IDEAFIX: Identifying Elephant Flows in P4-Based IXP Networks

Marcus Vinicius Brito da Silva, Arthur Selle Jacobs
Ricardo José Pfitscher, Lisandro Zambenedetti Granville
Institute of Informatics – Federal University of Rio Grande do Sul
Av. Bento Gonçalves, 9500 – Porto Alegre, Brazil
Email: {mvbsilva, asjacobs, rjpfitscher, granville}@inf.ufrgs.br

***Abstract*—Internet Exchange Points (IXPs) are high-performance networks that allow multiple autonomous systems to exchange traffic, with benefits ranging from cost reductions to performance improvements. In addition, performance requirements and a number of players involved in such networks bring out several issues to management tasks, such as elephant flows identification. This kind of flows, with high size and substantial duration, can severely impact the performance of smaller flows. In this paper, we present IDEAFIX, a mechanism to identify elephant flows in P4-based IXP networks. Our approach consists in analyzing flows features for each ingress packet immediately in the edge switch. These features are then stored in P4 registers, indexed by hash keys, and compared to predefined thresholds for flow classification. Experimental evaluations show that IDEAFIX is significantly more efficient than the state-of-the-art approaches implemented with sFlow and traditional Software-Defined Networking (SDN) tools (*e.g.*, OpenFlow). While state-of-the-art mechanisms add up to 17MB of monitoring data, our solution causes an overhead of only 25KB. Also, the implemented prototype takes less than 0.40ms to identify elephant flows with a 95% accuracy in scenarios with scarce memory resources.**

***Index Terms*—Software-Defined Networking, Internet Exchange Points, Network Management, Programmable Networks**

I. Introduction

Internet Exchange Points (IXPs) connect autonomous systems (ASes) of the Internet and enable service providers to perform traffic exchange in order to better serve their customers [1]. IXPs are distributed throughout the world and perform an essential role in the Internet ecosystem [2], accounting for at least 20% of all traffic exchanged between ASes [3]. The main benefits experienced by IXP participants are the low cost of deployment and maintenance [4]. IXP operators face several management challenges, such as Address Resolution Protocol (ARP) traffic monitoring, Virtual Private Network (VPN), and flow management [5].

Among the challenges of managing flows that travel in an IXP infrastructure, the identification of elephant flows [6] can promote a better use of the services provided by the network. A flow is considered to be an elephant when it has a traffic size and duration significantly high according to predefined thresholds [6]. These flows represent the major fraction of the traffic volume, although they tend to be a small subset of all flows. As a consequence, elephant flows can significantly impact traffic from smaller flows (those that do not exceed the thresholds) that are sharing the same path [5].

Some proposed solutions, such as SDEFIX [1], DevoFlow [7], and OpenSample [8] present mechanisms for identifying elephant flows in IXP networks using sFlow [9] for traffic monitoring and the OpenFlow protocol [10] for path management. However, these approaches rely on extracting flow samples from the data plane and analyze them in the control plane. This delays the identification process, since monitoring data need to be transmitted to the controller before any processing can be done, and only after the new routing decisions can be sent back to the switches.

The intrinsic delays of SDN switch-controller communication can be mitigated with the advent of programmable data planes, since programmable switches can perform the flow analysis internally. In HashPipe [11] and Hierarchical Heavy-Hitter Detection (HHH) [12], emerging programmable switches [13] are used to analyze flows and identify heavy-hitter directly in the data plane, performing processing on packets beyond traditional forwarding. These approaches compute only the flow size, storing this information in registers indexed by hash keys to identify the heavy-hitters. However, such approaches do not take flow duration into account, which is crucial in mitigating the impacts of elephant flows.

In this paper, we present IDEAFIX, an extension of the mechanism presented in our previous work [14]. The purpose of IDEAFIX is to identify elephant flows in IXP networks with programmable data planes. To do that, the mechanism computes the size and duration of flows and stores them in registers indexed by hash keys. This information is extracted from each packet that enters the network and compared to predefined thresholds for flow classification. When a flow is classified as elephant, a notification is sent to the control plane to report the detected elephant flow. Thus, the network operator can define actions to mitigate the effects of that flow, in accordance with, for example, quality of service policies.

IDEAFIX, whose prototype takes advantage of P4 [13], is significantly more efficient in identifying and reacting to elephant flows than the approach of traditional SDNs, *i.e.*, using OpenFlow, and sFlow. Our results demonstrate that it is possible to identify and react to elephant flows in less than 0.40ms while adding only $\approx$ 25KB of monitoring data to network traffic. In addition, as most solutions do, IDEAFIX allows adjusting the memory space on the switches dedicated to the identification process according to the desired accuracy.

When space to hash mapping increases (*e.g.*, 100% of flow number), the false positives and false negatives decrease (*i.e.*, less than 10% and 1%, respectively) and true positives and true negatives increase, about 95% of the accuracy.

The remainder of this paper is organized as follows. In Section 2, related work is discussed. In Section 3, we describe the mechanism to identify elephant flows. In Section 4, we present the evaluation of our proposal, as well as the achieved results. Finally, in Section 5 we present the main conclusions of the study and future work perspectives.

II. Related Work

In this section, we present work related to our proposal split into two categories. First, we show the use of SDN/OpenFlow in IXP networks for identifying elephant flows. Second, we discuss the proposals analyzing network traffic directly in the data plane using P4.

A. SDN in IXP

The use of SDN in IXPs is proposed in SDX [13] as a solution to the problems in these networks, such as the inherent issues of BGP and wide-area server load balancing [15]. However, that approach does not deal explicitly with elephant flows. In contrast, in DevoFlow [7], the OpenFlow protocol is used to keep track of elephant flows with different monitoring mechanisms, such as packet sampling and port triggering. When a predefined threshold is exceeded, in terms of byte count, the flow is classified as elephant and so it is rerouted to the least congested path between endpoints.

In OpenSample [8], sFlow is used to perform flow sampling. Then, flow rates are calculated by subtracting the TCP sequence numbers from two samples of the same flow and dividing the value by the elapsed time between them. When a flow is classified as elephant, it is redirected by the SDN controller to another path. In SDEFIX [1], an identification module is used to classify flows, analyzing the collected data by sFlow according to predefined rules. When a flow is classified as elephant, it is mitigated according to predefined policies provided by the network operator.

Even though such approaches are successful in identifying and mitigating elephant flows, they require flow samples to be analyzed in the control plane. This process, besides delaying analysis causes an additional volume of monitoring data in the network. In contrast, our approach performs elephant flow identification directly in the data plane.

B. Flows Analysis in Programmable Data Planes

In HashPipe [11], a packet processing algorithm, implemented in P4 switches, is proposed to identify heavy-hitter flows entirely on the data plane. Heavy-hitters refers to t largest flows, according to the number of packets or bytes, of the total number of packages traversing the link in a time window [11], regardless of the flow duration. In HHH [12], heavy-hitter detection is performed through a hierarchical aggregation of IP addresses (*i.e.*, subnet flow), in which flows are mapped by a hash function on flow info (*e.g.*, 5-tuple). Despite being very similar to elephant flows, heavy-hitters may also represent malicious traffic and must consequently be dropped. In contrast, elephant flows consist of legitimate high-volume and long-duration flow that needs to be managed.

III. IDEAFIX

Aiming to provide a way to detect elephant flows in P4-enabled IXP networks, we introduce IDEAFIX. The mechanism takes advantage of P4 features to store information about the size and the duration of network flows. A 5-tuple (*i.e.*, source and destination IP addresses and ports, and transport protocol type) is used to create a hash key about each packet arriving at the switch, which allows us to further analyze network flows.

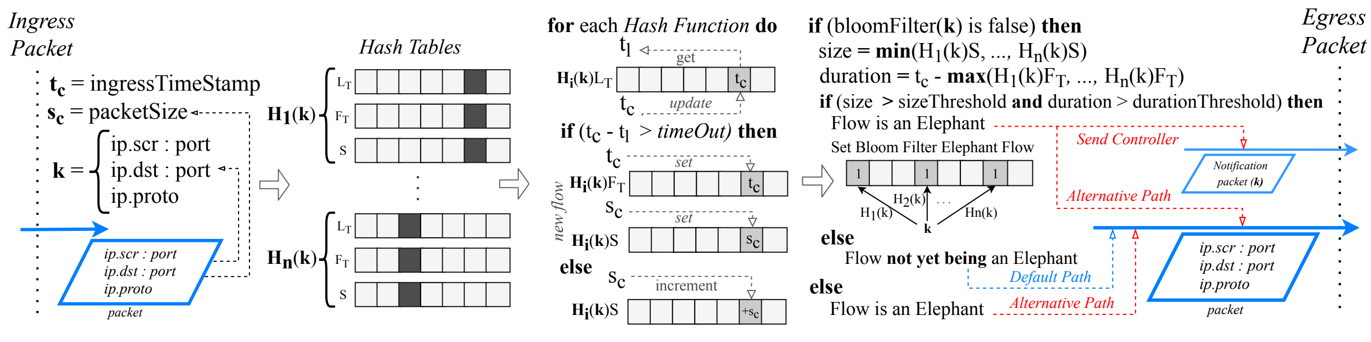
Figure 1 presents the four main steps of IDEAFIX detection process: *extraction*, *consolidation*, *classification*, and *notification*. First, when a packet arrives at the switch, we map the extracted 5-tuple of the packet through a hash function. Next, the consolidation step identifies to which network flow the packet belongs to update the size and duration of that flow. Then, the classification step compares the consolidated information to thresholds to identify whether the flow is an elephant one. Finally, the notification step acts to report the control plane about detected elephant flows, which will act according to the policies defined by network operators.

A. Extraction and consolidation

The extraction step (Figure 1(a)) collects from packets the information required to both identify flows and detect elephant characteristics. When a packet arrives in the switch, its time of ingress is saved to the packet metadata (*ingressTimeStamp*) in local variable t_c, the packet size is stored in local variable s_c, and and the flow identification (5-tuple) is stored in local variable k. As k identifies the flow, by using multiple hash functions, we can map k to the register indexes for storing flow data. A register is a stateful memory available in the P4 language whose values can be read/written by the indexed position. By relying on multiple hash functions instead of a single one, we reduce the possibility of information loss/overlap caused by hash collisions.

In the hashing process, each of the hashing functions maintains three registers, referred to as L_T, F_T, and S. Each slot of L_T (48 bits) stores the ingress time of the *last packet* of a particular flow (assuming no collisions happened). In turn, each slot of F_T (48 bits) stores the ingress time of the *first packet* of a flow. Finally, each slot of S (32 bits) stores the cumulative sum of the *sizes* of every packet of a particular flow (again, assuming no collisions). In case of collision, information could be wrongly overwritten or accumulated. There is no way to guarantee 100% that there will be no hash collisions, but our approach reduces those chances.

To mitigate the impact of hash collisions, we first verified whether the packet belongs to an already existing flow, or if it is a new flow entirely. This is done by calculating the time interval between the last packet and the current packet ($t_c - t_l$) and comparing this difference with a predefined



(a) Extraction and hash mapping (b) Consolidation (c) Classification and notification

Fig. 1. Packet process.

timeout (Figure 1(b)). If the timeout is exceeded, the packet represents a new flow and the register will be restarted for the information of the current packet. Then, F_T and S are updated to the values of the ingress time t_c and size s_c of the current packet, respectively. If the timeout is not exceeded, the packet belongs a current flow, then only its size s_c is incremented in S. Finally, in either case, the L_T register is updated to the ingress time t_c of the current packet.

It is important to point out that this hashing approach may generate false positives, *i.e.*, reporting a non-elephant flow as an elephant. Another method of updating these registers could generate false negatives, *i.e.*, not reporting an elephant flow. However, considering that false negatives can lead to network bottlenecks, where unidentified elephant flows can consume a lot of resources shared with other flows, it is preferable to treat with caution non-elephant flows that have been wrongfully classified (*i.e.*, false positives) than to admit false negatives.

B. Classification and notification

The goal of this step is to characterize a flow as an elephant or not. The first step of the analysis is to verify if the flow referring to the packet has already been previously identified as an elephant. The proposed approach uses a bloom filter [16] to flag which flows were already identified as elephants. Initially, all filter memory slots are set to zero. When a flow is identified as elephant, a fixed number of filter slots pointed by special hash functions are set to one. If the flow has not been identified previously, then the size and duration of the flow will be estimated considering all hash functions. For *size*, in Figure 1(c), a count-min strategy ($min(H_1(k)S, ..., H_n(k)S)$) is enough to determine the size closest to the actual size of the flow, because the minimum value is contained in the slot with the minimum (or even no) collisions. The flow duration is estimated by the difference between the current time and the ingress time of the first packet of that flow, which is stored in F_T registers. In this case, in contrast to the estimation of size, the maximum value recorded across all registers ($max(H_1(k)F_T, ..., H_n(k)F_T)$) is used as an approximation of the ingress time of the first packet.

As an example of the process mentioned above, consider that an F_3 flow, whose first packet arrived at the switch at time t_3 (Figure 2), had collisions on all its hash keys with current flows F_1 and F_2, whose first packet arrived at the switch at time t_1 and t_2, respectively. The strategy to update registers described in the previous subsection predicts that these will be restarted when a hash key collision occurs. Therefore, F_3 will assume the ingress time information already stored in F_1 and F_2. By retrieving the maximum value among the F_T registers, we get the time closest to the flow start time. In this example, that would be the ingress time of the first packet of F_2 (t_2), because it is larger than the ingress time of the first packet of F_1 (t_1) and the closest time to F_3 (t_3).

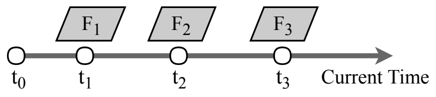


Fig. 2. Flows start time.

Next, flow size and duration are checked against thresholds (*sizeThreshold* and *durationThreshold*) configured by the network operator. If size and duration do not exceed predefined thresholds, the flow is classified as not yet being an elephant one, and the packet is routed to the default path. If the thresholds are exceeded, however, the flow is identified as elephant and a notification is sent to SDN controller for accounting. Also, the flow packets will then be routed through an alternative path. Several elephant flow mitigation policies can be applied; as it is not the focus of this paper to manage elephant flows, we use as an example, alternative path routing.

IV. Evaluation

In this section, we evaluate IDEAFIX performance for elephant flows identification in IXP networks. We first describe the comparison mechanism developed with OpenFlow/sFlow, based on state-of-the-art [1] [7]. Then we present the used experimental scenario and the evaluation methodology. Finally, we present and discuss the main results.

A. Comparison Approach

The state-of-the-art mechanism was developed with the OpenFlow and sFlow protocols, based on the related work [1]

[7]. In this mechanism, a sFlow collector obtains sample flows from the edge switches and estimates flow transmission rates. Sampling rates of 1/128 and 1/1024 were used to analyze the results in this range because the sFlow collector estimates the bandwidth in distinct sampling rates. In addition, an analysis module requests the flow estimates to the sFlow collector at a frequency of 0.01s, 0.1s, and 0.5s. This mechanism is illustrated in Figure 3. When an elephant flow is identified, the controller is notified to insert a rule into the edge switch to mark the packets of the identified elephant flow and route them by an alternative path.

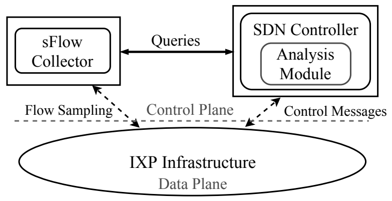


Fig. 3. State-of-the-art mechanism.

B. Scenario

Figure 4 depicts the topology used in the experiments, based on the AMS-IX [17] infrastructure, which has been also used in the related work [1]. It presents an approach where 8 ASes are connected to the IXP network by the edge switches. TCP flows were created between the ASes following a client-server model. The flows were generated considering two factors: bandwidth and duration. The flow bandwidth was established at 10 Mbps, and the flow durations were drawn from a normal distribution with a mean of 8 seconds and a standard deviation of 5 seconds. The thresholds were defined in 10 MB and 10 seconds [1] [5]. For each evaluation round, 2048 flows were generated, of which approximately 12% were elephant flows. Each experiment lasted 10 minutes and was repeated 32 times.

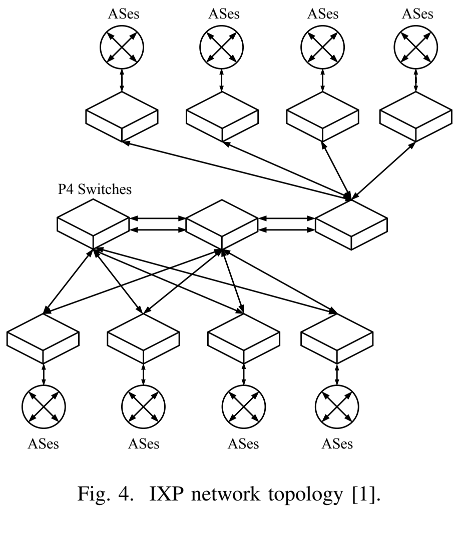


Fig. 4. IXP network topology [1].

In order to prioritize time requirements in the context of an IXP network, in which decisions/actions need to be taken quickly, a proactive approach has been developed in both mechanisms (IDEAFIX and state-of-the-art). That is, when a flow is started, the controller inserts the routing rules into the default path; as also, it inserts the routing rules for an alternate path. When an elephant flow is identified, its packets are marked to indicate the identification (*e.g.*, set an IP header flag) and forwarded by an alternate path. This approach demands more memory resources for the routing tables. However, this allows reacting to an elephant flow more quickly, as will be shown in the following results.

The experiments were performed on a computer with Intel Core i7-4790 processor with 8 cores of 3.6 GHz; 16 GB of RAM; and *Linux Ubuntu* 16.04 LTS. The prototype was implemented in the language $P4_{16}$ and using the software switch BMv2[1]. The state-of-the-art mechanism was developed with the *RYU SDN Framework* version 4.2, the *OpenFlow* protocol version 1.3, *InMon sFlow-RT*[2], and *Open vSwitch* 2.0.2. The infrastructure was emulated using *Mininet* version 2.3, with a bandwidth of 1 Gbps per link and no propagation delay. The workload was generated with *iPerf* version 3.0.11.

C. Metrics

The performance evaluation metrics are: (i) threshold violation/surpassing reaction time, (ii) excess data, (iii) monitoring data, (iv) resources utilization in the state-of-the-art mechanism, and (v) IDEAFIX accuracy. The threshold violation/surpassing reaction time is the interval between the instant of ingress of the packet that exceeds the thresholds and the instant when the first packet is routed through the alternative path. The excess data are the number of bytes that transited in default path since the flow has been identified as an elephant one (until a reaction occurs). The monitoring data is the data added in the network by the identification mechanism. In IDEAFIX, this is the amount (in bytes) of notification messages sent to the controller to report the identified flows. In the state-of-the-art mechanism, these are the data of flow sampling. The resource utilization is the memory and CPU usage by the state-of-the-art analysis module. For IDEAFIX accuracy, the percentage of false positives (flows unduly identified as elephants) and false negatives (elephant flows not identified), analyzed in function of memory space to hash mapping on switches. For all metrics, lower values are better.

D. Results

Figure 5 shows the results for the threshold violation/-surpassing reaction time to elephant flows. The approaches evaluated are plotted in the x-axis. The state-of-the-art analysis module requests the flow estimates to the sFlow collector at a frequency of 0.01s, 0.1s and 0.5s, and sampling rate 1/128 and 1/1024. In the y-axis, the average reaction time (in milliseconds) for each of the approaches evaluated is presented in logarithmic scale. IDEAFIX can identify and react to an elephant flow in 0.4ms, *i.e.*, basically the packet processing time of the software-emulated P4 switch. In a hardware switch, packet processing time would be in nanoseconds.

[1]https://github.com/p4lang/behavioral-model
[2]https://inmon.com/products/sFlow-RT

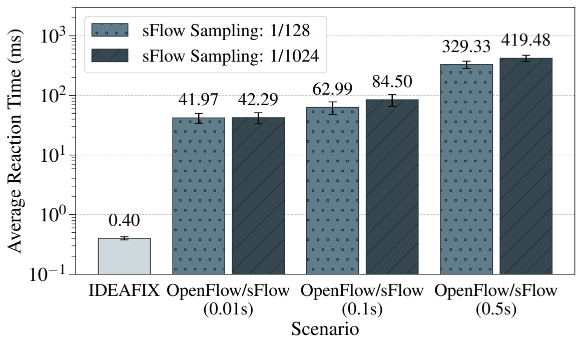


Fig. 5. Threshold violation/surpassing reaction time.

The average reaction time in the state-of-the-art mechanism varies between 41.97ms in the best case and 419.48ms in the worst case. The confidence intervals for packet sampling rates at 1/128 and 1/1024 overlap at their corresponding levels. Thus, with 95% confidence level there is no significant difference. This behavior was expected because of the sFlow collector flow bandwidth estimate in different sampling rates.

Figure 6 presents the volume of data that exceeded the thresholds and continue to be routed through the default path until the reaction occurred. In IDEAFIX, there is no excess data because the reaction is immediate with the packet processing that characterizes the elephant flow. However, in the state-of-the-art mechanism, there is a variation between 64.27 KB in the best case and 524 KB in the worst case, when the collector query interval is 0.01s and 0.5s, respectively.

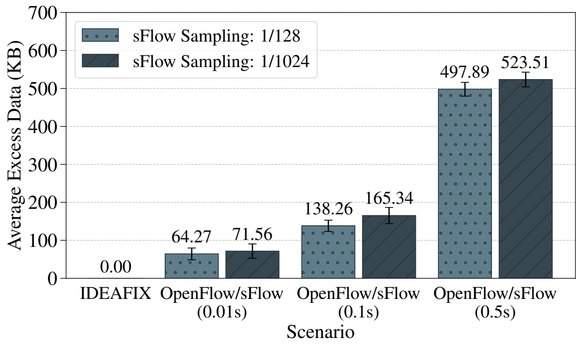


Fig. 6. Excess data.

These values are a consequence of the communication time between the switches and the control plane, and the time communication between the identification module and the sFlow collector. When this interval is short (*e.g.*, 0.01s) the reaction time and excess data decrease. However, the resources utilization by the analysis module in the state-of-the-art mechanism increases, as shown in the Table I. When this interval is long (*e.g.*, 0.5s), the computational cost is relatively small, although the reaction time and excess data are larger. When the interval is 0.1s, there is a computational cost close to that obtained with 0.5s, but the gains in the other metrics come close to that obtained with 0.01s.

TABLE I
USE OF RESOURCES BY THE IDENTIFICATION MODULE.

Query Interval (s)	0.01	0.1	0.5
CPU Used	27%	7%	3%
Memory Used (MB)	67.1	67.1	67.1

The data inserted into the network with the monitoring mechanism is shown in Figure 7. The IDEAFIX sends only a notification informing the identified flow to the control plane, a packet of the 96 bytes. Thus, the size of monitoring data is directly proportional to the amount of identified elephant flows. In the experiments, 2048 flows were inserted into the network per round, being approximately 256 elephant flows. The monitoring data on the network was 24.57 KB.

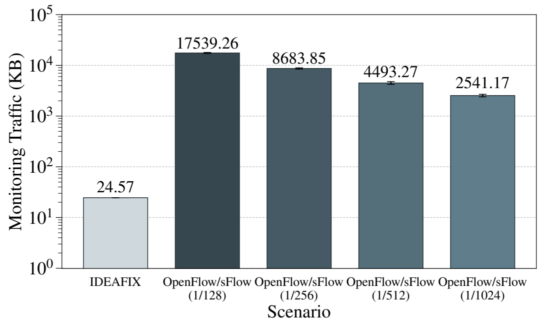


Fig. 7. Monitoring data.

In the state-of-the-art mechanism, this value depends on the packet sampling rate. When sampling 1/128, there is about 17 MB of monitoring data, and with a sampling of 1/1024, there is about 2.5 MB. This represents a difference of approximately 14.5% between these two sampling rates. There is a gradual increase in sampling levels.

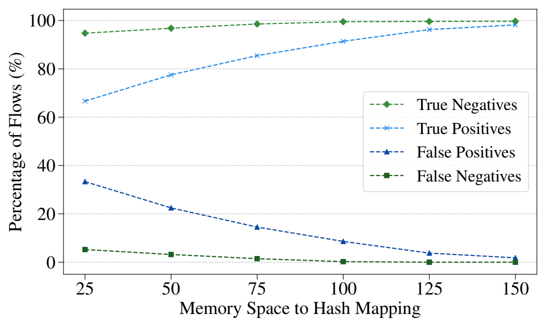


Fig. 8. IDEAFIX accuracy.

The IDEAFIX accuracy (Figure 8) was evaluated by varying the memory space to hash mapping on the switches in relation to the number of flows inserted in the network (*i.e.*, 2048). With 25% of available space, there is a percentage of about 5% of false negatives and 35% of false positives. This is a result of the strategy that overestimates the flows, wherein false positives are preferred over false negatives when hash key collisions occur. When the hash mapping space increases (*e.g.*, 100%), false positives and false negatives decrease (*i.e.*, less than 10% and 1%, respectively) and true positives and true negatives increase, about 95% of the accuracy.

Finally, Table II shows the average (μ) packet processing time, in milliseconds, on the P4 switches. That is an increase of approximately 30% compared with traditional processing, which only performs packet forwarding. This is the packet processing time of the software-emulated P4 switch. In a hardware switch, this would be in nanoseconds. The confidence intervals (ε) do not overlap with a confidence level of 95%.

TABLE II
PACKET PROCESSING TIME (ms).

witch P4	$\mu - \varepsilon$	μ	$\mu + \varepsilon$
With IDEAFIX	0.256	0.279	0.302
Only forwarding	0.205	0.216	0.227

E. Summary

The results demonstrate that IDEAFIX can identify and react to elephant flows quickly and cost-efficiently, inserting few monitoring data in the network compared to an approach that performs flow sampling with sFlow. However, more memory space is required on the switches for the engine (hash mapping) and in the routing tables (preventive strategy). Thus, if the network operator prioritizes the immediate identification of the elephant flows, IDEAFIX is most indicated in relation to an approach that involves the control plane in the identification loop. Otherwise, the network operator may not prioritize the identification time, either by memory requirements or by already having deployed in the infrastructure a mechanism with sFLow. In this case, the results of this paper indicate that it is possible to identify and react to elephant flows by combining the sampling rate and query rate parameters to the sFlow collector, to incur better use of resources, balancing the monitoring data in the network and the identification time.

V. CONCLUSION

Elephant flows identification in IXPs networks can promote a better use of the services provided to its participants. Thus, this paper presents IDEAFIX, a mechanism that allows performing the elephant flow analysis and identification directly in IXP networks using P4 switches.

IDEAFIX was shown to be significantly more efficient than the state-of-the-art mechanism [1] [7], in which the analysis process involves the control plane. The switch-controller communication delays the identification process, since monitoring data needs to be transmitted to the controller before any processing can be done, and only after that the new routing decisions can be sent back to the switches. In contrast, in IDEAFIX every process of analysis and identification is performed directly on the switches. Thus, each packet may be analyzed when it enters the network, allowing immediate identification of packets that exceed the size and duration thresholds that characterize a flow as an elephant one. The results demonstrate that it is possible to identify and react to these flows quickly and cost-efficiently, adding a significantly smaller amount of monitoring data in the network in relation to the mechanisms that use flow sampling. Lastly, IDEAFIX allows adjusting the memory space dedicated to the identification process according to the desired accuracy. As future work, we are studying methods based on machine learning and pattern recognition to define the thresholds dynamically.

ACKNOWLEDGEMENT

We thank CNPq for the financial support. This research has been supported by call Universal 01/2016 (CNPq), project NFV Mentor process 423275/2016-0.

REFERENCES

[1] L. A. D. Knob, R. P. Esteves, L. Z. Granville, and L. M. R. Tarouco, "SDEFIX—Identifying elephant flows in SDN-based IXP networks," in *Network Operations and Management Symposium (NOMS), 2016 IEEE/IFIP*. IEEE, 2016, pp. 19–26.

[2] B. Augustin, B. Krishnamurthy, and W. Willinger, "IXPs: mapped?" in *Proceedings of the 9th ACM SIGCOMM conference on Internet measurement conference*. ACM, 2009, pp. 336–349.

[3] J. C. Cardona Restrepo and R. Stanojevic, "IXP traffic: a macroscopic view," in *7th Latin American Networking Conf.* ACM, 2012, pp. 1–8.

[4] E. Gregori, A. Improta, L. Lenzini, and C. Orsini, "The impact of IXPs on the AS-level topology structure of the Internet," *Computer Communications*, vol. 34, no. 1, pp. 68–82, 2011.

[5] L. A. D. Knob, R. P. Esteves, L. Z. Granville, and L. M. R. Tarouco, "Mitigating elephant flows in SDN-based IXP networks," in *IEEE Symp. on Computers and Communications (ISCC)*, 2017, pp. 1352–1359.

[6] L. Guo and I. Matta, "The war between mice and elephants," in *Ninth International Conf. on Network Protocols.* IEEE, 2001, pp. 180–188.

[7] A. R. Curtis, J. C. Mogul, J. Tourrilhes, P. Yalagandula, P. Sharma, and S. Banerjee, "DevoFlow: Scaling flow management for high-performance networks," *ACM SIGCOMM Computer Communication Review*, vol. 41, no. 4, pp. 254–265, 2011.

[8] J. Suh, T. T. Kwon, C. Dixon, W. Felter, and J. Carter, "Opensample: A low-latency, sampling-based measurement platform for commodity sdn," in *IEEE 34th International Conference on Distributed Computing Systems (ICDCS)*, 2014, pp. 228–237.

[9] sFlow, "sflow.org," http://www.sflow.org, 2015.

[10] N. McKeown, T. Anderson, H. Balakrishnan, G. Parulkar, L. Peterson, J. Rexford, S. Shenker, and J. Turner, "Openflow: enabling innovation in campus networks," *ACM SIGCOMM Computer Communication Review*, vol. 38, no. 2, pp. 69–74, 2008.

[11] V. Sivaraman, S. Narayana, O. Rottenstreich, S. Muthukrishnan, and J. Rexford, "Heavy-hitter detection entirely in the data plane," in *Symposium on SDN Research.* ACM, 2017, pp. 164–176.

[12] R. Basat, G. Einziger, R. Friedman, M. Luizelli, and E. Waisbard, "Constant time updates in hierarchical heavy hitter," *In Proceedings of SIGCOMM '17, LA, CA, USA*, pp. 127–140, 2017.

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

[14] M. V. B. Silva, J. A. Marques, L. P. Gaspary, and L. Z. Granville, "Identificação de Fluxos Elefantes em Redes de Ponto de Troca de Tráfego com Suporte à Programabilidade P4 [in Portuguese]," *36th Simpósio Brasileiro de Redes de Computadores e Sistemas Distribuídos (SBRC)*, vol. 36, May 2018.

[15] M. Afaq, S. Rehman, and W.-C. Song, "Visualization of elephant flows and QoS provisioning in SDN-based networks," in *17th Asia-Pacific Network Operations and Management Symposium (APNOMS).* IEEE, Aug 2015, pp. 444–447.

[16] S. Geravand and M. Ahmadi, "Bloom filter applications in network security: A state-of-the-art survey," *Computer Networks*, vol. 57, no. 18, pp. 4047–4064, 2013.

[17] AMS-IX. (2015) Amsterdam Internet Exchange Infrastructure. https://ams-ix.net/technical/ams-ix-infrastructure.