# Detection of Bidimensional Hierarchical Heavy Hitters in OpenFlow Networks

**Mário Augusto da Cruz**[1], **Sand Luz Corrêa**[1] **Kleber Vieira Cardoso**[1]

[1]Instituto de Informática (INF) – Universidade Federal de Goiás (UFG)
Caixa Postal 131 – 74.001-970 – Goiânia – GO – Brasil

{marioaugusto, sand, kleber}@inf.ufg.br

***Abstract.*** *Traditional approaches for detecting Hierarchical Heavy Hitters (HHHs) can be also employed in OpenFlow Networks. However, OpenFlow devices offer the opportunity to develop new solutions. In this paper, we discuss how online pattern matching can be different in OpenFlow networks, mainly when transporting large traffic aggregates. In addition, we present a new mechanism for online detection of Hierarchical Heavy Hitters in networks based on OpenFlow switches. Our proposal is able to detect bidimensional HHHs based on source and destination IP addresses.*

## 1. Introduction

One of the key challenges in traffic monitoring is to detect high-volume traffic aggregates in real-time. Such aggregates often contain different ranges of IP addresses, i.e. they need to be treated in a hierarchical manner. Besides, these aggregates may involve the combination of different IP header fields, i.e. they may be multidimensional. In the literature, this problem is referred to as Hierarchical Heavy Hitters (HHHs) detection.

In the last decade, many solutions have been developed for the detection of HHHs under the constraints of the traditional IP networks [Cormode et al. 2003, Zhang et al. 2004, Hershberger et al. 2005]. In most of these solutions, the detection of HHHs is typically implemented in a network appliance which is able to process packet headers at least until the network layer and, thus, packet headers are assumed to be available to be checked when necessary. In general, some sampling technique is also employed in order to avoid computation overhead and forwarding delay, specially in high-speed links.

Applying traditional HHH detection solutions to Software-Defined Networks, such as OpenFlow (OF) networks, is not straightforward. In OF networks, basic header processing and forwarding are implemented on data plane of OF switches, and complex computations, such as HHH detection, are implemented externally in a controller. As a result, in OF switches, the sampling techniques cannot be applied in the conventional way, since most of the packets will not be available to the main processing unit, i.e. the controller. Theoretically, all packets can be sent to the controller, however this is far from realistic due to performance issues, mainly in high-speed networks. Besides, OF devices offer the opportunity to count rule matches in a per packet basis, which it is not available in traditional networks. Therefore, the HHH detection problem needs to be revisited taking in consideration the features of OpenFlow networks.

In [Jose et al. 2011], the authors have proposed an algorithm for unidimensional HHH identification as part of a measurement framework focused on OF switches. Basically, the proposed mechanism reads packet counters provided by switches. This information is used to find out the HHHs of the interval. The algorithm presents a fair tradeoff between measurement accuracy and switch overhead. However, the time for effectively identify HHH in large traffic aggregates is far from the desirable, as we show in our evaluation section. Recently, the authors of [Mitzenmacher et al. 2012] have proposed an algorithm with worst-case time and space bounds better than the previous algorithms in the literature. However, this algorithm needs to update rules on a per-packet basis, which is not viable in OF switches.

In this paper, we propose a new mechanism for the detection of bidimensional HHH that combines the packet counters available in OF switches with an on demand packet sampling. The packet counters are employed to build a trie with the IP prefixes that are identified as HHHs. Periodically, the packet counters are read from the switch in order to update the HHH information. In order to quickly react to changes in network flows, we mirror the traffic aggregate for a short period of time and send it to a collector device that employs packet sampling. The information returned by the collector is merged with the HHHs detected by the packet counters. This mechanism keeps the detection time low even when the set of HHHs changes frequently. In addition, our proposal achieves high accuracy and demands for low overhead.

This paper is organized as follows. In Section 2, the concept of Hierarchical Heavy Hitter is reviewed and the context of OF network is commented. Section 3 presents our proposal in detail. Section 4 describes the performance evaluation of our mechanism. Section 5 presents the conclusions and discusses future work.

## 2. Background

Current Internet Service Providers (ISPs) networks have high-speed links with several gigabits per second, which potentially means thousands of packets per second. In this context, it is critical to have automated mechanisms to detect suspicious behaviors. A typical suspicious behavior is high usage of network elements, thus it is important to track the flows which are the largest network consumers. In the literature, these flows are usually known as frequent items or Heavy Hitters (HHs) [Cormode et al. 2003]. Network flows are an example of stream data that can be summarized in a hierarchical structure due to their aggregatable format. Flows that consume resources above a certain threshold are called Hierarchical Heavy Hitters (HHHs) [Cormode et al. 2003]. In the following, we present a more formal description for the concepts of HH and HHH.

Heavy Hitter (HH) is a flow or a traffic aggregate that consumes a significant portion of the total network resources. Assuming the total amount of traffic, measured at regular time intervals, is $C$ and given a specific threshold $\phi$, a Heavy Hitter is a flow or a flow aggregation whose volume is at least $\phi C$ and the maximum number of HHs is $1/\phi$. As an example, if $\phi = 5\%$ and the total amount of traffic is $C = 1$ Gbps, all flows with rate of 50 Mbps or above are considered HHs and there are at most 20 HHs.

Hierarchical Heavy Hitters (HHHs) are the Heavy Hitters that are disposed into a hierarchy and obey the following rules. If an HH is on the bottom level of the hierarchy it is also an HHH, but its traffic volume must be discounted as we go up in the hierarchy. At any level above the bottom, an HH is an HHH if its traffic volume hits the threshold $\phi$ after discounting the sum of all HHHs descendants down in the hierarchy. For example, lets suppose we build a hierarchy based on the IP addressing and the following parameterization is employed: $\phi = 5\%$ and $C = 100,000$ pps. As part of the example, lets assume the following network usage hierarchy: 10.0.0.0/32 = 5,010 pps; 10.0.0.0/30 = 9,081 pps; 10.0.0.0/28 = 13,317 pps. In this example, 10.0.0.0/32 and 10.0.0.0/28 are HHHs, but 10.0.0.0/30 is not an HHH. Indeed, the HHH concept provides a more compact and precise information, since it removes eventual redundant information about HHs that achieved the threshold only due to theirs successors in the hierarchy.

Some works on HHH detection are limited to only one dimension [Truong and Guillemin 2007], including the paper [Jose et al. 2011] that is designed to operate in OF networks such as our proposal. However, our approach operates in two dimensions, the source and destination IP addresses, similar to the works [Zhang et al. 2004, Mitzenmacher et al. 2012]. According to [Zhang et al. 2004], the 1-dimensional and 2-dimensional HHH detection are arguably the two most important scenarios for traffic analysis.

## 3. Proposal

OF switches offer the opportunity to forward high-volume traffic quickly, but with minimum traffic monitoring. While a secondary device can inspect every packet, this would imply on performance penalties. The issues related to these extreme approaches suggest that a hybrid solution should be the best choice. We have built our solution based on this principle and we named it as *Online Detection of Hierarchical heavy hitters using In-depth inspection* or simply ODHIn. Our mechanism is composed of two parts: a controller that handles counters from OF switch tables and computes HHHs; and a collector which uses sampling to reduce the amount of packets and computes the HHHs over these samples.

In order to track the hierarchy established over the IP addresses, we employed a prefix tree or trie in the controller. The trie structure supports insertions of pairs of IP prefixes (source and destination) and also provides fast prefix searching. In this trie, each node keeps a pair of prefixes. The controller creates an initial hierarchy to accelerate the startup of the mechanism. This initial hierarchy is generic and is formed by simple expansion of prefix in a binary format, as partially illustrated by Fig. 1. The prefix in each trie node becomes a rule in the OF switch and, thus, it has an associated counter which is updated in a per packet basis. At each measurement interval $M$, the controller reads the counters in order to update the trie nodes. Based on this information, the controller computes the HHHs and evaluates if it is necessary to change the trie and to call the collector.

Due to memory and processing constraints, it is important to limit the number of wildcard rules in OF switches. To address this problem, we employed an approach similar to the one proposed in [Jose et al. 2011], with two modifications. First, our trie is not binary but quaternary. Second, we start our mechanism with a pre-built trie and, thus, we apply the defined bounds since the startup of the mechanism. The bounds are defined as a function of the maximum number of HHs, as described in the following.

The maximum number of HHs is $1/\phi$. In the worst case, when the number of predecessors is the maximum, every HH has a different predecessor in the trie, and we need additional $1/\phi$ wildcard rules to monitor the predecessors. In addition, every time a

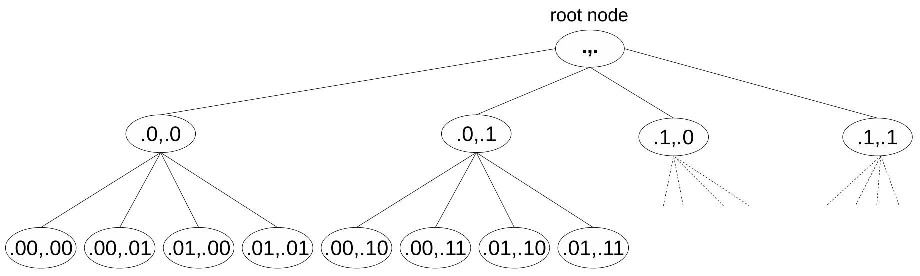

**Figura 1. Trie partially created – each node contains a tuple with source and destination prefix.**

leaf node becomes an HH, we need to install rules for monitoring its successors because they can become a new HH. As a consequence, we need additional $4/\phi$ wildcard rules to monitor the successors. Thus, the number of wildcard rules is given by:

$$wildcard\_rules = \frac{1}{\phi} + \frac{1}{\phi} + \frac{4}{\phi}. \quad (1)$$

As the number of nodes must be a multiple of 4, due to the trie structure, the effective number of rules must be constrained to $4^{\lceil log_4(wildcard\_rules) \rceil}$. Initially, the trie structure is perfectly balanced and has height equal to $\lceil log_4(wildcard\_rules) \rceil$. For example, if $\phi = 5\%$ and $C = 1\ Gbps$, we would need 120 wildcard rules and an initial trie with height 4.

At each measurement interval $M$, the controller computes the HHHs based on the dynamic total amount of traffic $C$. This approach is different from some traditional solutions that assume $\phi$ as function of a fixed parameter, e.g. the link capacity. Our approach is best fitted to traffic conditions, which varies quickly over time.

The collector is called every time a prefix achieves the threshold $\phi$ and if there is no pending call to the collector. In order to send packets to the collector, the controller sets a temporary rule in the switch to copy or mirror the traffic to the device running the collector. Since our mechanism is designed to deal with large traffic aggregates, the collector employs a sample technique to keep the response time small and to avoid processing and memory overhead. The main purpose of the collector is to accelerate the discovery of HHHs. This is important because the approach employed in the controller presents large delays when there is a large distance between the prefix set in the node trie and the effective prefix of an HHH.

Since the collector makes an in-depth inspection into the traffic it receives, it can be as accurate as an offline algorithm over this limited set of samples. In addition, the collector algorithm creates a list of HHHs based on the longest prefix match approach. As a consequence, the collector provides the most specific prefixes that are identified as HHHs. The HHHs detected by the collector are returned to the controller that merges the information with the HHHs computed previously. The controller is an OF application that runs over a true OF controller, e.g. NOX [Gude et al. 2008].

## 4. Performance Evaluation

The authors of [Jose et al. 2011] kindly have made available the source code of theirs simulator. Based on this code, we developed a new simulator that implements the ODHIn algorithm, i.e, which is able to deal with bidimensional HHHs and to mirror traffic to a collector device. The simulator operates over real packet traces and each packet is taken as a discrete event.

In the following, we present results of our simulator running over a CAIDA trace. This trace is an updated version of the one employed in [Jose et al. 2011]. It was collected in a OC192 link with packet rate of 400 Kpps or 5 Gbps. Similar to [Jose et al. 2011], we take 1 minute from the traffic trace.

We evaluated our mechanism under two traditional metrics of accuracy: recall and precision [Cormode and Hadjieleftheriou 2010, Jose et al. 2011]. Recall is the total number of true HHHs reported by a non-exact algorithm over the number of true HHHs given by an exact (offline) algorithm. Thus, recall represents the misdetected HHHs and the higher the recall, the lower the false negatives. In order to compute the recall rate, we developed an exact algorithm that processes the CAIDA trace and provides all prefixes that are true bidimensional HHHs.

The second metric, precision, is the number of true HHHs reported by our algorithm over the total number of answers reported. Thus, precision represents the amount of false positives reported. Figure 2 shows the recall and precision values obtained by ODHIn over different sampling intervals (from 10 $\mu$s to 1000 $\mu$s). Similar to [Jose et al. 2011], we have employed a measurement interval ($M$) of 5 s and the threshold $\phi$ is configured to 5%. Each result is the average of the 12 samples collected using the 1 minute-trace. Mean values are presented with confidence level of 95%.

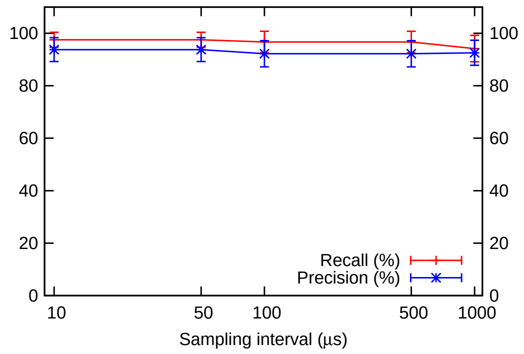

**Figura 2. Recall and precision as function of the sampling interval.**

| *algorithm* | *precision* | *# intervals* | *# HHH* |
|---|---|---|---|
| [Jose et al. 2011] | 75-88% | 5 | 8 |
| ODHIn | 86-95% | 1 | 10 |

**Tabela 1. Comparison of both algorithms.**

Figure 2 shows that ODHIn achieves good accuracy for both metrics. However, the accuracy decreases for sampling intervals of 500 $\mu$s or above. This result is expected since less frequent sampling implies in leaving more traffic without analysis.

Table 1 compares the results achieved by ODHIn and the algorithm proposed in [Jose et al. 2011]. The comparison is made taking into account the precision rate, the number of HHHs detected and the number of measurement intervals taken by the algorithm to provide the answer. In this evaluation, ODHIn uses a sampling interval of 100 $\mu$s, and the threshold $\phi$ is configured to 5% in both algorithms. From the results, we can observe that ODHIn achieves higher precision and takes less measurement intervals to detect the HHHs. The reason is that the algorithm in [Jose et al. 2011] builds a tree from scratch and, at each interval, the tree grows until it reaches a stable state. Unlike, we have designed our solution with a pre-built tree, which accelerates the process of detection.

## 5. Conclusion and Future Work

In this paper, we described an algorithm for efficient detection of bidimensional Hierarchical Heavy Hitters, designed to work in OpenFlow networks. Using counters from flow tables of OF switches and packet sampling, we proposed an algorithm that quickly detects large aggregates. We presented an evaluation of the algorithm and we shown that the proposed mechanism has good accuracy and can detect HHHs in few measurement intervals. We also compared the performance of this algorithm with other algorithm of the literature, also designed to work with OpenFlow. Our algorithm overcomes the previous one in terms of accuracy and detection time. As future work, we plan to do tests on commodity switches with real network streams, and to extend the algorithm to work with more than 2 dimensions.

## Referências

Cormode, G. and Hadjieleftheriou, M. (2010). Methods for finding frequent items in data streams. *The International Journal on Very Large Data Bases*, 19(1):3–20.

Cormode, G., Korn, F., Muthukrishnan, S., and Srivastava, D. (2003). Finding hierarchical heavy hitters in data streams. In *International conference on Very large data bases*, pages 464–475.

Gude, N., Koponen, T., Pettit, J., Pfaff, B., Casado, M., McKeown, N., and Shenker, S. (2008). NOX: towards an operating system for networks. *SIGCOMM Comput. Commun. Rev.*, 38(3):105–110.

Hershberger, J., Shrivastava, N., Suri, S., and Tóth, C. D. (2005). Space complexity of hierarchical heavy hitters in multi-dimensional data streams. In *ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems*, pages 338–347.

Jose, L., Yu, M., and Rexford, J. (2011). Online measurement of large traffic aggregates on commodity switches. In *USENIX conference on Hot topics in management of internet, cloud, and enterprise networks and services*, pages 13–13.

Mitzenmacher, M., Steinke, T., and Thaler, J. (2012). Hierarchical Heavy Hitters with the Space Saving Algorithm. In *Meeting on Algorithm Engineering and Experiments*, pages 160–174.

Truong, P. and Guillemin, F. (2007). Dynamic binary tree for hierarchical clustering of ip traffic. In *Global Telecommunications Conference, 2007. GLOBECOM '07. IEEE*, pages 6 –10.

Zhang, Y., Singh, S., Sen, S., Duffield, N., and Lund, C. (2004). Online identification of hierarchical heavy hitters: algorithms, evaluation, and applications. In *ACM Internet Measurement Conference (IMC)*, pages 101–114.