Mitigating Elephant Flows in SDN-Based IXP Networks

Luis Augusto Dias Knob*‡, Rafael Pereira Esteves*†,
Lisandro Zambenedetti Granville*, Liane Margarida Rockenbach Tarouco*
*Institute of Informatics – Federal University of Rio Grande do Sul
Av. Bento Gonçalves, 9500 – Porto Alegre, Brazil
†Federal Institute of Rio Grande do Sul - Campus Restinga
Rua Alberto Hoffmann, 285 – Porto Alegre, Brazil
‡Federal Institute of Rio Grande do Sul - Campus Sertao
Rodovia RS 135, Km 25 – Sertao, Brazil
Email: {luis.knob, rpesteves, granville}@inf.ufrgs.br, liane@penta.ufrgs.br

***Abstract*—Internet Exchange Points (IXPs) play a key role in the Internet architecture, enabling cost-effective connections among multiple autonomous systems (ASes). The management of IXP networks includes the activity of taking care of elephant flows; they represent a small number of the total flows of an IXP, but have high impact on the overall network traffic. Managing elephant flows involves adequate identification and eventually rerouting of such flows to more appropriate locations, to minimize the possible negative impact on the other (mice) flows. Elephant flow management becomes even more important in SDN-based IXPs that require controllers to have a consistent view of the underlying network to allow fine-grained adjustment. In this paper, we propose, develop, and evaluate a recommendation system to suggest alternative configurations to previously identified elephant flows in an SDN-based IXP network. In our solution, the IXP operator can define templates that ultimately define how elephant flows can be rerouted to achieve a specific objective. We demonstrate that our system can help IXP operators to mitigate the impact of elephant flows on the IXP network.**

***Keywords*-Software Defined Networking, Internet Exchange Points, Network Management**

I. Introduction

Accounting for at least 20% of all traffic exchanged among Autonomous Systems (ASes), Internet Exchange Points (IXPs) play a critical role in the Internet ecosystem, both with regards to the communication between different parties (*e.g.*, content providers, CDNs, ISPs), and in the relation between content providers (such as Google [1], Facebook [2], and Netflix [3]) and end-users [4]. Recently, a number of solutions for IXP management based on Software-Defined Networking (SDN) has been proposed, mainly to overcome the difficulties inherent to the Border Gateway Protocol (BGP) [5] and to instantiate functionalities that were hard or even impossible before, such as application-specific peering [6], inbound traffic engineering [7], wide-area server load balancing [8], and upstream blocking of DoS attacks [9] [10].

Usually entirely based on layer-2 switches, IXP networks can be as complex as a single device or present quite intricate topologies too [11]. In such networks, identification of the so-called elephant flows [12] is critical to assist the IXP operator to, for example, spot links that form bottlenecks and, through the IXP's SDN controller, engineer the network load by moving elephant flows from their original busier paths to more appropriate ones. Although several works present solutions for elephants flows identification [13] [14] [15], only a few develop methods to handle these flows along the network [7]. According to our knowledge, there is no solution today to manage elephant flows in IXP networks by exploiting SDN benefits.

In this paper, we introduce an SDN-based IXP management solution that enables the IXP operator to define how to distribute elephant flows along the IXP network. In a previous work [16], we exploited how sFlow and OpenFlow can be used to snapshot the IXP network to identify the current elephant flows. In this paper, we enable the network operator to handle the identified elephant flows by using SDN rules. Our solution includes a recommendation system that, by employing templates (*e.g.*, Optimize Links Consumption, Quality of Service, and Discard Packages) assists the IXP operator in the mitigation of elephants flows effects over his/her IXP. Our recommendation system suggests SDN rules over the IXP network that can be tuned by the operator, thus delivering a refined yet human-centered control loop.

The remainder of this paper is organized as follows. In Section 2, we discuss some relevant concepts and review related work on the management of elephant flows in both traditional and SDN networks. In Section 3, we present our proposed solution and report on the implementation of an associated system prototype. Our solution is then evaluated in Section 4. We finally close this paper presenting concluding remarks and future work in Section 5.

II. Background and Related Work

In this section, we review the main concepts related to our work, focusing on the management of IXPs and on the use of SDN in IXP management. Then we present and discuss some prominent related work in the area.

A. *Management of IXPs and Elephant Flows*

Internet eXchange Points (IXPs) are facilities composed of networking equipment (*i.e.*, switches, routers) enabling cost-effective traffic exchange between autonomous systems (ASs). The management of IXPs includes monitoring networking resources to identify bottlenecks, handle failures, and enforce traffic engineering policies [17] [18]. A management aspect that gains more importance in IXP networks is the management of the so-called *elephant flows*. Elephant flows are typically small in number but are characterized by having high throughput and long duration compared to other (mice) flows [19] [12]. Because elephant flows are long-lived and rapidly consume network buffers they usually impose undesirable queuing delays to mice flows. Managing elephant flows is important because of the potential impact of such flows in the IXP network. Therefore, mitigating the negative influence of elephant flows over mice flows and improving overall network utilization require appropriate management actions from the IXP operator.

Possible management actions to mitigate the impact of elephant flows in the IXP network can be the application of traffic engineering mechanisms over such flows. Traffic engineering (TE) can be realized by various techniques such as admission control, bandwidth enforcement, and traffic classification, which were employed in ATM networks [20]. Admission control allows the network to accept or reject a request based on its requirements, bandwidth enforcement is used to avoid traffic volume exceeds a determined level, while traffic classification allows the network to support different traffic requirements. TE mechanisms for IP networks include mechanisms to guarantee QoS based on the Diffserv and IntServ architectures and to enforce load balancing through Equal Cost Multipath (ECMP) [21]. MPLS-based traffic engineering through explicitly routing attempted to overcome the limitations of TE mechanisms proposed for IP networks. By using LSP (Label Switching Paths) tunnels between source and destination traffic is not restricted a limited set of paths, which improves the overall network utilization and [22] [20].

Although the aforementioned TE mechanisms represented important advances when they were proposed, the complexity of today's networks hinder their benefits. Modern networks are difficult to manage because of the high level of heterogeneity and distribution in the control, data, and management planes [20]. In this regard, SDN allows flexible and efficient management of large scale networks, such as IXP networks.

B. *SDN-based traffic engineering*

Software-Defined Networking (SDN) [23] [24] has been considered a viable alternative to tackle several issues in the networking area. By decoupling the control plane from the forwarding plane, SDN empowers operators with full control of forwarding devices. Recently, the management of SDN and the use of SDN as a management tool have gained importance in the research community [25] [26].

The logically centralized control provided by SDN facilitates the implementation of traffic engineering mechanisms because SDN offers a complete view of the network to the IXP operator, allows the IXP operator to program networking devices and adjust traffic flow to meet particular needs, and provides a open interface to networking devices enabling interoperability [20]. In an SDN-operated IXP, the controller can be fed with information about the IXP's elephant flows. With such information, a set of SDN applications can optimize the IXP network according to some objectives, *e.g.*, reduce power consumption or decrease elephant flow traversal delay.

C. *Related Work*

In this section, we review the state-of-the-art on SDN-based IXPs and management of elephant flows.

SDX [6] uses SDN concepts to build a software-defined IXP to overcome the limitations of the BGP protocol, namely routing based on the destination prefix, limited application of policies, and indirect path selection. In SDX, participant ASes run SDN applications in a virtual SDN switch abstraction. SDX then combines the policies generated by the ASes in low-level policies that are deployed in the infrastructure. Although SDX allows ASes to perform wide-area load balancing, it does not handle elephant flows in the core IXP network explicitly.

Mahout [27] is a solution to reduce monitoring overhead in switches by detecting elephant flows at end hosts in a data center network. Mahout requires the inclusion of a shim layer at the operation system of the end host, which is not applicable in IXPs. DevoFlow [28] uses OpenFlow [29] to keep track of elephant flows only, while offering different mechanisms to computer statistics, such as sampling and triggering. In DevoFlow, if a flow reaches a predefined threshold in terms of number of bytes, the flow is classified as an elephant one and a decreasing best-fit bin packing algorithm is applied to calculate the least congested path between the flow's endpoints. However, DevoFlow does not allow the network operator to define the criteria used to select alternative paths for elephant flows.

ESHSP (Elephant Sensitive Hierarchical Statistics Pulling) [14] proposes an iterative process to detect elephant flows by decomposing the flow space until an elephant flow is isolated from the others. ESHSP uses a combination of aggregate and individual statistics messages of OpenFlow to reduce bandwidth consumption. Flow detection is enhanced through two additional functions: elephant store avoids having the same elephant counted multiple times, while range splitting makes flow blocks as equal as possible. The limitation of ESHSP is that it focuses only on elephant flow detection and does not perform further actions with such flows, which can lead to network congestion.

Similar to our proposal, OpenSample [30] relies on sFlow for monitoring and enables traffic engineering in SDNs. OpenSample relies on TCP sequence numbers to improve accuracy of elephant flow detection without requiring a large number of samples. In OpenSample, flow rates are calculated by subtracting the TCP sequence numbers of two samples of the same flow and dividing by the time between the samples. After a flow is classified as elephant, it is rerouted to another path using a global first fit algorithm. Although flow scheduling is not the focus of OpenSample, traffic engineering is restricted to a simple first fit algorithm and the network operator is not able to define how elephant flows are rerouted.

Afaq *et al.* [7] explore the Enqueue action of OpenFlow to apply traffic shaping to elephant flows in order to achieve QoS in terms of bandwidth. Authors set rate limited queues to the ports of an OpenFlow-based switch to handle elephant flows. With such traffic shaping an elephant flow can be either routed through a high bandwidth queue or reshaped in case the available bandwidth is not enough to handle the flow. The problem with such approach is that elephant flow management is performed only at the switch level without considering other switches that could better accommodate elephant flows, which limits its benefits. Besides, the IXP operator may be interested in other objectives such as low latency and load balancing.

In summary, current proposals for elephant flow management present limitations in the context of SDN-based IXPs. Most solutions were proposed in the context of data center networks, taking advantage of the path diversity typically found in such networks and leveraging the centralized control provided by SDN. We believe that IXP networks can also benefit from SDN-based solutions to tackle the elephant flow phenomenon. Also, the IXP operator may want to employ diverse traffic engineering mechanisms to achieve distinct performance objectives according to his/her needs, which is not possible in current flow management solutions. In the next section, we introduce and detail our proposed solution for mitigating the impact of elephant flows in the IXP network.

III. The SDEFIX Recommendation System

We developed a recommendation system that is integrated into the SDEFIX elephant flow identification system [16]. The recommendation systems extends SDEFIX allowing IXP operators to mitigate the impact of elephant flows in the IXP network by applying traffic engineering mechanisms in the form of *templates*, which ultimately define how elephant flows are managed in the IXP network. We consider an environment where the IXP network is controlled by one logically centralized SDN controller that establishes the physical paths used for communication inside the IXP. Switches communicate with the controller through the OpenFlow protocol [29], which is the most popular SDN implementation.

A. System Architecture

We use elephant flow snapshots generated by SDEFIX as input to the recommendation system. SDEFIX samples the network (via sFlow) to identify elephant flows and their associated paths in the SDN infrastructure. That is done periodially, generating snapshots of the network that report the identified elephant flows present in the network at the moment the snapshot has been taken. The last snapshot, obviously, report the last seen elephant flows. Figure 1 presents the architecture of our recommendation system. New components are depicted with bolder black borders; previous SDEFIX components are drawn with lighter, gray boders. Our discussion here will be centered on the new components, which encompass the recommendation features; the interested reader can observe the details of old components in our previous paper [16].

Three new modules and one database have been incorporated into the SDEFIX architecture: *Recommendation framework*, *Rules installer*, *Template manager*, and *Templates* database. The *Recommendation framework* module applies *templates* created by the IXP operator, which encapsulate traffic engineering mechanisms (*e.g.*, rerouting algorithms), over the elephant flows reported in a network's snapshot. Once the IXP operator chooses a template, the framework applies it to all the existing elephant flow paths of a given snapshot of the IXP network. One template can be active in the recommendation framework but the IXP operator can switch between templates according to his/her needs. The result of the application of a template over a snapshot is a recommended set of SDN (OpenFlow-like) rules that can be deployed in the IXP.

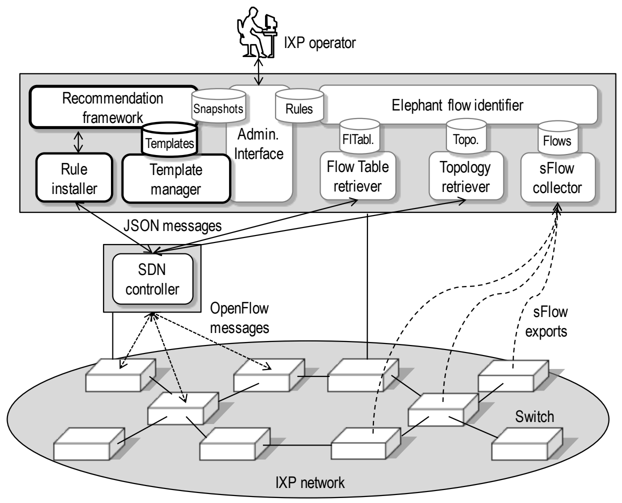

Fig. 1. SDN-based IXP Recommendation System

The IXP operator may want to adjust a recommended rule before applying it to the network, for example, by removing specific switches/links from a suggested path and running the template again, with the operator's modifications fixed. The *Template manager* module is where the IXP operator adds different templates to be used to achieve different objectives, such as load balancing, improved link utilization, and bandwidth/delay guarantees. Finally, the *Rules installer* module is responsible for installing the flow rules on the SDN controller after the IXP operator tunes the template results. The SDN controller then installs the rules on the appropriate switches using the OpenFlow protocol.

The SDEFIX recommendation system uses elephant flow snapshots as input. However, the operator may indicate which elephant flows from a whole snapshot are expected to be observed again a recommendation template, *i.e.*, recommendations can be about either a whole snapshot or a subset of it. Moreover, by default, the system generates recommendations based on the last snapshot, but this can be modified to allow recommendations to be generated against other previously stored snapshot. Figure 2 shows a sequence of how SDEFIX recommendation flow works.

After the IXP operator requests a recommendation indicating the elephant flow snapshot he/she wants to manage, the *Recommendation Framework* module forwards the request to the *Template manager* module along with the template to be used. The *Template manager* can retrieve information from other modules such as the network topology from the *Topology retriever* component. Once the template is evaluated, the *Template manager* returns a recommendation in the form of OpenFlow rules that is confirmed by the IXP operator and modified if necessary. Afterwards, the recommendation is finalized and sent to the *Rules installer* module, which forwards the rules to the SDN controller though a JSON API. Finally, the system indicates to the IXP operator if the rules were successfully installed or not.

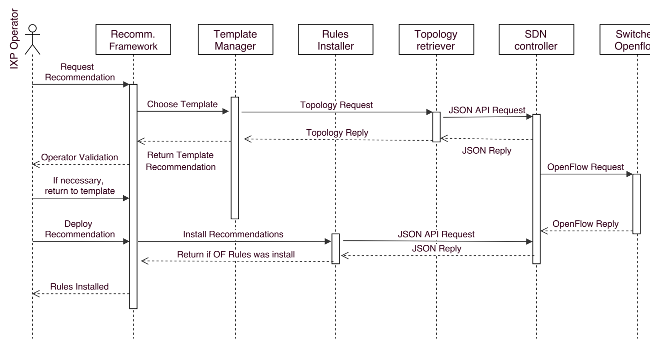

Fig. 2. Sequence diagram of the SDEFIX recommendation system

Algorithm 1 illustrates how the recommendation system works. The algorithm is triggered by the IXP operator when he/she wants to reallocate elephant flows that are active in the IXP network. First, the operator chooses one of the templates available in the system. An example of such template is shown in Algorithm 2. Thereafter, the template is applied for each elephant flow path present in the snapshot (lines 9-13). For each generated recommendation, the operator can make changes to the result of the recommendation in order to remove any undesirable characteristic from it, such as paths having high latencies, and rerun the chosen template (lines 14-20). Once the recommendation is confirmed by the IXP operator, the rules are installed to the SDN controller (lines 17-18).

To illustrate the SDEFIX recommendation system, we created a template that selects the second best route for an elephant flow aiming at two goals: (i) alleviate the impact of elephant flows over the mice flows, releasing default (shortest) paths for mice flows, and (ii) reroute elephant flows to alternative paths to improve the overall utilization of the IXP network. This template is named BAP (Best Alternative Path) and is depicted in Algorithm 2. The main idea of BAP is to find the best alternative path for a given elephant flow path by increasing the weight of the links of the original path and recomputing the shortest path between the source and destination endpoints of the elephant flow. Figure 3 summarizes the BAP template.

Algorithm 1 Recommendation approach

```
 1: $SNP$: set of snapshotted elephant flows
 2: $T$: set of configuration templates
 3: $P$: physical topology
 4: $l(i, j)$: link between switches $i, j \in P$
 5: $F(sw)$: flow table of the switch $sw \in P$
 6: $R(snp)$ recommendations for elephant flow path $snp \in SNP$
 7: **Generate recommendations**:
 8: choose a template $t \in T$
 9: **for** each $snp \in SNP$ **do**
10:     $r \leftarrow ApplyTemplate(t, snp)$
11:     add $r$ to $R(snp)$
12:     $ApplyRecommendation(R(snp))$
13: **end for**
14: **procedure** APPLYRECOMMENDATION(R(snp))
15:     choose a recommendation $r \in R(snp)$
16:     perform any modifications to $r$
17:     **for** each $l(i, j) \in r$ **do**
18:         install rules to $F(i)$, $F(j)$
19:     **end for**
20: **end procedure**
```

We can divide template execution in three parts: (i) loading of objects that are required by the template, such as the network topology and the snapshot of the elephant flows to be managed; (ii) execution of the template logic, which in the case of BAP corresponds to the adjustment of the weights of the links and the calculation of the second shortest path; and (iii) the validation of the generated recommendation by the IXP operator that, in his/her turn, can modify the topology and run again the template if necessary.

Algorithm 2 Example of ApplyTemplate function - Best Alternative Path (BAP)

```
$snp$: snapshotted elephant flow
$P$: physical topology
$l(i, j)$: link between switches $i, j \in P$
**function** APPLYTEMPLATEBAP(snp)
    $s \leftarrow$ source endpoint of $snp$
    $d \leftarrow$ destination endpoint of $snp$
    **for** each $l(i, j) \in snp$ **do**
        increase weight of $l(i, j)$
    **end for**
    **return** shortest path between $s$ and $d$
**end function**
```

We use the Ryu OpenFlow controller [31] in our implementation, since it is one of few controllers that supports the latest OpenFlow versions and implements a well-defined REST API. Nevertheless, our solution is decoupled from any particular SDN controller and can be ported to other modern controllers, such as OpenDaylight [32].

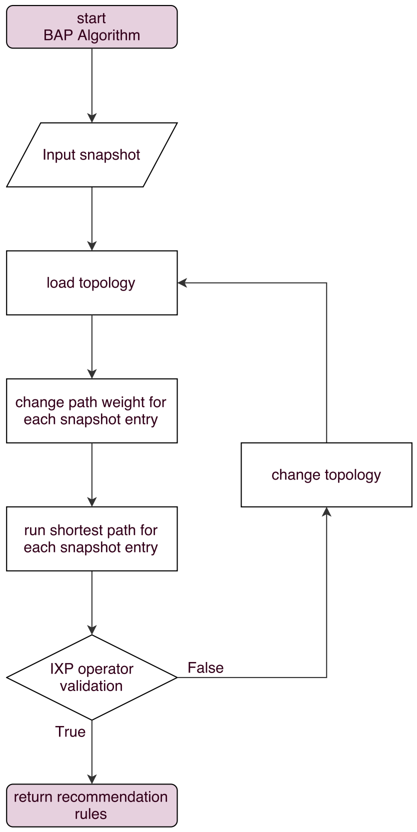
start
BAP Algorithm
Input snapshot
load topology
change path weight for each snapshot entry
run shortest path for each snapshot entry
IXP operator validation
False
True
change topology
return recommendation rules

Fig. 3. Flowchart of the BAP template

B. Templates Specification

In our solution, the IXP operator can add or create templates to reduce the impact of the elephant flows in the network. Common solutions can include new routing strategies, quality of service enforcement, or optimized energy management.

The template used to exemplify the proposed recommendation framework is presented in Algorithm 2. It computes the best alternative shortest path for every elephant flow stored in SDEFIX. The idea is to migrate an elephant flow to a new, possibly not congested path, while not significantly increasing network latency. This is done by increasing the weights of the links belonging to the original path of the snapshotted elephant flows (Algorithm 2, line 8) and recomputing the elephant flow path using a shortest path first algorithm. In this way, the IXP operator expects that elephant flows are rerouted to alternative, possibly not congested, links of the IXP network.

Templates are written in Python and use a well-defined API as shown in Figure 4. There are classes that provide methods that are useful to the IXP operator who is writing a template. For example, information about the physical topology can be obtained through the methods getLinks, getSwitches, and getPorts of the Topology class. Similarly, the OFRulesInstaller class provides methods to allow the installation of new rules or modifications in the rules currently installed in an OpenFlow controller. A complete example of the BAP template can be found here[1].

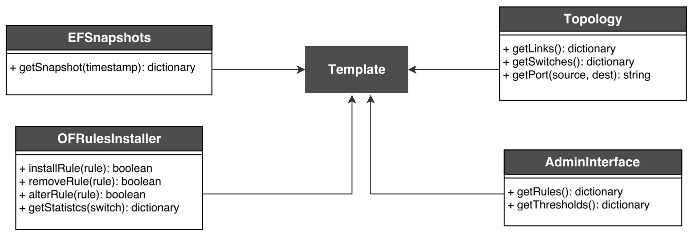
EFSnapshots
+ getSnapshot(timestamp): dictionary
Template
Topology
+ getLinks(): dictionary
+ getSwitches(): dictionary
+ getPort(source, dest): string
OFRulesInstaller
+ installRule(rule): boolean
+ removeRule(rule): boolean
+ alterRule(rule): boolean
+ getStatistcs(switch): dictionary
AdminInterface
+ getRules(): dictionary
+ getThresholds(): dictionary

Fig. 4. Template API

C. IXP Operator Validation

An essential part of the framework is the interplay between the IXP operator, the suggested recommendations, and their deployment in the network. In an infrastructure where each change can bring consequences, such as traffic loops, BGP misconfigurations, pilot errors, and service loss besides requiring the operator to manage several SLAs, fine-grained control over every action is required and fully automated decisions must be constantly monitored in order to ensure proper IXP operation.

Thus, recommendations proposed must be validated by the IXP operator before they are applied to the IXP network. This validation may add new parameters to the template. For example, in the alternative shortest path template, the IXP operator can increase the weight of paths that he/she does not want to use, for any reason, such as to avoid high latencies, or to exclude specific network equipments to avoid loops. If any modification is performed in the recommendation returned by the template, the template should be run one more time and has to be revalidated. After the operator validates the template, the rules are then installed to the SDN controller.

It is important to point out that the objective of the SDEFIX recommendation system is to allow flexible template definition that ultimately defines how elephant flows are rerouted in the IXP network. Best alternative path is an example of template that can be used. Other templates can be written and applied to the IXP network depending on the goal the IXP operator wants to achieve. Because it is not possible to predict all types of templates one can write and the physical topology can also vary, in our solution, the IXP operator is the one responsible for writing correct templates and validating them. For example, for BAP the IXP operator has to review suggested paths in order to avoid loops.

D. Example

To illustrate the use of the SDEFIX recommendation system we provide a simple example. The snapshot used as input for the recommendation system is shown in Figure 5. The elephant flow enters the IXP network at a switch identified as DPID 101 and terminates at switch DPID 105. We simulate the application of the Best Alternative Path Template (Algorithm 2) by the IXP operator. After the template is applied, it

[1]https://bitbucket.org/luisdknob/sdefix-web

generates the recommendation illustrated in Figure 6 that removes a hop of the path traversed by the elephant flow. The IXP operator can apply the recommendation without any modification or can make changes to the outcome such as removing a link or a switch and then run the template again.

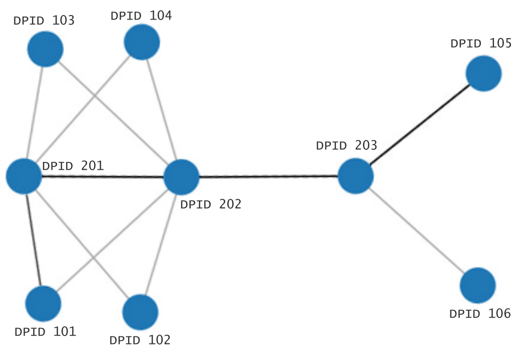

Fig. 5. Snapshot used as input

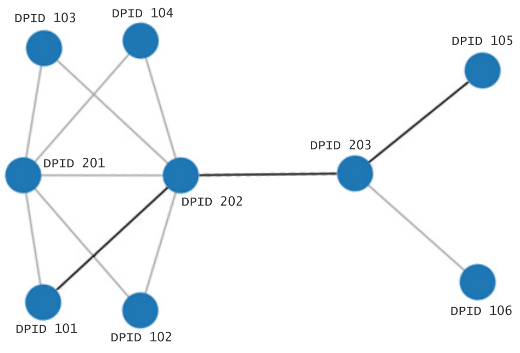

Fig. 6. Elephant flow after template validation

With respect to the effort that an IXP operator must employ to use our proposed system, he/she can create templates that ultimately defines how elephant flows are placed in the IXP network. The system allows the IXP operator to write and/or modify a template using Python-based scripts and a set of auxiliary functions. At this point, the IXP operator has to load snapshots, run templates, and install rules to the SDN controller using simple scripts. In the next section, we evaluate the performance of our proposed recommendation system.

IV. Evaluation

In this section, we evaluate the performance of SDEFIX recommendation system in mitigating the impact of elephant flows in the IXP network. We begin by describing the experimental scenario and the methodology used in the tests. Following that, we present and discuss the main results achieved so far.

A. Scenario

The evaluation scenario is a simplified version of the AMS-IX infrastructure [33] replacing AMS-IX's original MPLS routers by SDN-enabled switches as shown in Figure 7. We emulate the topology using the Mininet emulator [34] running in a virtual machine (VM) with 2GB of RAM. OpenVSwitch [35] is used for the swtiches since it supports the latest OpenFlow implementations. The Ryu SDN controller is hosted in a second VM, while SDEFIX and the recommendation system run in a third VM. The VMs are hosted in a Core i7 4790 server with 16 GB of RAM through VirtualBox. In this scenario there are 8 ASes with 32 hosts generating traffic inside each one.

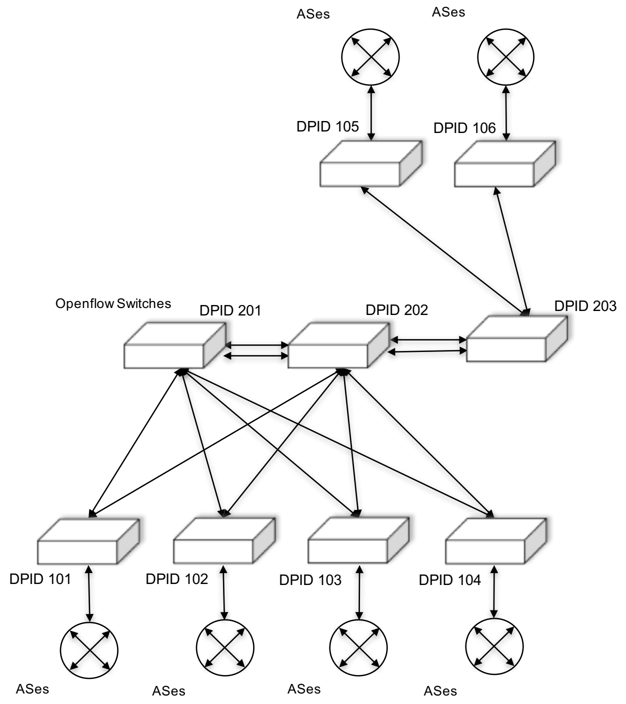

Fig. 7. Topology used in the experiments

We evaluate the SDEFIX recommendation system in terms of (1) *mitigation time* and (2) *traffic peak*. The mitigation time is the total time taken by the recommendation system to generate and apply a recommendation to the IXP network regarding elephant flows. The traffic peak reflects the highest load in each switch of the IXP network. Each test was repeated 30 times with a confidence interval of 95%. In the next subsection, we present the main results of our evaluation.

B. Results

To better illustrate the results regarding mitigation time we decompose the mitigation process in three distinct phases. The first phase is the Snapshot Load Phase, where all snapshots of previously identified elephant flows are loaded in the system. On the Template Execution Phase the chosen template (BAP) is applied to the snapshots and generates recommendations. After template validation, rules are then installed in the SDN controller to reflect a recommendation. In this experiment, the number of elephant flows is 128 and the total number of flows is 1408. Elephant flows have a mean throughput of 10 Mbps, while short (mice) flows have throughput varying from 400 Kbps to 1 Mbps. Figure 8 depicts the time taken in each phase separately.

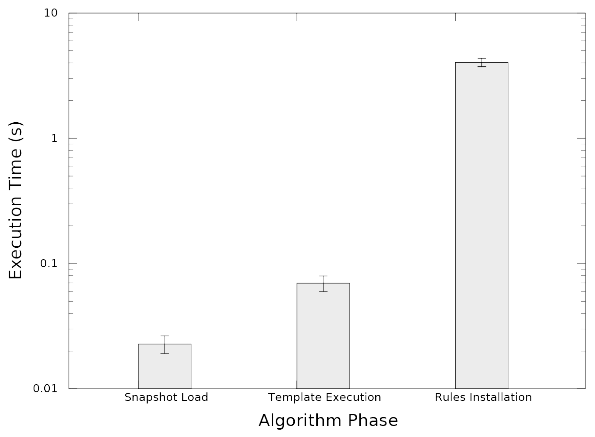

Fig. 8. Mitigation time

It is possible to observe from Figure 8 that the snapshot load time is much smaller (20 milliseconds) if compared to template execution and rule installation. Template execution is highly dependent on the strategy used and may have different performance for different templates. In our case, the BAP template takes less about 70 milliseconds to generate a recommendation, which can be considered small for a large IXP. The rules installation time accounts for most of the time taken in the mitigation approach because it is dependent on the number of rules that have to be installed. Since our scenario is very dense in terms of number of mice and elephant flows, it was expected that the recommendation system generated a high number of rules to be installed in the SDN controller. In our experiments, 650 rules were installed in about 4 seconds, thus each rule was installed in about 6 miliseconds in average.

Figure 9 shows the traffic peak observed in each switch of the IXP network after the application of the BAP template compared to the traffic peak when no template is used, that is, paths are computed using default routing mechanisms that try to place flows in the shortest path available. In order to better visualize the impact of the recommendation system in mitigating elephant flows, we have reduced the number of elephant flows to 16 and the total number of flows (mice + elephant) to 176. In this experiment, elephant flows have a mean throughput of 10 Mbps, while mice flows have throughput varying from 400 Kbps to 1 Mbps Mbps. Switches are identified by DPID N, where N is the number of the switch. DPID 201, DPID 202, and DPID 203 are the core switches, whilst DPID 101 to DPID 106 are edge switches in the IXP network (refer to Figures 5 and 6) .

The effect of the application of the BAP template can be better noticed in the core switches that handle most of the

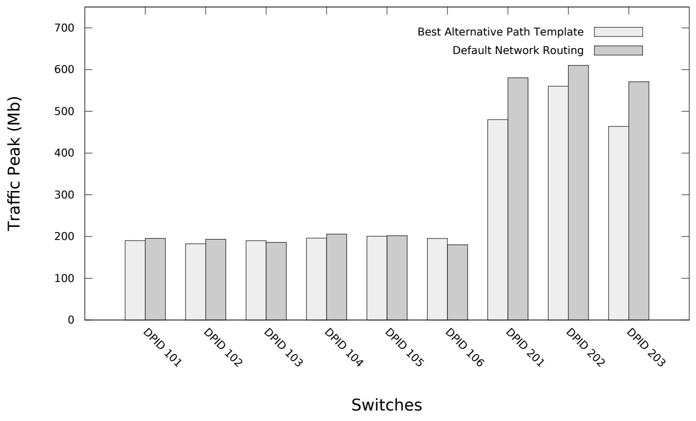

Fig. 9. Traffic peak in switches

traffic of the IXP network. The reduction in traffic peak was of 17%, 8%, and 19% in average for DPID 201, DPID 202, DPID 203, respectively. The reduction in traffic peak is not higher in core switches mainly because of the small number of alternative links connecting these switches. Nevertheless, the SDEFIX recommendation system was able to improve resource utilization even in constrained scenarios.

C. Summary

In summary, there are aspects regarding the performance of the SDEFIX recommendation system that are highly coupled with the template defined by the IXP operator to handle elephant flows. With respect to mitigation time, we can separate the amount of time spent in the execution of the template, which depends on the template being used, from the times spent during snapshot loading and installation of a single rule, which do not depend on the template. From the results it is possible to conclude that the performance of mitigation is highly influenced by the template chosen by the IXP operator and the recommendation system imposes little overhead to the mitigation process. Besides, the whole mitigation took less than 5 seconds even in a very dense scenario with a large number of elephant flows, demonstrating the scalability of the recommendation system.

As expected, there is a tradeoff between the impact of the mitigation on the IXP network using the BAP template and the internal physical topology of the IXP. If the IXP network does not offer alternative paths, the effect of the BAP template will be reduced, which can be observed from the results. However, the operator may designed different templates to achieve other objectives than load balancing. For example, a template aimed at reducing energy consumption may perform better in the same scenario. Therefore, the internal topology of the IXP can also influence the performance of the template.

V. Conclusion

In this paper we have proposed and presented an SDN-based IXP management solution to handle elephant flows in the IXP network. Our solution consists of a recommendation system that uses snapshots of elephant flow paths to assist the IXP operator in the mitigation and/or control of these flows. Unlike other approaches, our system delivers a recommendation that can be validated by the operator before its actual deployment in the network.

Evaluation results show that the proposed recommendation system allows the IXP operator to handle elephant flows using user-defined templates that suggest alternative configurations for the IXP network. Furthermore, by using the BAP algorithm as a template, the system achieved relatively small mitigation times even for a large number of elephant flows, with the installation of rules on switches as bottleneck. Even a simple algorithm as the BAP can bring significant results to flow management, reducing traffic load in core switches and enabling better utilization of the IXP network.

The definition of a recommendation system to mitigate and control elephant flows in a SDN-based IXP is only the first step towards an effective flow management. The study and definition of different templates, such as QoS-driven templates or new routing strategies in critical networks such as IXP networks is necessary to enable the IXP operator with means to choose well-suited elephant flow forwarding strategies and evaluate their relationship with the mice flows. We also intend to define and incorporate other templates to the recommendation system and take advantage of new features present in the latest OpenFlow versions. Also, sampling alternatives based on the native OpenFlow RESTful interface need to be further investigated.

References

[1] Google. (2015) Peering Policy. https://peering.google.com/about/ peering_policy.html.

[2] Facebook. (2015) Peering Policy. https://www.facebook.com/peering/.

[3] Netflix. (2015) Peering Policy. https://www.netflix.com/openconnect/deliveryOptions.

[4] "on the importance of internet exchange points for today's internet ecosystem."

[5] J. Stringer, D. Pemberton, Q. Fu, C. Lorier, R. Nelson, J. Bailey, C. Correa, and C. Esteve Rothenberg, "Cardigan: SDN distributed routing fabric going live at an Internet exchange," in *Computers and Communication (ISCC), 2014 IEEE Symposium on*, June 2014, pp. 1–7.

[6] A. Gupta, M. Shahbaz, L. Vanbever, H. Kim, R. Clark, N. Feamster, J. Rexford, and S. Shenker, "SDX: A Software Defined Internet Exchange," *ACM Conference on Applications, Technologies, Architectures, and Protocols for Computer Communication (SIGCOMM)*, pp. 163–174, 17–22 August 2014.

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

[8] B. Raghavan, M. Casado, T. Koponen, S. Ratnasamy, A. Ghodsi, and S. Shenker, "Software-defined Internet Architecture: Decoupling Architecture from Infrastructure," in *Proceedings of the 11th ACM Workshop on Hot Topics in Networks*, ser. HotNets-XI. New York, NY, USA: ACM, 2012, pp. 43–48.

[9] L. Vanbever, "Novel Applications for a SDN-enabled Internet Exchange Point," Berlin, Germany, SDN Research Group meeting, IETF 87, 29 July 2013.

[10] K. Giotis, C. Argyropoulos, G. Androulidakis, D. Kalogeras, and V. Maglaris, "Combining OpenFlow and sFlow for an Effective and Scalable Anomaly Detection and Mitigation Mechanism on SDN Environments," *Comput. Netw.*, vol. 62, pp. 122–136, Apr. 2014.

[11] B. Augustin, B. Krishnamurthy, and W. Willinger, "IXPs: Mapped?" in *ACM SIGCOMM Conference on Internet Measurement Conference (IMC)*, Chicago, USA, 4–6 November 2009, pp. 336–349.

[12] L. Guo and I. Matta, "The War Between Mice and Elephants," in *International Conference on Network Protocols (ICNP)*, Riverside, USA, 11–14 November 2001, pp. 180–188.

[13] R. Zhou, "Datacenter Network Large Flow Detection and Scheduling from the Edge," in *Reading & Research Project*, 2014.

[14] C.-Y. Lin, C. Chen, J.-W. Chang, and Y. H. Chu, "Elephant Flow Detection in Datacenters using OpenFlow-based Hierarchical Statistics Pulling," in *Global Communications Conference (GLOBECOM), 2014 IEEE*, Dec 2014, pp. 2264–2269.

[15] R. Narayanan, S. Kotha, G. Lin, A. Khan, S. Rizvi, W. Javed, H. Khan, and S. Khayam, "Macroflows and microflows: Enabling rapid network innovation through a split sdn data plane," in *Software Defined Networking (EWSDN), 2012 European Workshop on*, Oct 2012, pp. 79–84.

[16] L. Knob, R. Esteves, L. Z. Granville, and L. M. R. Tarouco, "SDEFIX - Identifying Elephant Flows in SDN-Based IXP Networks," in *NOMS 2016*, Istambul, Turkey, apr 2015.

[17] Euro-IX. (2015) IXP Best Common Operational Practices. European Internet Exchange Association. https://www.euro-ix.net/documents/1391-euro-ix-ixp-bcops-221014-pdf.

[18] ISOC. (2009) Promoting the Use of Internet Exchange Points: A Guide to Policy, Management, and Technical Issues. Internet Society. http://www.internetsociety.org/promoting-use-internet-exchange-points-guide-policy-management-and-technical-issues.

[19] M. Casado and J. Pettit. (2013) Of Mice and Elephants. http://networkheresy.com/2013/11/01/of-mice-and-elephants/.

[20] I. F. Akyildiz, A. Lee, P. Wang, M. Luo, and W. Chou, "A Roadmap for Traffic Engineering in SDN-OpenFlow Networks," *Computer Networks*, vol. 71, pp. 1–30, Oct. 2014.

[21] C. Hopps, "RFC 2992: Analysis of an Equal-Cost Multi-Path Algorithm," IETF, 2000.

[22] D. Awduche, J. Malcolm, J. Agogbua, M. O'Dell, and J. McManus, "RFC 2702: Requirements for Traffic Engineering Over MPLS," IETF, 1999.

[23] W. Xia, Y. Wen, C. H. Foh, D. Niyato, and H. Xie, "A Survey on Software-Defined Networking," *IEEE Communications Surveys and Tutorials*, vol. 17, no. 1, pp. 27–51, Firstquarter 2015.

[24] ONF. (2015) Open Networking Foundation. https://www.opennetworking.org.

[25] J. Wickboldt, W. De Jesus, P. Isolani, C. Both, J. Rochol, and L. Granville, "Software-Defined Networking: Management Requirements and Challenges," *IEEE Communications Magazine*, vol. 53, no. 1, pp. 278–285, January 2015.

[26] R. P. Esteves, L. Z. Granville, and R. Boutaba, "On the Management of Virtual Networks," *IEEE Communications Magazine*, vol. 51, no. 7, pp. 2–10, July 2013.

[27] A. Curtis, W. Kim, and P. Yalagandula, "Mahout: Low-overhead Datacenter Traffic Management using End-host-based Elephant Detection," in *Proceedings of IEEE INFOCOM 2011*, April 2011, pp. 1629–1637.

[28] A. R. Curtis, J. C. Mogul, J. Tourrilhes, P. Yalagandula, P. Sharma, and S. Banerjee, "DevoFlow: Scaling Flow Management for High-performance Networks," *SIGCOMM Comput. Commun. Rev.*, vol. 41, no. 4, pp. 254–265, Aug. 2011.

[29] N. McKeown, T. Anderson, H. Balakrishnan, G. Parulkar, L. Peterson, J. Rexford, S. Shenker, and J. Turner, "OpenFlow: Enabling Innovation in Campus Networks," *SIGCOMM Computer Communication Review*, vol. 38, no. 2, pp. 69–74, March 2008.

[30] J. Suh, T. T. Kwon, C. Dixon, W. Felter, and J. Carter, "OpenSample: A Low-Latency, Sampling-Based Measurement Platform for Commodity SDN," in *Proceedings of the 2014 IEEE 34th International Conference on Distributed Computing Systems*, ser. ICDCS '14. Washington, DC, USA: IEEE Computer Society, 2014, pp. 228–237.

[31] Ryu. (2015) Ryu openflow controller. http://osrg.github.com/ryu.

[32] OpenDayLight. (2015) The OpenDayLight Platform. https://www.opendaylight.org.

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

[34] Mininet. (2015) Mininet: An Instant Virtual Network on your Laptop (or other PC). http://mininet.org/.

[35] Openvswitch. (2015) Open vSwitch: Production Quality, Multilayer Open Virtual Switch. http://openvswitch.org/.