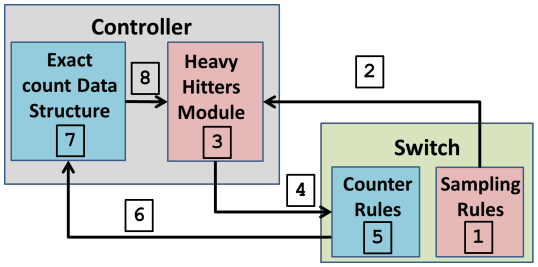
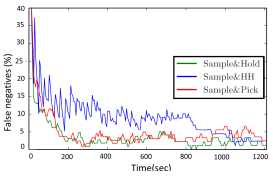
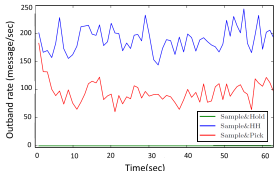


Figure 2: Sample&Pick overview



(a) False negative errors, shown by the ratio between the Heavy Hitter (HH) flows missed to the total number of HH flows.



(b) Rate of PacketIn messages (samples) from switch to controller. In Sample&Hold, sampling is switch-contained.

Figure 3: Heavy Flow Detection test results

3.1 Evaluation

We compare our Sample&Pick algorithm to the two additional solutions described above Sample&Hold and Sample&HH.

Testing shows: Considering a line rate of $2 \cdot 10^4$ pps, it takes the scheme about 10^7 packets to stabilize (Figure 3a). Furthermore, the controller in Sample&Pick processes approximately half the samples than in Sample&HH (Figure 3b), due to the fact that Sample&HH has no counters in the switch so all traffic is sampled, whereas Sample&Pick uses switch counters for heavy flows; Parameters used: $T = 10^{-3}$, $p = 0.5 \cdot 10^{-2}$, $t = T/2 = 10^{-3}/2$.

4. CONCLUSIONS AND FUTURE WORK

We have presented techniques for sampling and large flow detection in SDN. Our sampling techniques are unique in that they are simple and remain within the confinements of the OpenFlow standard. Our algorithmdetects large flows with a relatively small error rate while minimizing the computation overhead in the switch and requiring little controller-switch communication.

5. REFERENCES

[1] [Online]. Available: http://www.cisco.com/c/en/us/tech/quality-of-service-qos/netflow/index.html

[2] C. Estan and G. Varghese, "New directions in traffic measurement and accounting: Focusing on the elephants, ignoring the mice," *ACM Trans. Comput. Syst.*, vol. 21, no. 3, pp. 270–313, 2003.

[3] P. Bosshart, D. Daly, G. Gibb, M. Izzard, N. McKeown, J. Rexford, C. Schlesinger, D. Talayco, A. Vahdat, G. Varghese, and D. Walker, "P4: programming protocol-independent packet processors," *Computer Communication Review*, vol. 44, no. 3, pp. 87–95, 2014.

[4] A. Metwally, D. Agrawal, and A. El Abbadi, "Efficient computation of frequent and top-k elements in data streams," in *ICDT*, ser. Lecture Notes in Computer Science, T. Eiter and L. Libkin, Eds., vol. 3363. Springer, 2005, pp. 398–412.