UNIVERSIDADE FEDERAL DO RIO GRANDE DO SUL
INSTITUTO DE INFORMÁTICA
PROGRAMA DE PÓS-GRADUAÇÃO EM COMPUTAÇÃO

LUIS AUGUSTO DIAS KNOB

SDEFIX: Gerenciando fluxos elefantes em pontos de troca de tráfego baseados em redes definidas por software

Dissertação apresentada como requisito parcial para a obtenção do grau de Mestre em Ciência da Computação

Orientadora: Profa. Dra. Liane Margarida Rockenbach Tarouco

Porto Alegre
2016

CIP – CATALOGAÇÃO NA PUBLICAÇÃO

Knob, Luis Augusto Dias

SDEFIX: Gerenciando fluxos elefantes em pontos de troca de tráfego baseados em redes definidas por software / Luis Augusto Dias Knob. – Porto Alegre: PPGC da UFRGS, 2016.

78 f.: il.

Dissertação (mestrado) – Universidade Federal do Rio Grande do Sul. Programa de Pós-Graduação em Computação, Porto Alegre, BR–RS, 2016. Orientadora: Liane Margarida Rockenbach Tarouco.

1. Pontos de Troca de Tráfego. 2. Redes Definidas por Software. 3. Gerência de Redes. I. Tarouco, Liane Margarida Rockenbach. II. Título.

AGRADECIMENTOS

Inicialmente, gostaria de agradecer a toda minha família pelo total apoio e incentivo durante minha caminhada acadêmica. Em especial, a minha mãe, Dinah, pela educação e valores ensinados, tendo sido sempre um exemplo para mim. Para você mãe, meu eterno agradecimento.

Um carinhoso agradecimento à minha esposa Natália, pelo apoio, pela compreensão e pela infinita paciência durante este período. Obrigado por ser minha amada companheira tanto nos momentos alegres, quanto naqueles não tão felizes. Saiba que te amo incondicionalmente.

Agradeço a professora Liane Tarouco pela oportunidade de realizar um mestrado em uma das melhores instituições de ensino do país. Ao professor Lisandro, agradeço por todas as lições, ensinamentos e duras que sem sombra de dúvidas foram essenciais na minha formação acadêmica. Também agradeço pela confiança em mim depositada, durante os diversos trabalhos que realizamos juntos neste período em que de certa forma fui seu orientando. Aos professores Luciano Paschoal Gaspary, Marinho P. Barcellos e Alberto E. Schaeffer-Filho agradeço pela qualidade das disciplinas ministradas e por sempre esperarem o melhor de mim. Tenho muito orgulho de ter sido aluno de um grupo tão distinto de professores.

Sou muito grato a todos os amigos que criei no Grupo de Redes, principalmente aqueles que tiveram que me aguentar por dois longos anos no laboratório 212. Obrigado Gabriel, Alexandre, Rafael Martins, Pedro, Cristian, Juliano, Lucas, Giovani, Vinicius e aos meus dois "irmãos" de mestrado, Eduardo e Ânderson, pelo companheirismo. Um agradecimento especial para Rafael Esteves pelas revisões, críticas e sugestões nos artigos que realizamos juntos. Agradeço também aos demais colegas de UFRGS que, embora não estejam aqui nomeados, foram fundamentais para esse momento.

Finalmente, gostaria de agradecer a todos que de alguma forma incentivaram ou participaram, direta ou indiretamente de minha formação acadêmica.

"Isn’t it enough to see that a garden is beautiful without having to believe that there are fairies at the bottom of it too?".

— Douglas Adams.

RESUMO

Os Pontos de Troca de Tráfego participam de maneira substancial e crítica no ecossistema da Internet, possibilitando conexões entre múltiplos Sistemas Autônomos (ASes, do inglês *Autonomous Systems*). O gerenciamento das redes de PTT possui como objetivos primários, o gerenciamento dos chamados fluxos elefante (do inglês, *elephant flows*). Fluxos elefante tendem a existir em número reduzido, porém correspondem à maioria do tráfego em uma infraestrutura de rede. O gerenciamento dos fluxos elefante envolve uma adequada identificação e quando necessário, um redirecionamento destes fluxos para caminhos mais apropriados, de forma a minimizar os possíveis impactos sobre os outros fluxos ativos na rede. Além disso, o gerenciamento de fluxos elefante tornou-se um importante objeto de discussão em PTTs baseados em redes SDN, principalmente porque estas redes dispõem de controladores que possuem uma visão consistente da rede subjacente, o que permite uma gerência destes fluxos de forma refinada. Nesta dissertação, será proposto, desenvolvido e avaliado um sistema de identificação dos fluxos elefante e seus respectivos caminhos de rede, em conjunto com um sistema de recomendação, que possui o objetivo de sugerir configurações alternativas para os fluxos elefante identificados anteriormente nas redes de PTTs baseadas em SDN. Neste sistema, o operador do PTT pode definir *templates* que em última instância definem como os caminhos dos fluxos elefante serão modificados para atender objetivos específicos. Por fim, será demonstrado que o sistema proposto pode auxiliar o operador do PTT a identificar, gerenciar e mitigar o impacto dos fluxos elefante da rede do PTT.

Palavras-chave: Pontos de Troca de Tráfego. Redes Definidas por Software. Gerência de Redes.

SDEFIX: Manage Elephant Flows in SDN-Based IXP Networks

ABSTRACT

Internet Exchange Points (IXPs) play a key role in the current Internet architecture enabling cost-effective connections among multiple autonomous systems (ASes). Management of IXP networks is primarily concerned with the management of the so-called elephant flows. Such flows represent a small portion of the total flows of a IXP network but usually have high impact on the overall 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 active in the network. Elephant flow management becomes 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 master thesis, we propose, develop, and evaluate an identification system to identify elephant flows and their respectively paths, as well as a recommendation system to suggest alternative configurations to previously identified elephant flows in an SDN-based IXP network. In this solution, the IXP operator can define templates that ultimately define how elephant flows can be reconfigured to achieve a specific objective. We demonstrate that our system can help IXP operators to identify, handle and mitigate the impact of elephant flows in the IXP network.

Keywords: Software Defined Networking. Internet Exchange Points. Network Management.

LISTA DE FIGURAS

2.1 Plataforma de camada-1 MPLS/VPLS do PTT AMS-IX 18
2.2 Arquitetura SDN 21

3.1 Arquitetura do sistema de identificação para PTTs baseados em rede SDN - SDEFIX 27
3.2 Campos de entrada do *framework* SDEFIX 28
3.3 Visualização da Topologia na *Admin interface* 31
3.4 Arquitetura do sistema de recomendação para PTTs baseados em redes SDN 32
3.5 Descrição da API para a construção de *templates* 34
3.6 *Snapshot* utilizado como entrada 35
3.7 Novo caminho do fluxo elefante após a validação do *template* 35

4.1 Coleções criadas no mongoDB para o SDEFIX 40
4.2 Conversão da ferramenra sflowtool para as coleções *sflowpacket* e *samples* 41
4.3 Diagrama de Sequência: o sistema de identificação 42
4.4 Diagrama de Sequência: o sistema de recomendação 43
4.5 Fluxograma do algoritmo BAP 44

5.1 Uma infraestrutura de PTT baseada no AMS-IX 46
5.2 Topologia utilizada nos experimentos 47
5.3 Tempo de identificação pelo número de regras de identificação 48
5.4 Número de fluxos elefante identificado em relação a taxa de amostragem de pacotes 50
5.5 Tempo de Mitigação da solução 53
5.6 Pico de tráfego nos *switches* 54

LISTA DE TABELAS

5.1 Tráfego de amostragem (Mbytes) 50
5.2 Utilização de recursos do servidor 50

LISTA DE ABREVIATURAS E SIGLAS

API	*Application Programming Interface*
AS	*Autonomous System*
BAP	*Best Alternative Path*
BGP	*Border Gateway Protocol*
BSON	*Binary JavaScript Object Notation*
CDN	*Content Delivery Network*
CIX	*Commercial Internet eXchange*
CPU	*Central Processing Unit*
CSS	*Cascading Style Sheets*
DoS	*Denial of Service*
ECMP	*Equal-cost multi-path routing*
HTML	*HyperText Markup Language*
ICMP	*Internet Control Message Protocol*
IETF	*Internet Engineering Task Force*
IP	*Internet Protocol*
IRTF	*Internet Research Task Force*
ISP	*Internet Service Provider*
JSON	*JavaScript Object Notation*
LRU	*Least Recent Used*
MAC	*Media Access Control*
MAE	*Metropolitan Area Exchanges*
MPLS	*Multi Protocol Label Switching*
NAP	*Network Access Points*
NFV	*Network Function Virtualisation*
NSFNET	*National Science Foundation Network*
ONF	*Open Network Foundation*

PTT	Pontos de Troca de Tráfego
QoS	*Quality of Service*
REST	*Representational State Transfer*
SDN	*Software-Defined Networking*
SQL	*Structured Query Language*
STP	*Spanning Tree Protocol*
SVG	*Scalable Vector Graphics*
TCP	*Transport Control Protocol*
TI	Tecnologia da Informação
UDP	*User Datagram Protocol*
VPLS	*Virtual Private LAN Service*

SUMÁRIO

1 INTRODUÇÃO 13

2 FUNDAMENTAÇÃO TEÓRICA 16
2.1 Pontos de Troca de Tráfego 16
2.1.1 Infraestrutura de um PTT 17
2.2 Fluxos Elefante 18
2.2.1 Gerência de Fluxos em PTTs 19
2.3 Redes Definidas por Software (SDN) 20
2.3.1 SDN em PTTs 21
2.3.2 Fluxos Elefante e SDN 22
2.3.3 Fluxos Elefantes em Pontos de Troca Definidos por Software 25

3 GERENCIANDO FLUXOS ELEFANTES EM PTTS BASEADOS EM SDN . 26
3.1 Sistema de Monitoramento para PTTs baseados em SDN 26
3.1.1 Arquitetura do Sistema 26
3.1.2 Especificação dos Fluxos Elefantes 27
3.1.3 Identificação dos Fluxos Elefantes 28
3.1.4 Apresentando os caminhos dos Fluxos Elefante 30
3.2 Sistema de Recomendação para PTTs baseado em SDN 30
3.2.1 Arquitetura do Sistema 31
3.2.2 Especificação dos Templates 34
3.2.3 Validação pelo operador do PTT 34
3.2.4 Exemplos de uma recomendação 35

4 IMPLEMENTAÇÃO DO PROTÓTIPO 37
4.1 Implementação do Ambiente de Desenvolvimento 37
4.2 Modelo de Dados 39
4.3 Obtendo informações da Rede 39
4.4 Identificação de Fluxos Elefantes 41
4.5 Implementação do Sistema de Recomendação 43
4.5.1 Implementação dos Templates 43

5 AVALIAÇÃO EXPERIMENTAL 46
5.1 Avaliação da Identificação dos Fluxos Elefantes 46
5.1.1 Cenário 46
5.1.2 Resultados 48

5.1.3 Resumo dos Experimentos 51
5.2 Avaliação do Sistema de Recomendação 51
5.2.1 Cenário 51
5.2.2 Resultados 52
5.2.3 Resumo dos Experimentos 53

6 CONCLUSÃO 55
6.1 Principais Contribuições e Resultados Obtidos 55
6.2 Considerações Finais e Trabalhos Futuros 56

REFERÊNCIAS 57

APÊNDICEA ARTIGO PUBLICADO – NOMS 2016 63

APÊNDICEB ARTIGO SUBMETIDO – GLOBECOM 2016 72

1 INTRODUÇÃO

Os Pontos de Troca de Tráfego conectam Sistemas Autônomos (ASes, do inglês *Autonomous Systems*) e permitem que provedores de serviços (SPs, do inglês *service providers*) troquem tráfego diretamente para melhor servirem aos seus usuários. Responsável por pelo menos 20% de todo o tráfego trocado entre Sistemas Autônomos (ASes) (RESTREPO; STANOJEVIC, 2012), os PTTs participam de maneira substancial e crítica no ecossistema da Internet, tanto no que diz respeito à comunicação entre as diferentes partes (ex. provedores de conteúdo, CDNs e ISPs), quanto na relação entre os provedores de conteúdo e os usuários finais (CHATZIS; SMARAGDAKIS; FELDMANN, 2013). Além disso, o tráfego diário de um grande PTT, como o DE-CIX, pode ser comparado com o tráfego de um provedor de serviço da Internet (ISP, do inglês *Internet Service Provider*) *tier*-1 como a AT&T (AUGUSTIN; KRISHNAMURTHY; WILLINGER, 2009). Um dos principais benefícios experimentados pelos participantes de um PTT é o baixo custo de implementação e manutenção da estrutura de conexão (GREGORI et al., 2011). Como consequência, grandes companhias da Internet como Google (GOOGLE, 2015), Facebook (FACEBOOK, 2015) e Netflix (NETFLIX, 2015) procuram conectar suas infraestruturas aos maiores PTTs, de forma a reduzir os custos de conexão e o tempo de acesso de seus consumidores a seus serviços.

Recentemente, com o surgimento da arquitetura de Redes Definidas por Software (SDN, do inglês *Software-Defined Networking*) (WICKBOLDT et al., 2015), diversas soluções para gerenciamento de PTTs baseados em SDN foram propostas. Isto ocorreu principalmente para superar as dificuldades inerentes do Border Gateway Protocol (BGP) (STRINGER et al., 2014), e para o instanciamento de funcionalidades que seriam difíceis ou até impossíveis de serem implementadas sem essa tecnologia. Como exemplo, podemos citar aplicações específicas para *peering* (GUPTA et al., 2014), engenharia de tráfego de entrada (AFAQ; REHMAN; SONG, 2015a), balanceamento de carga de servidores alocados em longas distâncias (RAGHAVAN et al., 2012) e bloqueio de ataques de negação de serviço (DoS, do inglês *Denial os Service*) em tempo de execução (VANBEVER, 2013) (GIOTIS et al., 2014).

Neste contexto, um controlador SDN é imprescindível para gerenciar o conjunto de *switches* que compõem a rede de um PTT. Entretanto, um processo complementar que realize *snapshots* que registrem o estado atual da rede é requerido, de forma a aprimorar o funcionamento da rede SDN. Com estes *snapshots*, o controlador SDN do PTT pode aprimorar a rede subjacente para cumprir suas metas. Nesta dissertação, interessa-se, especificadamente, pela identificação os chamados fluxos elefante que transitam em uma rede de PTT. Um fluxo é considerado elefante quando possui uma longa duração e gera uma quantidade significante de tráfego (GUO; MATTA, 2001). Fluxos elefante também tendem a existir em número reduzido, porém eles correspondem à maioria do tráfego em uma infraestrutura de rede. Em conjunto à identificação dos fluxos elefante, também cabe descobrir os caminhos que estes fluxos realizam nas redes de PTTs. Conhecer esse caminhos permite ao operador do PTT, por exemplo, alterar a carga

da rede movendo os fluxos elefante dos seus caminhos originais, que estão congestionados, para um mais apropriado. Embora diversos trabalhos apresentem soluções para a identificação de fluxos elefante (ZHOU, 2014) (LIN et al., 2014) (NARAYANAN et al., 2012), apenas uma pequena quantidade apresenta métodos para lidar com os fluxos ao longo da rede (AFAQ; REHMAN; SONG, 2015a). De acordo com o nosso conhecimento, não existem hoje soluções para o gerenciamento de fluxos elefante em redes de PTTs que explorem os benefícios das redes SDN e entreguem um controle tão granular da solução ao operador do PTT.

Têm-se como objetivo desta dissertação implementar uma solução que: (i) permita que operador do PTT, através de um sistema de monitoramento, identifique os fluxos elefante existentes na rede e os caminhos a eles associados respeitando regras predefinidas pelo operador e; (ii) através de um sistema de recomendação, auxilie o operador do PTT a mitigar o efeito destes fluxos elefante na rede.

Desta forma, criou-se uma solução que disponibiliza ao operador do PTT um *snapshot* dos fluxos elefante identificados e os caminhos a eles associados. Criou-se também um sistema de gerenciamento que permite ao operador do PTT definir como serão distribuídos os fluxos elefante dentro da rede do PTT, permitindo o tratamento destes fluxos utilizando regras SDN. De fato, a solução em questão inclui um sistema de recomendação que, por implementar *templates* (*ex.* Otimização do Consumo de Enlaces, Qualidade de Serviço (QoS) e Descarte de Pacotes), auxilia o operador do PTT na mitigação dos efeitos dos fluxos elefante na rede através de sugestões de modificações na rede do PTT que podem ser personalizadas pelo operador de forma a entregar um refinado ciclo de controle.

De forma a validar este sistema, desenvolveu-se o protótipo SDEFIX (*Software-Defined Elephant Flow Identifier for Internet Exchange*), o qual foi dividido em duas partes. A primeira, responsável pela identificação dos fluxos elefante e seus respectivos caminhos de rede e a segunda, descrevendo um sistema de gerenciamento dos fluxos baseado em recomendações que utiliza os *snapshots* gerados pela primeira parte. Além disso, implementou-se uma rede com o emulador Mininet utilizando uma topologia inspirada na rede do PTT AMS-IX, e realizou-se um série de testes relacionados à identificação dos fluxos elefante no que se refere ao tempo de identificação, ao consumo de recursos, à taxa de transferência e à melhor abordagem das configurações, e ao sistema de recomendação, onde testamos o tempo de execução do sistema e a otimização da rede na utilização de um *template* demonstrativo que calcula a segunda melhor rota para cada fluxo elefante.

O restante desta dissertação é organizado da seguinte forma: no Capítulo 2 são apresentados conceitos sobre redes definidas por software, gerenciamento de PTTs e o controle de fluxos, em destaque dos fluxos elefante. São apresentadas também soluções do estado-da-arte entre os três conceitos. No Capítulo 3 são apresentados os detalhes da solução proposta, como modelo de informação e as arquiteturas dos dois principais sistemas do *framework*, a identificação dos fluxos e o sistema de recomendação. No Capítulo 4 são apresentados os detalhes de implementação dos componentes do *framework* SDEFIX, bem como um apanhado das tecnologias envolvidas

no funcionamento da solução. No Capítulo 5 são apresentados os resultados das avaliações em relação ao tempo de identificação, total de fluxos identificados, consumo de processamento, memória e os resultados da avaliação de escalabilidade na identificação dos fluxos, o tempo de execução e pico de tráfego utilizando um *template* para demostrar o funcionamento do sistema de recomendação. No Capítulo 6 são apresentadas as conclusões finais e trabalhos futuros no contexto desta dissertação.

2 FUNDAMENTAÇÃO TEÓRICA

Neste capítulo, serão revisados os conceitos que permeiam esta dissertação. Inicialmente será apresentado um histórico dos pontos de troca de tráfego, bem como, uma descrição de seu funcionamento. Após, serão discutidas soluções de classificação de fluxo, focando nas diferenças entre fluxos elefante e fluxos diminutos (camundongos), sendo apresentados, ao final desta seção, os trabalhos que realizam a classificação de fluxos elefante em redes de PTTs ou similares. Posteriormente, revisar-se-á os conceitos de redes definidas por software, e sua interação com a classificação de fluxos elefante e as redes de PTTs.

2.1 Pontos de Troca de Tráfego

Pontos de Troca de Tráfego (PTTs) são infraestruturas de redes que permitem que distintos Sistemas Autônomos (ASes, do inglês *Autonomous Systems*) troquem tráfego de forma mais eficiente e com um melhor custo-benefício. Ao utilizarem PTTs, Provedores de Serviço de Internet (ISPs, do inglês *Internet Service Providers*) e provedores de redes de distribuição de conteúdo (CDNs, do inglês *Content Delivery Networks*) podem trocar tráfego sem depender de terceiros, e com isso podem diminuir o custo total de comunicação com outros ASes. Os PTTs evoluíram dos antigos *Network Access Points* (NAPs), que foram criados para orquestrar a transição de redes acadêmicas patrocinadas pelo governo americano (ex. NSFNET) para as redes comerciais da Internet atual. Outras iniciativas similares incluem o *Commercial Internet eXchange* (CIX) e a *Metropolitan Area Exchanges* (MAEs). Todos estes esforços resultaram nos PTTs modernos que estão implantados mundialmente (CHATZIS et al., 2013) (AGER et al., 2012).

As operações comerciais dos PTTs podem ser divididas entre dois grupos distintos: Privados e Comunitários. PTTs Privados, ou “com fins lucrativos”, são gerenciados por companhias que cobram uma taxa dos participantes, sendo ela mensal ou baseada no volume total de tráfego trocado. Por outro lado, os PTTs comunitários, ou “sem fins lucrativos”, são administrados pelos próprios membros ou por organizações sem fins lucrativos. Tipicamente, o modelo adotado pelos PTTs alocados na Europa, Africa e América Latina é o comunitário, enquanto que nos Estados Unidos e Canadá existe dominância dos PTTs comerciais. Isto ocorre por dois motivos: razões comerciais (desde suas criações, os maiores ASes americanos possuem conexões diretas entre si) e; razões legislativas, já que em muitos países é determinado que o tráfego entre provedores de serviço ocorra dentro de seu próprio território (CHATZIS et al., 2013).

Hoje, existem mais de 300 PTTs ativos conectando milhares de ASes através do mundo (CHATZIS et al., 2013). PTTs são tipicamente agrupados de acordo com a sua localização geográfica ou em federações. Exemplos de federações incluem as Euro-IX (Europa) (EURO-IX, 2015a), LAC-IX (América Latina) (LAC-IX, 2015), APIX (Ásia-Pacífico) (APIX, 2015), e Af-IX (África) (AF-IX, 2015). O principal objetivo por trás destas federações é promover e

regular a troca de tráfego entre seu membros. Além disso, todas as federações citadas anteriormente são associadas globalmente ao IX-F (Internet eXchange Federation). Refletindo sua natureza comercial, maioria dos PTTs alocados nos Estados Unidos e Canadá não pertencem a nenhuma federação. Recentemente, as maiores empresas da Internet, como o Google, Netflix e Akamai, começaram a incentivar a associação Open-IX (OPEN-IX, 2015). A Open-IX define um conjunto de padrões técnicos e operacionais para PTTs e surge como uma organização sem fins lucrativos na América do Norte, buscando a implementação de novos PTTs que adotem o modelo usualmente utilizado na Europa (CHATZIS et al., 2015).

No Brasil, o Comitê Gestor da Internet (CGI.br) é o responsável pelo projeto PTTMetro, que define as políticas de criação de PTTs no território nacional (SILVA et al., 2004). Entre os diversos pontos tratados no documento, um dos principais é o caráter comunitário dos PTTs brasileiros. O projeto, operacionalizado desde 2004, e distribuído entre 26 localidades, possuía em abril de 2016, um tráfego médio de 1.33 Tb/s, majoritariamente vindo do PTTMetro São Paulo, que centraliza aproximadamente 80 % de todo o tráfego da rede PTTMetro (PTTMETRO, 2015).

2.1.1 Infraestrutura de um PTT

Por definição, os PTTs são infraestruturas compartilhadas que recebem a conexão de diversos ASes com objetivo principal de otimizar a troca de tráfego entre operadores, além de diminuir o tempo de acesso entre redes localizadas em uma mesma região, através da minimização do número de saltos entre eles. Por conectarem normalmente uma grande quantidade de ASes, os PTTs possuem um núcleo bastante complexo, principalmente pela quantidade de conexões entre seus clientes, necessitando de equipamentos com um alto desempenho que possam processar altas quantidades de pacotes por segundo (CHATZIS et al., 2013).

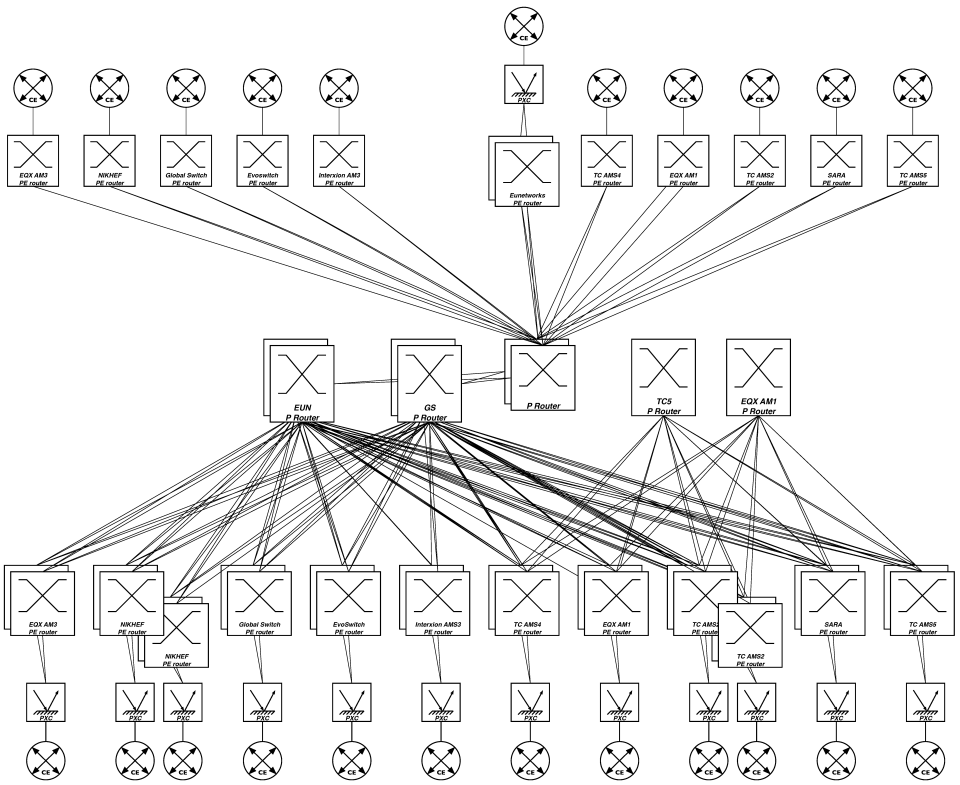
Embora estas infraestruturas pareçam complexas, um PTT pode ser resumido como um ponto centralizador de conexões entre diversas redes, sendo normalmente implementado através de soluções *switching-fabric* baseadas em tecnologia *Ethernet* (camada 2). Uma vez conectados fisicamente ao PTT, os membros devem promover alguma forma de *peering* com os outros ASes, sendo usualmente realizado: ou via um contrato multilateral, com todos os membros do PTT; ou via bilateral, onde são realizados acordos entre dois ou mais membros. A realização destes *peerings* e o acesso aos prefixos IPs de cada AS são feitos por meio do protocolo Border Gateway Protocol (BGP) (BRITO et al., 2015).

Alguns estudos revelam a importância das infraestruturas dos PTTs para a Internet. No trabalho apresentado por Ahmad em 2012, este visualizou que pelo menos 10% de todas as rotas de melhores caminhos disponíveis passavam por PTTs. Além disso a taxa de rotas com baixa latência entre múltiplos provedores de Internet estava disponível entre 30-60% do tempo nos PTTs. (AHMAD; GUHA, 2012)

Durante esta dissertação, utilizar-se-á como modelo de rede de PTTs, a infraestrutura pre-

sente no AMS-IX. O AMS-IX é um PTT alocado na cidade de Amsterdam na Holanda com aproximadamente 800 ASes e picos de transferência de 4.6 Tb/s, o que o torna no maior PTT em atividade (AMS-IX.NET, 2016). Atualmente, a infraestrutura de *peering* utilizada pelo AMS-IX é a MPLS/VPLS. Esta, escolhida principalmente pela grande quantidade de ASes conectadas, pela resiliência e pela alta escabilidade inerente a este tipo de rede. Os equipamentos de rede suportam uma ou mais conexões com interface *Ethernet* de até 100 Gigabit, através dos *Providers Edge Routers*, que são conectados a dois *switches* centrais redundantes. A Figura 2.1 apresenta a infraestrutura presente na AMS-IX.

Figura 2.1: Plataforma de camada-1 MPLS/VPLS do PTT AMS-IX



Fonte: (AMS-IX.NET, 2016).

2.2 Fluxos Elefante

Não obstante os diferentes tipos de comunicações e protocolos existentes na Internet, uma característica singular no tráfego destes pacotes, é que a maioria dos fluxos são pequenos em tamanho e/ou tempo de vida (usualmente chamados de fluxos camundongos, do inglês *mice flows*). Contudo existe um pequeno número de fluxos, que ao contrário dos anteriores, carrega a maior quantidade do tráfego de pacotes na rede (chamados de fluxos elefante, do inglês *elephant flows*). Estes dados foram validados em alguns trabalhos no início da década de 2000 (FANG; PETERSON, 1999) (ZHANG et al., 2002) (MORI et al., 2004) (GUO; MATTA, 2001) e é

esperado que estes valores tenham se mantidos estáveis, ainda que tenham ocorridas alterações na forma em que os fluxos são distribuídos pela internet (ZHANG; WANG; LAN, 2015).

Embora fosse de se esperar que em uma rede justa os fluxos pequenos apresentassem serviços relativamente rápidos em relação a conexões mais demoradas, isso não acontece como previsto. Tal fato ocorre principalmente pela forma como são geradas as conexões TCP e pela janela de transferência deste protocolo. Somando-se a isso, a atual infraestrutura de rede nas camadas inferiores dificulta o encaminhamento por caminhos diferenciados devido ao seu tamanho, mas sem grandes desafios de implementação.

Em conjunto à definição qualitativa dos fluxos elefante (determinada pelo seu tamanho e tempo de vida), pode utilizar-se também uma definição quantitativa, aonde o fluxo elefante é determinado arbitrariamente pelo operador de rede de acordo com seus próprios critérios ou limiares que são estritamente definidos. Nesta dissertação, definiu-se um fluxo elefante como um fluxo que contribui com pelo menos 1% de todos os pacotes não amostrados da rede.

2.2.1 Gerência de Fluxos em PTTs

O gerenciamento de um PTT procura entregar a operação adequada à infraestrutura do PTT. Por exemplo, ao utilizar uma ferramenta para monitoramento, o operador do PTT pode obter uma melhor visualização da infraestrutura de rede do PTT e a partir disto, identificar gargalos, lidar com falhas e realizar uma engenharia de tráfego apropriada de forma a obter uma melhor utilização dos recursos de rede disponíveis (EURO-IX, 2015b) (ISOC, 2009). Operar com confidencialidade e promover o uso justo dos recursos entre as ASEs são outros exemplos de problemas que precisam ser gerenciados em um PTT.

Um aspecto fundamental no gerenciamento de PTTs é o controle dos fluxos elefante. Em geral, um fluxo é considerado elefante quando ele possui uma alta taxa de transferência por um longo período de tempo (CASADO; PETTIT, 2013) (GUO; MATTA, 2001) em comparação com outros fluxos da mesma rede. Fluxos elefante somam para uma pequena porção do total de fluxos de uma rede, porém podem carregar a maior parte dos dados trafegados (GUO; MATTA, 2001). Gerenciar os fluxos elefante é importante pelo potencial de impacto que estes fluxos possuem em um PTT. Fluxos elefante são fluxos com longo período de vida que rapidamente consumem as memórias dos equipamentos de rede e impõem indesejáveis atrasos no enfilamento dos fluxos com curto período de vida. Além disso, mitigar a influência negativa dos fluxos elefante sobre os outros fluxos e otimizar a utilização dos recursos de rede de forma mais homogênea são ações de gerência que devem partir do operador do PTT.

A primeira etapa da gerência dos fluxos elefante em um PTT é a identificação destes. Uma vez que o operador do PTT possui o conhecimento de cada um deles, ele pode realizar ações apropriadas que otimizem a utilização dos recursos de rede e que diminuam o potencial negativo do impacto dos fluxos elefante sobre a rede do PTT. Por exemplo, os fluxos elefante podem ser isolados ou migrados para outros caminhos de redes disponíveis, disponibilizando o caminho

padrão para os fluxos pequenos. Utilizar estratégias de rotas diferentes dos fluxos tradicionais ou diferenciá-los por diferentes esquemas de QoS (Qualidade do Serviço, do inglês *Quality of Service*) (GUO; MATTA, 2001) (CASADO; PETTIT, 2013) também são alternativas.

2.3 Redes Definidas por Software (SDN)

Redes Definidas por Software (SDN, do inglês *Software-Defined Networking*) é um tipo de arquitetura de rede dinâmica que busca auxiliar e acelerar a implementação de novos serviços de rede e responder rapidamente às mudanças de requisitos destes serviços (ONF, 2015). Desta forma, SDN apresenta uma separação do plano de controle do plano de dados, diferente dos atuais dispositivos de rede (como *switches* e roteadores). Nesta divisão, o plano de controle fica responsável pelas tomada de decisões sobre os fluxos e a atualização das tabelas de encaminhamento, enquanto o plano de dados administra a comutação dos pacotes pela rede (FEAMSTER; REXFORD; ZEGURA, 2013).

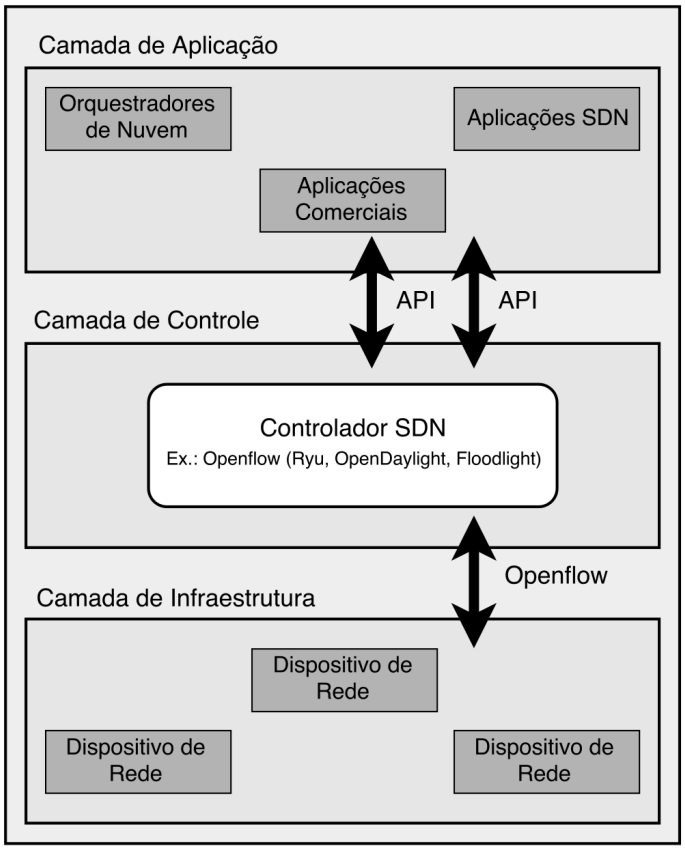
Em redes tradicionais, o plano de controle é executado individualmente em cada dispositivo de rede. Isso faz com que a tomada de decisões, como roteamento, seja realizada sem um conhecimento completo da rede. Em SDN, o plano de controle torna-se logicamente centralizado, permitindo que a parte lógica de decisão seja movida dos dispositivos para um único ponto chamado controlador. Esta centralização, além de tornar os dispositivos externos simples comutadores de pacotes, incrementa a programabilidade da rede, pois o controlador através de uma interface aberta como o protocolo OpenFlow (MCKEOWN et al., 2008), torna-se ciente de todos os elementos da rede e suas características.

A Figura **??** apresenta uma visão geral de uma arquitetura SDN baseada em OpenFlow. Resumidamente, os principais elementos que compõem essa arquitetura são: (i) a camada de controle onde fica o controlador Openflow, como Ryu, OpenDaylight ou Floodlight; (ii) a camada de infraestrutura onde estão localizados os equipamentos de redes responsáveis pelo encaminhamento dos pacotes; (iii) a camada de aplicação, onde ferramenta de terceiros trazem novas funcionalidades para a rede e (iv) a comunicação entre as camadas, estando a inferior o protocolo Openflow e na superior, uma API REST, normalmente utilizando JSON.

O OpenFlow permite o desenvolvimento de mecanismos programáveis baseados em uma tabela de fluxos para os dispositivos de encaminhamento. O protocolo OpenFlow estabelece um canal de comunicação seguro entre *switches* OpenFlow e o controlador, usando-o para controlar e instalar regras específicas para cada fluxo de acordo com programas personalizados implantados nos controladores de rede. Desta forma, o controlador executa estas aplicações decidindo quais as regras e ações devem ser instaladas no encaminhamento de pacotes em cada dispositivo de rede.

Atualmente, SDN tem atraído significativa atenção tanto da indústria como da academia. Grupos de administradores de redes, provedores de serviços e fornecedores recentemente fundaram a Open Network Foundation (ONF), uma organização voltada à área industrial, focada

Figura 2.2: Arquitetura SDN



Fonte: adaptado de (ONF, 2015).

em promover as redes SDN e a padronização do protocolo OpenFlow. Do lado acadêmico, foi criado o grupo OpenFlow Network Research Center. Existem também esforços de padronização em SDN na IETF e na IRTF, além de outros padrões definidos por outras organizações (NUNES et al., 2014).

2.3.1 SDN em PTTs

Redes Definidas por Software (XIA et al., 2015) (ONF, 2015) tem sido considerada uma alternativa viável para resolver diversas questões na área de redes de computadores. Por desacoplar o plano de controle do plano de dados, SDN possibilita ao operador um total controle sobre os dispositivos de encaminhamento. Recentemente, o gerenciamento em SDN e o uso de SDN como uma ferramenta de gerência tem ganhado importância na comunidade científica (WICKBOLDT et al., 2015) (ESTEVES; GRANVILLE; BOUTABA, 2013).

No contexto de PTTs, SDN pode ultrapassar as dificuldades inerentes do *Border Gateway Protocol* (BGP), definindo rotas baseadas no prefixo de destino, além de promover a aplica-

ção limitada de políticas (apenas vizinhos diretos) e seleção de caminho indiretos (GUPTA et al., 2014). Neste sentido, SDN torna-se uma poderosa ferramenta para melhorar o gerenciamento de um PTT, já que entrega funcionalidades que seriam difíceis ou impossíveis de serem implementadas antes dela. Estas funcionalidades são realizadas por novos recursos, incluindo aplicações específicas de *peering*, engenharia de tráfego de entrada, servidores de balanceamento de carga de longa distância e bloqueio de ataques DoS em tempo real (GUPTA et al., 2014) (VANBEVER, 2013).

Outros trabalhos apresentam soluções em SDN para a comunicação Inter-AS, que intrinsecamente podem ocorrer dentro de PTTs. O artigo (RAGHAVAN et al., 2012) apresenta soluções conjuntas de SDN, MPLS e software de encaminhamento para revolucionar como é redirecionado o tráfego entre os múltiplos agentes da Internet, como os provedores de serviço e os sistemas autônomos. Já, (BENNESBY et al., 2012) descreve uma solução para comunicação inter-AS baseado em SDN, como alternativa para as comunicações BGP, criando uma série de módulos para o NOX. A ferramenta funciona como uma articuladora entre os ASes, sendo requisitada toda vez que uma regra não é encontrada nas tabelas de fluxo.

O *framework* proposto por Kotronis *et al.* apresenta uma solução para criar um controlador inter-AS que gerencia as conexões entre estes enquanto mantêm uma compatibilidade com os ASes que utilizam BGP (KOTRONIS; GAMPERLI; DIMITROPOULOS, 2015). Diferente da ferramenta SDX apresentada por Gupta *et al.*, este trabalho apresenta como diferencial a implementação de um plano de controle de roteamento interdomínio separado do plano de controle tradicional (GUPTA et al., 2014). Uma solução semelhante é apresentada em Lin, onde é criado uma ponte Leste-Oeste (não alterando a ponte Norte-Sul, onde usualmente é utilizado o protocolo OpenFlow) de forma a comunicar diferentes ASes e suas redes privadas repassando informações de roteamento entre elas (LIN et al., 2015). Stringer apresenta a solução Cardigan, que implementa uma infraestrutura de rede SDN para o encaminhamento de pacotes entre redes remotas utilizando uma abstração de serviço integrada ao RouteFlow (STRINGER et al., 2014).

Neste trabalho, considerou-se um cenário de PTTs operado com SDN onde o controlador lógico precisa ser alimentado com informações sobre os fluxos elefante ativos na rede do PTT. Com estas informações, um conjunto de aplicações pode otimizar a rede SDN de acordo com alguns objetivos, tais como reduzir o consumo de energia e diminuir o atraso transversal destes fluxos na rede. Objetiva-se, com esta dissertação, a criação de *snapshots* do tráfego do PTT, e baseado em requisições do operador, identificar os fluxos elefante e os caminhos utilizados por eles na rede do PTT, projetou-se e implementou-se um sistema de gerência para PTTs de forma a obter estes *snapshots*, o que será apresentado no próximo capítulo.

2.3.2 Fluxos Elefante e SDN

Existem diversas estratégias para a identificação de fluxos elefante em SDN. Uma simples porém ingênua estratégia, é definir entradas específicas na tabela de fluxos, de forma a cap-

turar os fluxos elefante utilizando apenas o protocolo OpenFlow (MCKEOWN et al., 2008). Nesta estratégia, são instaladas regras em cada *switch* da rede. Se um fluxo bater com uma ou mais regras de identificação, então o fluxo é classificado como um fluxo elefante. Outras estratégias propostas no contexto de redes de *datacenter* baseadas em SDN incluem consultas hierárquicas de estatísticas (LIN et al., 2014), detecção baseada no hospedeiro (CURTIS; KIM; YALAGANDULA, 2011), e combinação de amostragem e limiares (CURTIS et al., 2011).

Os problemas na utilização destas soluções, quando da aplicação em PTTs, são listados a seguir. Primeiro, é necessário que seja coletada uma grande quantidade de tráfego de forma a permitir ao operador do PTT distinguir os fluxos existentes, o que pode diminuir consideravelmente a velocidade do processo de identificação. Segundo, são necessárias diversas entradas para cada potencial fluxo elefante, o que pode levar a exaustão das tabelas de fluxos dos *switches*, se o número de fluxos elefante identificados for alto. Finalmente, estas soluções podem requerer modificações nos dispositivos finais ou demandar suporte a *hardwares* especiais, o que não é possível em todos os PTTs.

Entre os trabalhos que realizam controle de fluxos e buscam mitigar fluxos elefante dentro de redes de SDN, podemos citar o DevoFlow, que altera os *switches* OpenFlow adicionando novas funcionalidades e contadores ao *hardware*, de forma a facilitar a identificação dos fluxos na rede. Além disso, a solução também utiliza amostragem para melhorar a performance. A adição desta característica tem como objetivo otimizar o tempo de identificação dos fluxos e diminuir a quantidade de requisições ao controlador (CURTIS et al., 2011).

O liteFlow, é uma solução de monitoramento de fluxos baseado nas informações armazenadas nas *flow tables* dos *switches* OpenFlow. A principal justificativa para a não utilização de amostragem de fluxos é o alto consumo deste modelo e a dificuldade de obter uma amostra confiável quando os fluxos são muito pequenos (GROVER; AGARWAL; KATAOKA, 2015). No trabalho realizado por Narayanan, é apresentada uma solução implementada diretamente nos *switches*, onde os fluxos pequenos, ou *microflows*, são direcionados para um subsistema que por sua vez encaminha estes pacotes na rede de forma tradicional, enquanto os fluxos elefante são direcionados para o protocolo OpenFlow. Para determinar se um fluxo é considerado elefante ou não, é utilizado o protocolo sFlow, já implementado nos *switches* (NARAYANAN et al., 2012).

A solução de identificação e mitigação de anomalias apresentada por Giotis, utiliza uma solução mista de sFlow e SDN, sendo que o sFlow é somente utilizado para a identificação dos fluxos, sem validação pelos contadores do OpenFlow. Após identificado um possível ataque, regras de controle são adicionadas via OpenFlow na rede (GIOTIS et al., 2014).

O *framework* OpenSample apresenta uma solução para gerenciamento de fluxos em rede SDN utilizando sFlow. Usando o cabeçalho coletado dos pacotes TCP, a ferramenta avalia o número sequencial do cabeçalho para medir a média de fluxos por segundo e desta forma gerar as estatísticas de cada fluxo. Diferentemente da nossa ferramenta, eles não utilizam, em nenhum momento, as estatísticas dos *switches* e não fazem uma verificação completa do fluxo na rede

(SUH et al., 2014). O trabalho realizado por Zhang apresenta uma solução para detecção de anomalias baseada em SDN, o qual realiza um *zooming* nas regras de forma a identificar os fluxos elefante. A partir da realização de um cálculo matemático, e da utilização de limiares, o algoritmo cria regras mais específicas a fim de otimizar a identificação (ZHANG, 2013).

No trabalho de Afek *et al*., os autores descrevem uma solução para amostragem em Redes Definidas por Software, utilizando apenas o protocolo Openflow 1.3. Após a captura das amostras, o artigo apresenta uma avaliação de três algoritmos para identificação de fluxos grandes, sendo eles: *Sample&Pick*, *Sample&Hold* e *Sample&HH* (AFEK et al., 2015). Já no trabalho apresentado por Xiao *et al*., os autores demonstraram o uso de algoritmos de aprendizagem de máquina para a identificação de fluxos elefante em uma rede OpenFlow. Utilizando o algoritmo C4.5 de decisão baseado em árvores, a classificação ocorre a partir da captura dos fluxos por um analisador de tráfego como o Wireshark ou o tcpdump. Entretanto, não é realizada a implementação de um mecanismo de mitigação dos fluxos elefante na rede (XIAO et al., 2015).

O artigo apresentado por Bi relata uma solução para identificação de fluxos elefante em estruturas mistas com encaminhamento tradicional e Openflow, verificando via amostragem os pacotes que ultrapassam um limiar dinâmico, definido através do tamanho médio do fluxo na rede. Após ultrapassar o limiar, são adicionadas ao controlador regras para validar se aquele fluxo realmente é um fluxo elefante. Se for confirmado, o fluxo então recebe um tratamento separado no controlador (BI et al., 2013). O *framework* Mahout para mitigação de fluxos elefante em redes SDN, apresentado por Curtis, diferente das outras soluções, adiciona uma camada no Sistema Operacional do equipamento de destino, que retorna informações do *stack* de rede local. O *framework* então utiliza as estatísticas obtidas por essa camada para identificar os fluxos elefante. Desta forma, o protocolo Openflow é apenas utilizado para aplicar as alterações ao tratamento do fluxo, e não na identificação dos mesmos (CURTIS; KIM; YALAGANDULA, 2011).

Nos dois trabalhos apresentados por Afaq *et al.*, os autores descrevem uma solução de identificação e mitigação de fluxos elefante em redes de *data centers* utilizando sFlow e OpenFlow. Na identificação é utilizado o sFlow, capturando amostragens da rede a fim de classificar os fluxos a partir de limiares pré estabelecidos para UDP, TCP e ICMP. Após, os fluxos selecionados são enviados para uma aplicação, a qual altera a prioridade destes nos *switches*, através da implementação de filas QoS via Floodlight. Contrariamente, a solução proposta nesta dissertação, o uso exclusivo de amostragem apresenta um nível de identificação consideravelmente baixo, além de não apresentar uma visão completa da rede, sendo necessário que todos os *switches* reportem uma quantidade mínima de amostras para que o fluxo seja mitigado fim-a-fim (AFAQ; REHMAN; SONG, 2015a) (AFAQ; REHMAN; SONG, 2015b).

Liu *et al.* apresenta uma solução de roteamento por múltiplos caminhos, utilizando Openflow de forma a diminuir os gargalos existentes, que ocorrem por causa dos fluxos elefante, na utilização de algoritmos padrões de múltiplas rotas, a exemplo do ECMP. Embora seja descrito de forma superficial, a identificação dos fluxos ocorre através de uma aplicação nos dispositivos

finais que monitoram os fluxos TCP, repassando para a ferramenta proposta os fluxos que ultrapassarem um limiar predefinido. Para o roteamento, é utilizada a opção *Group Tables* definida pela especificação 1.1 do protocolo OpenFlow (LIU et al., 2014).

Utilizando apenas o Protocolo Openflow, o trabalho apresentado por Lin discute sobre uma estrutura hierárquica de estatísticas onde regras mais específicas são instaladas em etapas a fim de descobrir fluxos elefante na rede. A ferramenta busca dentro das regras instaladas no *switch* quais ultrapassaram um determinado limiar e divide cada uma dela em quatro regras menores, de forma recursiva até encontrar o fluxo elefante (LIN et al., 2014).

2.3.3 Fluxos Elefantes em Pontos de Troca Definidos por Software

Durante esta dissertação não foram encontrados trabalhos que tentassem identificar e mitigar fluxos elefante em PTTs baseados em redes SDN. Embora tenha sido observado que diversos trabalhos possuem como objetivo a gerência de fluxos inter-AS ou de PTTs, e outros em que se mitiga os fluxos elefante em redes SDN, os nuances na resolução deste problema de gerenciamento em redes de PTTs ainda não foram abordados.

A solução apresentada por Zhang visa melhorar o desempenho do encaminhamento de pacotes em *backbones* na Internet identificando os fluxos elefante e mantendo informações referentes a eles nas tabelas de encaminhamento dos *switches* a partir da implementação via *hardware* de um algoritmo misto de *Bloom Filters* e *least recent used* (LRU) (ZHANG; WANG; LAN, 2015). Este é o trabalho mais próximo do nosso cenário, no qual é realizada a mitigação de fluxos elefante. Porém esta solução não utiliza SDN, o que pode reduzir sua performance e escopo.

Resumindo, as propostas correntes para identificação/mitigação de fluxos elefante não suprem a necessidade dos PTTs baseados em SDN. E considerando isso no próximo capítulo será introduzida e detalhada uma proposta de sistema de monitoramento/gerenciamento para PTTs baseados em SDN de forma a superar as limitações acima citadas.

3 GERENCIANDO FLUXOS ELEFANTES EM PTTS BASEADOS EM SDN

3.1 Sistema de Monitoramento para PTTs baseados em SDN

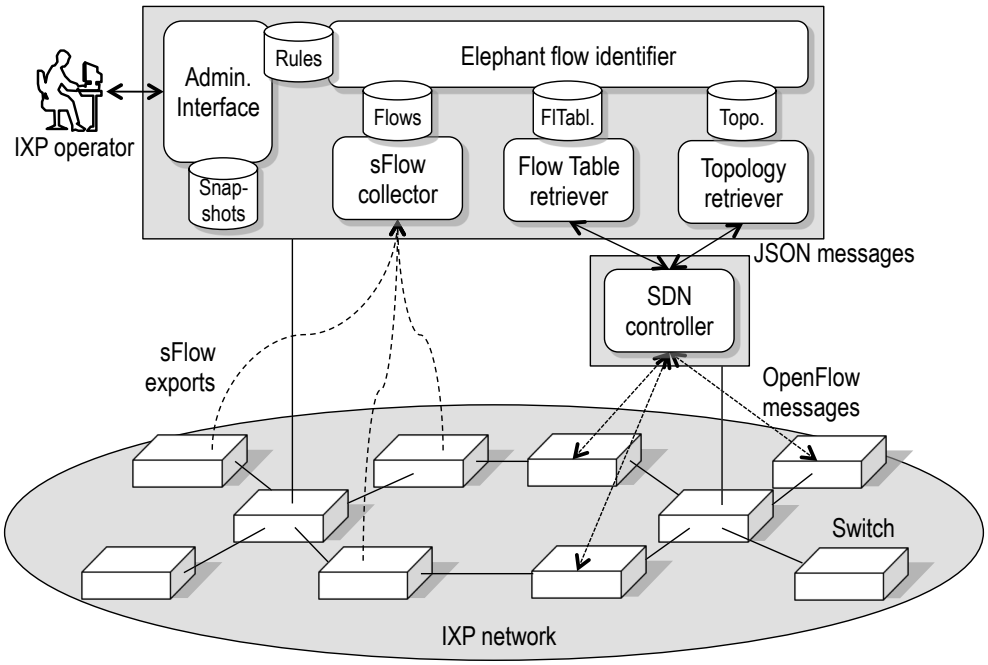
Desenvolveu-se nesta dissertação um sistema de monitoramento chamado SDEFIX (*Software-Defined Elephant Flow Identifier for Internet Exchange*), o qual permite ao operador de um PTT identificar fluxos elefante e os caminhos de rede associados. Definiu-se a solução em um cenário aonde a rede do PTT é controlada por um controlador SDN logicamente centralizado, o qual roda aplicações que por sua vez controlam os *switches* subjacentes. Os *switches* se comunicam com o controlador através do protocolo OpenFlow. Cada *switch* Openflow mantém uma tabela de fluxos (*flowtable*) que é inspecionada considerando-se os pacotes de entrada a fim de determinar a ação que será tomada (*e.g.*encaminhamento do pacote a algum porta específica) (MCKEOWN et al., 2008).

3.1.1 Arquitetura do Sistema

Para ultrapassar as limitações de escalabilidade dos trabalhos que se propõem a identificar fluxos elefante no contexto de PTTs, desenvolveu-se uma solução que utiliza sFlow (SFLOW, 2015) combinada com o protocolo OpenFlow. O protocolo sFlow permite monitorar um grande número de fluxos por empregar um método de amostragem. Usando o sFlow, depois que um número definido de pacotes é recebido por um determinado *switch*, este envia um conjunto de amostras dos pacotes trafegados a um servidor remoto chamado coletor. Na solução apresentada nessa dissertação, o coletor armazena as amostras em um banco de dados MongoDB, do tipo NoSQL. O MongoDB provê uma estrutura flexível, possibilitando um melhor armazenamento das amostras dos pacotes. No MongoDB, o esquema do banco de dados pode ser dinamicamente modificado de acordo com as amostras capturadas, o que permite um armazenamento em uma mesma coleção de diferentes protocolos de redes ou camadas superiores. No total, utilizamos cinco bancos de dados MongoDB para armazenar informações referentes às regras de identificação, às amostras coletadas, às tabelas de fluxos de cada *switch*, à topologia da rede e aos *snapshots* dos fluxos identificados, respectivamente.

A Figura 3.1 retrata a arquitetura do sistema proposto. A arquitetura é composta por cinco componentes principais e seus respectivos banco de dados: *Elephant flow identifier, sFlow collector, Topology retriever, Flow Table retriever* e a *Admin Interface*. O módulo *Elephant flow identifier* compara as amostra de fluxos contra as regras de identificação e os limiares definidos pelo operador do PTT através do módulo *Admin interface* e retorna o caminho de rede completo de cada fluxo elefante identificado. O módulo *sFlow collector* coleta amostras de fluxos em determinados intervalos e envia estas amostras de fluxos para o banco de dados. Os módulos *Topology retriever* e *Flow Table retriever* comunicam-se com o controlador SDN e buscam informações referentes a topologia de rede e as tabelas de fluxos de todos os *switches* e atu-

Figura 3.1: Arquitetura do sistema de identificação para PTTs baseados em rede SDN - SDEFIX



Fonte: do Autor (2016).

alizam as coleções de *topology* e *flow table* no banco de dados, respectivamente. O módulo *Admin interface* permite a visualização dos caminhos dos fluxos elefante e das *snapshots* dos fluxos identificados anteriormente, que podem ser usados como registros históricos, ou como fontes de dados para o gerenciamento da infraestrutura de TI. Para cada *switch* da rede, são capturadas amostras do sFlow com um tempo específico de consulta. Implementamos o intervalo convencional de 10 segundos na configuração padrão do sFlow.

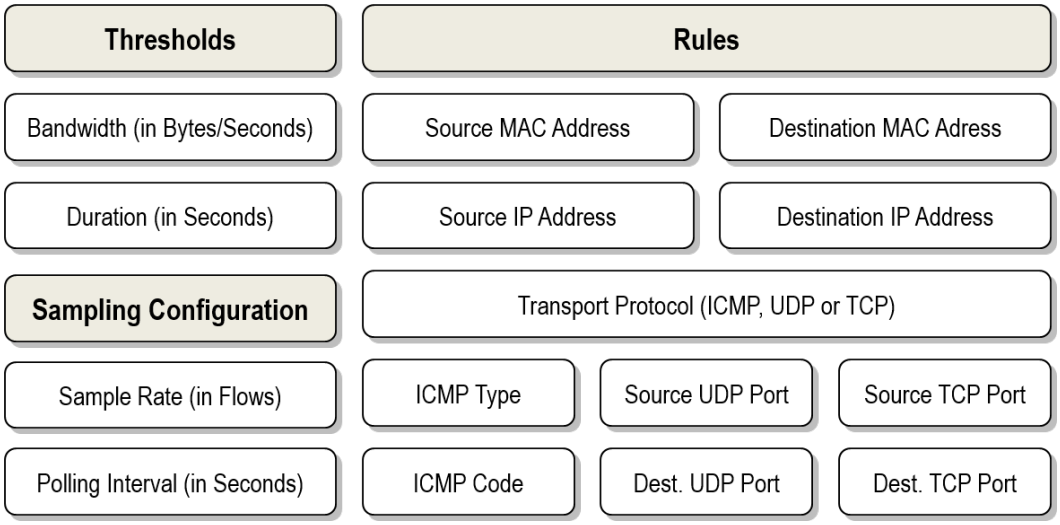
Embora a solução aqui apresentada seja desacoplada de qualquer controlador SDN em particular, utilizou-se o controlador OpenFlow Ryu (RYU, 2015) em sua implementação, por este ser um dos poucos controladores a suportar as últimas versões do protocolo OpenFlow e implementar uma REST API bem definida.

3.1.2 Especificação dos Fluxos Elefantes

Na solução proposta nesta dissertação, o operador do PTT é responsável por definir o que é um fluxo elefante. O operador do PTT especifica regras de identificação que caracterizam os fluxos elefante baseado nas seguinte informações: Endereço IP de origem e destino, Endereço MAC de origem e destino, protocolo de transporte e número das portas utilizadas. O operador do PTT pode utilizar multiplas regras para identificar os fluxos elefante e apenas regras explicitamente escritas são reconhecidas pelo algoritmo. Adicionalmente, são definidos globalmente limiares de largura de banda trafegada (em bytes/segundos) e de duração (em segundos) para todos os fluxos. A Figura 3.2 apresenta uma lista de todos os campos de entrada do sistema,

incluindo os campos para criação de novas regras de identificação dos fluxos, para a definição dos limiares de largura de banda e duração e para as configurações de amostragem, com tempo de coleta e taxa de amostragem.

Figura 3.2: Campos de entrada do *framework* SDEFIX



Fonte: do Autor (2016).

As regras de identificação de fluxos são então utilizadas pelo identificador de fluxos elefante que periodicamente compara estas regras com os fluxos ativos na rede, procurando por potenciais candidatos. Uma visão detalhada deste módulo é descrita na próxima subseção.

3.1.3 Identificação dos Fluxos Elefantes

Atualmente, alguns trabalhos apresentam soluções para identificação de fluxos elefante em redes SDN. O artigo apresentado por Bi descreve uma solução para identificação de fluxos elefante presentes em estruturas mistas com encaminhamento tradicional e Openflow, verificando via amostragem os pacotes que ultrapassam um limiar dinâmico, criado através do tamanho médio do fluxo na rede. Se o limiar for ultrapassado, é adicionado ao controlador regras para validar se aquele fluxo realmente é um fluxo elefante. Se for confirmado, o fluxo então recebe um tratamento separado no controlador (BI et al., 2013). Já no trabalho realizado por Afaq, é apresentada uma solução de identificação utilizando sFlow, capturando amostras da rede a fim de classificar os fluxos a partir de limiares pré estabelecidos para UDP, TCP e ICMP (AFAQ; REHMAN; SONG, 2015b). Para isso foi utilizada a ferramenta da InMon sFlow-RT (SFLOW, 2015).

Diferentemente das soluções descritas acima para identificação de fluxos elefante, nosso sistema permite a identificação dos caminhos de rede associados a eles. As soluções atuais executam a identificação apenas localmente, sem uma visão completa da rede. Isto é, a identificação de fluxos é executada individualmente por *switch*. O problema com esta abordagem é

que o operador do PTT não possui um lista compilada dos dispositivos transitados pelos fluxos elefante na rede. Tendo em vista que o sistema mantém a sequência de todas as tabelas de fluxos de todos os *switches*, podemos validar os fluxos identificados com as tabelas de fluxos disponíveis. Se um *switch* possuir uma entrada na tabela de fluxos correspondente ao do fluxo elefante, podemos incluir este *switch* no caminho percorrido pelo fluxo elefante. Desta maneira, é possível derivar o caminho completo realizado por cada fluxo elefante, ajudando o operador do PTT a tomar decisões de gerência apropriadas para cada tipo de fluxo.

Algoritmo 1 Identificação dos Fluxos Elefantes

```
1: R: set of rules defined by the IXP operator
2: S: set of collected flow samples
3: T: physical topology
4: F(sw): flow table of the switch sw ∈ T
5: EF: set of identified elephant flows
6: EFP(ef) network path of the elephant flow ef ∈ EF
7: Elephant Flow Identification:
8: EF ← ∅
9: for each s ∈ S do
10:     for each r ∈ R do
11:         if s matches r and s.bandwitdh > r.bandwitdh_threshold and s.duration > r.duration_threshold then
12:             add s to EF
13:         end if
14:     end for
15: end for
16: Elephant Flow Path Discovery:
17: for each ef ∈ EF do
18:     EFP(ef) ← ∅
19:     for each sw ∈ T do
20:         for each f ∈ F(sw) do
21:             if ef = f then
22:                 add sw to EFP(ef)
23:             end if
24:         end for
25:     end for
26: end for
```

O Algoritmo 1 ilustra o processo de identificação de fluxos elefante. A iniciliazação do algoritmo acontece quando o operador do PTT deseja verificar os fluxos elefante que estão ativos na rede do PTT em um determinado momento juntamente com os caminhos de rede percorrido por eles. Na fase de identificação dos fluxos elefante, o sistema compara as amostras coletadas com as regras definidas pelo operador do PTT (linhas 9-11). Se alguma amostra coincidir com alguma regra pré-definida, o fluxo correspondente é adicionado ao conjunto de fluxos elefante identificados (linha 12). Após isto, o algoritmo computa os caminhos de rede associados a cada fluxo elefante identificado, o que realiza uma checagem das tabelas de fluxos

de cada *switch* da topologia a fim de encontrar uma entrada de fluxo correspondente ao fluxo elefante (linhas 17-21). Se uma entrada é encontrada, o *switch* é incluído no caminho percorrido pelo fluxo elefante (linha 22).

Como resultado, uma lista de fluxos elefante correspondentes às regras pré-estabelecidas é retornada para a interface administrativa, de forma a entregar uma visualização apropriada pelo operador do PTT.

3.1.4 Apresentando os caminhos dos Fluxos Elefante

Recursos de visualização e monitoramento podem auxiliar os operadores de um PTT nas tarefas de controle e configuração de uma rede (MACHADO et al., 2014) (TAVARES GUIMARAES et al., 2014). Nesta perspectiva, redes SDN requerem uma atenção especial, principalmente porque elas apresentam um alto nível de customização e programabilidade. Neste sentido, soluções tradicionais de visualização e monitoramento não foram desenvolvidas para lidar com os requisitos de uma rede SDN, principalmente pela separação do plano de encaminhamento do plano de controle e a necessidade de gerenciá-los de forma homogênea. Portanto, uma ferramenta desenvolvida exclusivamente para a visualização e monitoramento de redes SDN pode aprimorar o entendimento do comportamento da rede e simplificar outras tarefas de gerenciamento (ISOLANI et al., 2015).

Após receber as informações vindas dos módulos *Topology Retriever* e do *Elephant Flow Identifier*, a *Admin interface* retorna ao usuário uma visualização da topologia física, conforme mostrado na figura 3.3. O administrador pode então visualizar cada fluxo elefante identificado individualmente na topologia ou combinar múltiplos fluxos elefante em uma única visualização para realçar os caminhos de rede mais usados.

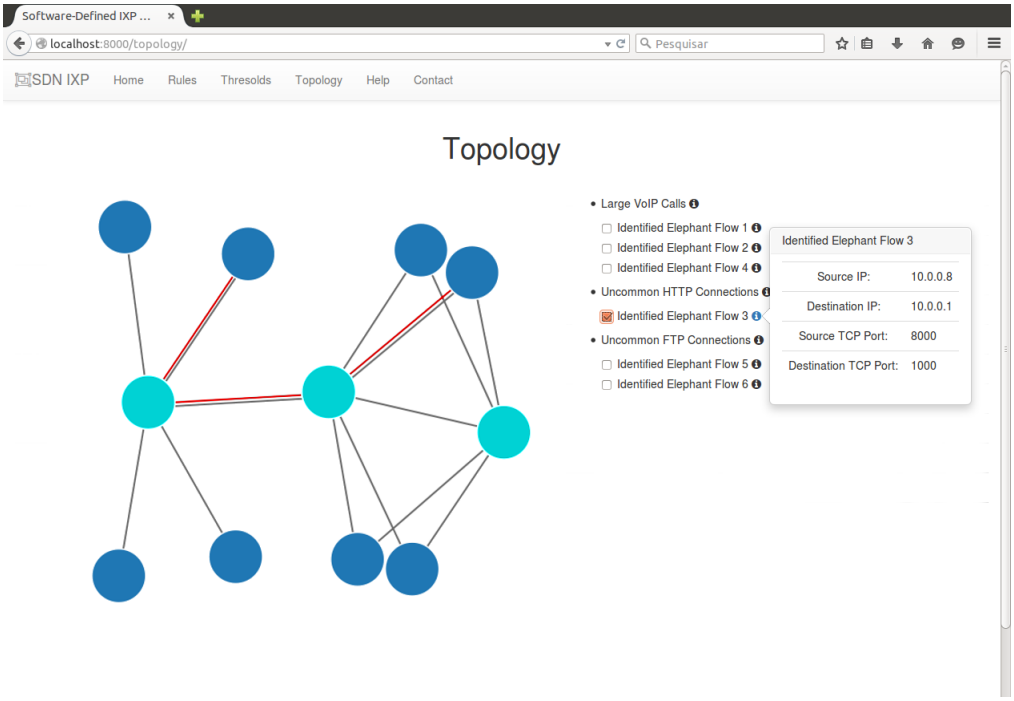
O operador pode então gerar um *snapshot* do atual estado da rede e os fluxos elefante identificados para análise futura. Além disso, *snapshots* podem ser utilizados para construir um histórico dos fluxos elefante identificados e seus respectivos caminhos, e com isso, auxiliar o operador no plano de atualização da infraestrutura do PTT. Quando o sistema é acessado novamente, ele retorna o último *snapshot* salvo para visualização do usuário.

Adicionalmente, a interface administrativa foi desenvolvida com o *framework* Django (FRAMEWORK, 2015) que implementa um modelo MVT (Model-View-Template) como padrão de projeto. Ademais, a visualização da topologia foi implementada utilizando a biblioteca D3js (D3JS, 2014), uma poderosa ferramenta que utiliza Javascript e SVG para criar visualizações interativas.

3.2 Sistema de Recomendação para PTTs baseado em SDN

Desenvolveu-se com essa dissertação um sistema de recomendação integrado ao *framework* SDEFIX que permite o operador do PTT mitigar o impacto dos fluxos elefante da rede do

Figura 3.3: Visualização da Topologia na *Admin interface*



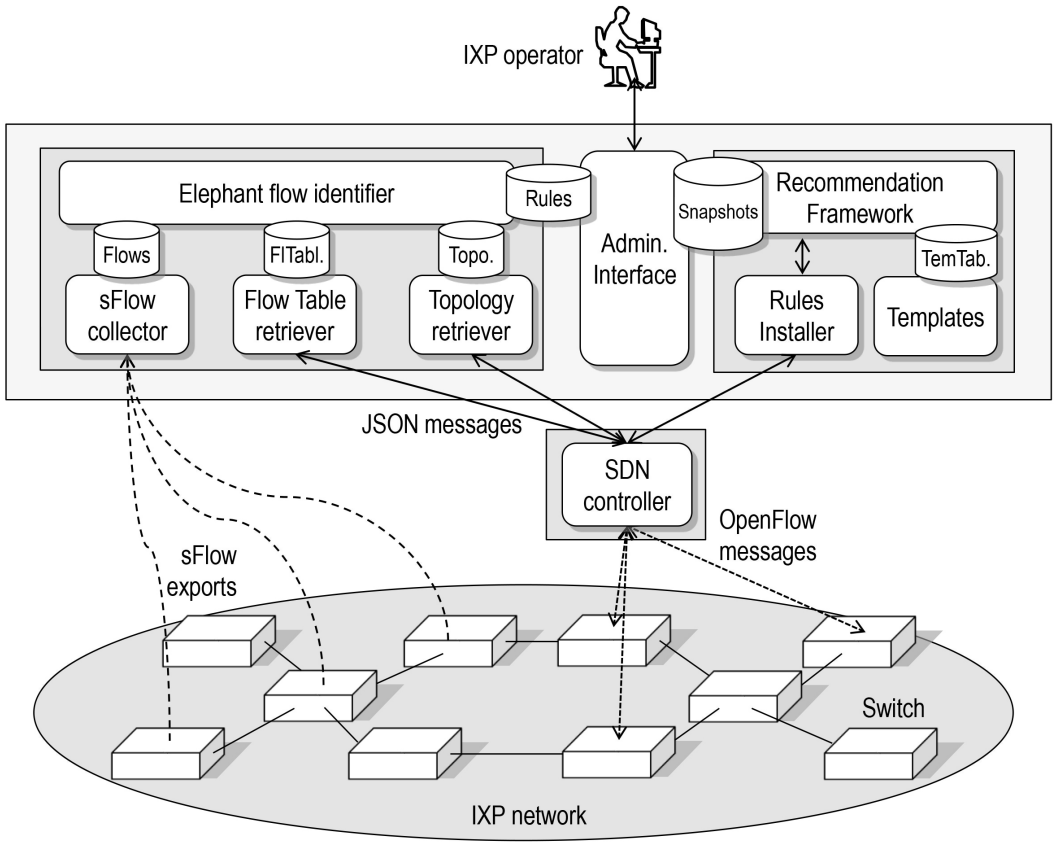
Fonte: do Autor (2016).

PTT. Consideramos um ambiente onde a rede do PTT é gerenciada por um controlador SDN logicamente centralizado que define quais caminhos físicos serão usados dentro da comunicação do PTT. Os *switches* da infraestrutura se comunicam com o controlador através do protocolo OpenFlow (MCKEOWN et al., 2008).

3.2.1 Arquitetura do Sistema

Utilizou-se, como entrada para o sistema de recomendação, os *snapshots* dos fluxos elefante gerados pelo SDEFIX. A figura 3.4 detalha as adições do sistema de recomendação à arquitetura existente do SDEFIX. Além de adicionar funções específicas ao módulo *Admin Interface* para ajudar na administração do *framework* pelo operador do PTT (ex. controle de acesso para os novos módulos), a infraestrutura possui 3 novos modelos: *Rules Installer*, *Template Management*, *Recommendation Framework*, e seus respectivos bancos de dados, quando necessário. O módulo *Recommendation Framework* aplica *templates*, que são algoritmos criados para modificar os fluxos existentes, utilizando os *snapshot* dos fluxos elefante coletados. O operador do PTT é responsável por definir e adicionar *templates* ao módulo. Uma vez que o operador escolha o *template*, o *framework* aplica ele a todos os caminhos dos fluxos elefante existentes de um determinado *snapshot* da rede do PTT. O resultado da aplicação do *template* sobre o *snapshot* é um conjunto de recomendações de regras SDN (regras do protocolo Openflow) que podem ser aplicadas na infraestrutura. O operador do PTT pode, ao seu interesse, ajustar as recomenda-

Figura 3.4: Arquitetura do sistema de recomendação para PTTs baseados em redes SDN



Fonte: do Autor (2016).

ções antes de aplicá-las na rede, por exemplo, removendo *switches* ou *links* específicos para o caminho sugerido e rodando novamente o *template* com as modificações propostas. O módulo *Template Management* é o local onde o operador do PTT pode adicionar diferentes *templates* que podem cumprir diversas funções, como balanceamento de carga, otimização da utilização de enlaces, garantia de banda e limite de atraso, etc. Finalmente o módulo *Rules installer* é responsável por enviar as regras de fluxos para o controlador SDN após o operador do PTT validar o resultado do *template*. O controlador SDN, então, instala as regras nos *switches* apropriados utilizando o protocolo OpenFlow.

O Algoritmo 2 ilustra como o sistema de recomendação funciona. O algoritmo é iniciado pelo operador do PTT quando este deseja realocar/gerenciar os fluxos elefante que estão ativos na rede do PTT. Primeiro, o operador deve escolher qual dos *templates* disponíveis no sistema será utilizado. Um exemplo de como um *template* pode funcionar é apresentado no Algoritmo 3. Em seguida, o *template* é aplicado para cada fluxo elefante presente no *snapshot* (linhas 9-13). Para cada recomendação gerada, o operador pode realizar alterações no resultado da recomendação de forma a remover qualquer característica indesejada nele, como enlances que

possuam altas latências, rodando novamente o *template* escolhido (linhas 14-20). Se a recomendação é validada pelo operador do PTT, as regras são então enviadas para o controlador SDN para a instalação destas (linhas 17-18). O algoritmo 3 é um exemplo de *template* que pode ser aplicado sobre um *snapshot* de fluxos elefante. Este algoritmo tem como estratégia, encontrar o melhor caminho alternativo (BAP, do inglês *Best alternative path*) para um determinado fluxo, através do aumento do custo dos *links* do caminho original e recomputando o menor caminho entre os pontos de origem e destino de um fluxo elefante.

Algoritmo 2 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**

Algoritmo 3 Example of ApplyTemplate function - Best Alternative Path (BAP)

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

Embora tenha-se utilizado o controlador OpenFlow Ryu (RYU, 2015), por sua bem definida REST API, a solução apresentada é desacoplada de qualquer controlador SDN em particular

e pode ser portada para outros controladores modernos, como o OpenDaylight (OPENDAYLIGHT. . . , 2016).

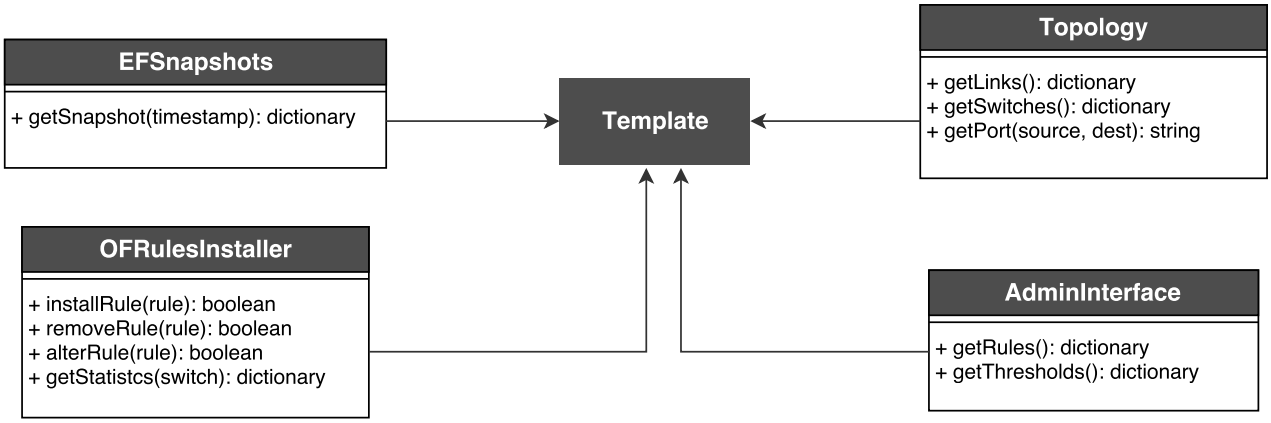
3.2.2 Especificação dos Templates

Na solução proposta, o operador do PTT pode adicionar ou criar *templates* para reduzir o impacto dos fluxos elefante na rede. Soluções comuns incluem estratégias de novas rotas, garantia de qualidade de serviço (QoS), ou otimização do gerenciamento de energia.

Um *template* utilizado para exemplificar o *framework* de recomendação proposto é apresentado no Algoritmo 3. Ele computa o menor caminho alternativo, ou o segundo menor caminho, para cada fluxo elefante armazenado no SDEFIX. A ideia por trás dele é migrar o fluxo elefante para um novo, e possivelmente não congestionado caminho, enquanto não aumenta significantemente a latência da rede.

Os *Templates* desenvolvidos para nossa ferramenta de recomendação são escritos em Python e utilizam uma API bem definida, conforme a Figura 3.5. As classes disponibilizadas por nossa API trazem métodos que auxiliam o operador do PTT a escrever um *template*. Por exemplo, informações sobre a topologia física podem ser obtidas através dos métodos *getLinks*, *getSwitches* e *getPorts* da classe *Topology*. De igual forma, a classe *OFRulesInstaller* disponibiliza métodos que permitem instalações de novas regras ou modificações nas regras instaladas pelo controlador OpenFlow. O algoritmo do *template* BAP pode ser encontrado no capítulo 4.

Figura 3.5: Descrição da API para a construção de *templates*



Fonte: do Autor (2016).

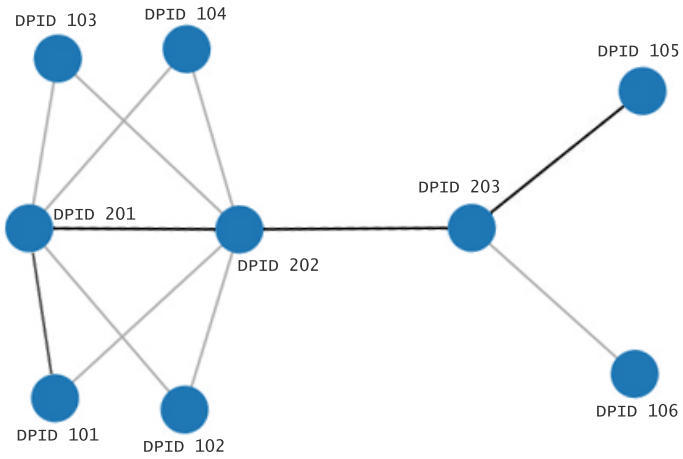
3.2.3 Validação pelo operador do PTT

Uma parte essencial para o *framework,* é a relação existente entre o operador do PTT, entre as recomendações sugeridas, e entre a aplicação destas na rede. Por possuir uma infraestrutura sensível a mudanças, em que cada alteração pode trazer consequências como *loops* de tráfego,

configurações erradas no BGP ou quedas de serviço, um controle de grão fino sobre todas as alterações na rede é necessário. Além disso o operador do PTT necessita gerenciar as diversas SLAs (Acordo de nível de serviço, do inglês *Service level agreement*) às quais o PTT está vinculado e sistemas automatizados devem ser constantemente monitorado de forma a assegurar a correta operação do PTT.

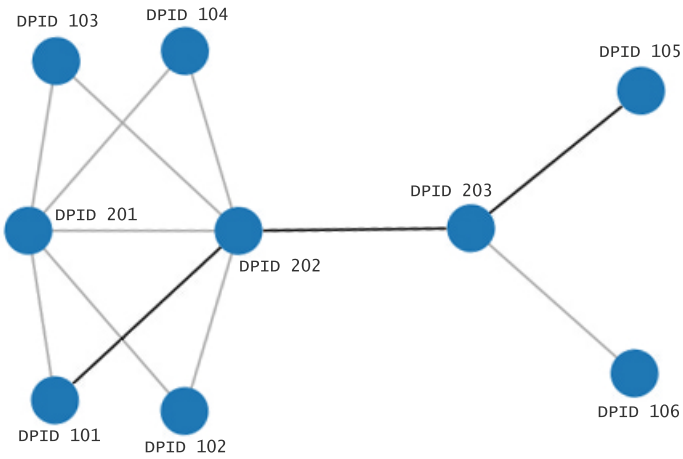
3.2.4 Exemplos de uma recomendação

Figura 3.6: *Snapshot* utilizado como entrada



Fonte: do Autor (2016).

Figura 3.7: Novo caminho do fluxo elefante após a validação do *template*



Fonte: do Autor (2016).

Para ilustrar o uso do sistema de recomendação do SDEFIX, apresentamos um exemplo trivial. Um *snapshot* usado como entrada para o sistema de recomendação é apresentado na

Figura 3.6. O fluxo elefante entra na rede do PTT por um *switch* identificado como DPID 101 e tem como destino o *switch* DPID 105. Foi simulada a aplicação do *template* de melhor caminho alternativo (Algoritmo 3) pelo operador do PTT. Após o *template* ser aplicado, este gerou a recomendação ilustrada na Figura 3.7, que remove um salto do caminho percorrido anteriormente pelo fluxo elefante. Isto acontece devido ao controle de *loopings* existente na topologia através de protocolos, como o *Spanning Tree Protocol* (STP) que é ignorado pelo *template* BAP, a qual por sua vez altera apenas a rota do fluxo elefante, sem alterar contudo as configurações padrões do funcionamento da rede. Além disso, o operador do PTT pode aplicar as recomendações sugeridas sem nenhuma modificação, ou pode alterar a saída removendo, por exemplo, um *switch* ou um *link* e executando novamente o *template*.

No que tange aos esforços necessários para que o operador do PTT possa aprimorar o sistema proposto para que este funcione de forma otimizada em seu cenário, o operador poderá implementar ou modificar *templates* que em última análise, definirão como os fluxos elefante serão alocados dentro da rede do PTT. As diversas opções de estratégias que podem ser desenvolvidas possuem como limitações as funções definidas na API descrita na Figura 3.5.

4 IMPLEMENTAÇÃO DO PROTÓTIPO

Neste capítulo serão apresentados os detalhes de implementação dos componentes da arquitetura do SDEFIX ilustrada na Figura 3.1. Conforme discutido no final do Capítulo 3, o *framework* possui a função de identificar os fluxos elefante a partir das definições do operador do PTT. Então o caminho de cada um dos fluxos é entregue a um sistema de recomendação, que retorna a partir de determinados *templates* uma solução para o tratamento dos fluxos na rede. Na Seção 4.1 são apresentadas as ferramentas e bibliotecas utilizadas no desenvolvimento da solução. Na Seção 4.2 é apresentado o modelo de dados do *framework* proposto. Na seção 4.3 são apresentados os detalhes na obtenção de informações e estatísticas da rede. Na Seção 4.4 é detalhada a implementação dos módulos de identificação dos fluxos elefante e descoberta de seus caminhos na rede. E finalmente, na Seção 4.5 é apresentada a implementação do sistema de recomendação.

4.1 Implementação do Ambiente de Desenvolvimento

Nesta seção, será mostrado um pequeno resumo das principais ferramentas utilizadas na implementação e execução do *framework* SDEFIX:

- Controlador de Rede: O Ryu (RYU, 2015) é um controlador OpenFlow desenvolvido em Python, e fornece uma REST API, que pode ser estendida e modificada, que retorna informações como estatísticas da rede ou alterações na topologia. O controlador Ryu possui uma comunidade de desenvolvedores ativa, que buscam implementar as últimas novidades do OpenFlow antes de qualquer outro controlador disponível. Além disso, toda a captura das informações de redes solicitadas pelo SDEFIX são realizadas via REST e recebe como retorno um arquivo em formato JSON. Esta escolha possui diversas razões: primeiro, outros controladores além do Ryu, como o Floodlight (PROJECT..., 2016) e o Opendaylight (OPENDAYLIGHT..., 2016), podem exportar requisições para este formato; e segundo, REST é o método mais usado de interface de comunicação entre o plano de aplicação e o plano de controle (SEZER et al., 2013).
- Emulador de Rede: O emulador Mininet2 (TEAM, 2016) foi desenvolvido para oferecer a pesquisadores um ambiente de simulação realístico e confiável para testes e implementações de ideias em SDN. Altamente aceito e reconhecido, o Mininet é o ambiente ideal para testes de roteamento com estruturas complexas, como múltiplos caminhos.
- Protocolo OpenFlow 1.0: Embora outras versões do protocolo Openflow tenham sido propostas, como o OpenFlow 1.5, a maioria das implementações do protocolo não trazem as novidades descritas nestas versões. Por exemplo, o Open vSwitch (OPENVSWITCH, 2015) presente no emulador Mininet, não implementa diversas funcionalidades das versões 1.3 e 1.5 do protocolo. Por esta razão, escolheu-se a versão 1.0 como base para a implementação do *framework* SDEFIX. Entretanto, se este cenário vier a mudar, ou-

tras versões do protocolo podem ser suportadas com pequenas alterações nos módulos *OFRules Installer*, *Flow Table Retriever* e *Topology Retriever*.

- sFlow: Como solução para a amostragem dos pacotes na rede, foi utilizado o protocolo sFlow (SFLOW, 2015). Este foi escolhido por ser um padrão da indústria e estar implementado na maioria dos *switches* comerciais. Possui dois tipos de amostras: uma de pacotes randômicos da rede e outra de contadores de fluxos, que são capturados a partir da configuração de dois parâmetros, tempo de requisição e taxa de amostragem e enviados para um servidor para que sejam utilizados para a análise e relatórios sobre o estado atual da rede.
- Python: Para a implementação do *framework*, foi utilizada a linguagem de programação Python (PYTHON..., 2015). Esta escolha derivou-se das características apresentadas pelo Python, como linguagem de alto nível, interpretada, orientada a objeto, com semântica dinâmica. A utilização de tipagem dinâmica, torna o Python profundamente atrativo para o desenvolvimento rápido de aplicações, bem como na produção de *scripts* e na conexão entre componentes de diferentes programas. Além disso, a linguagem foi escolhida por facilitar a comunicação com o protocolo JSON e o controlador Ryu.
- mongoDB: Diferente das soluções tradicionais, o mongoDB (MONGODB..., 2016) é um banco de dados orientado a objetos, *opensource* e multiplataforma, que utiliza uma linguagem própria ao invés do SQL, tornando ele uma das soluções NoSQL existentes. Ao invés de utilizar tabelas relacionais para o armazenamento de valores, o mongoDB utiliza JSON modificados chamados BSON em coleções. As principais vantagens em utilizar este tipo de estrutura, é o desvinculamento da estrutura de tabela dos objetos, além da possibilidade de executar diferentes tipos de consulta e operações de agregação.
- Django: Para o projeto foi escolhido como *framework* de aplicação o Django (FRAMEWORK, 2015), uma solução *opensource*, escrita em Python, que facilita o desenvolvimento de aplicações para a internet. A principal característica deste *framework* é a utilização do modelo MVT, ou *Model-View-Template*, onde diferentemente dos modelos tradicionais de desenvolvimento, pode-se perceber uma abstração na interface com o usuário baseada em *Templates*, que diminuem o tempo de criação de cada interface.
- Outras Bibliotecas: Além das bibliotecas listadas acima, ainda foram utilizadas no desenvolvimento do *framework*:

  D3.js: uma biblioteca em Javascript para manipulação de documentos e gráficos dinâmicos, utilizando HTML, SVG, e CSS (BOSTOCK, 2016).

  NetworkX: um pacote do Python para a criação e manipulação de estruturas baseado em gráfico e funções complexas de cálculos para redes (NETWORKX..., 2016).

  Bootstrap: Uma solução *opensource* de biblioteca *frontend* para criação de sites e aplicação para a Web (OTTO; CONTRIBUTORS, 2016).

4.2 Modelo de Dados

Como ponto de partida para a definição do modelo de dados do *framework*, foram analisados os objetos que seriam utilizados durante o seu desenvolvimento. Por ter como objetivo a identificação dos fluxos e obtenção das estatísticas da rede, foi descartada a utilização de um banco relacional devido à heterogeneidade das informações da rede a serem adquiridas. Desta forma foi escolhida uma solução NoSQL para o armazenamento dos dados do sistema. O mongoDB, ao invés de trabalhar com tabelas, utiliza um modelo de coleções, onde a estrutura de cada objeto pode ser armazenada de forma diferente, mesmo que em uma única coleção. Este é o ponto fundamental para a escolha deste modelo, já que os dados obtidos tanto das tabelas de fluxos, quanto do sFlow, podem ser armazenados em uma mesma coleção, ainda que possuindo campos diferentes.

Desta forma, a figura 4.1 apresenta as coleções adicionadas ao mongoDB, bem como os principais campos existentes em cada uma delas. Outro ponto fundamental é que o mongoDB não possui um modelo de relacionamento. Por este motivo, foi adicionado quando necessário, o "_id"de um objeto de outra coleção.

Os objetos das coleções *samples* e *sflowpackets* são obtidos pelo *sFlow Collector* e armazenam os pacotes amostrados da rede. É interessante notar que estes objetos podem possui diferentes valores em sua estrutura decorrente do tipo de amostra e os protocolos existentes em cada amostra.

As coleções *switch*, *link* e *flowtable*, são obtidos a partir do controlador OpenFlow pelos módulos *FlowTable Retriever* e *Topology Retriever* e armazenam as informações referentes a infraestrutura de rede e as tabelas de fluxos dos *switches* durante a execução do *framework*.

Por último as coleções *threshold* e *rule* apresentam os limiares e regras definidas pelo operador de rede para a identificação dos fluxos e a coleção *snapshot* armazena os caminhos dos fluxos elefante identificados e salvos pelo operador.

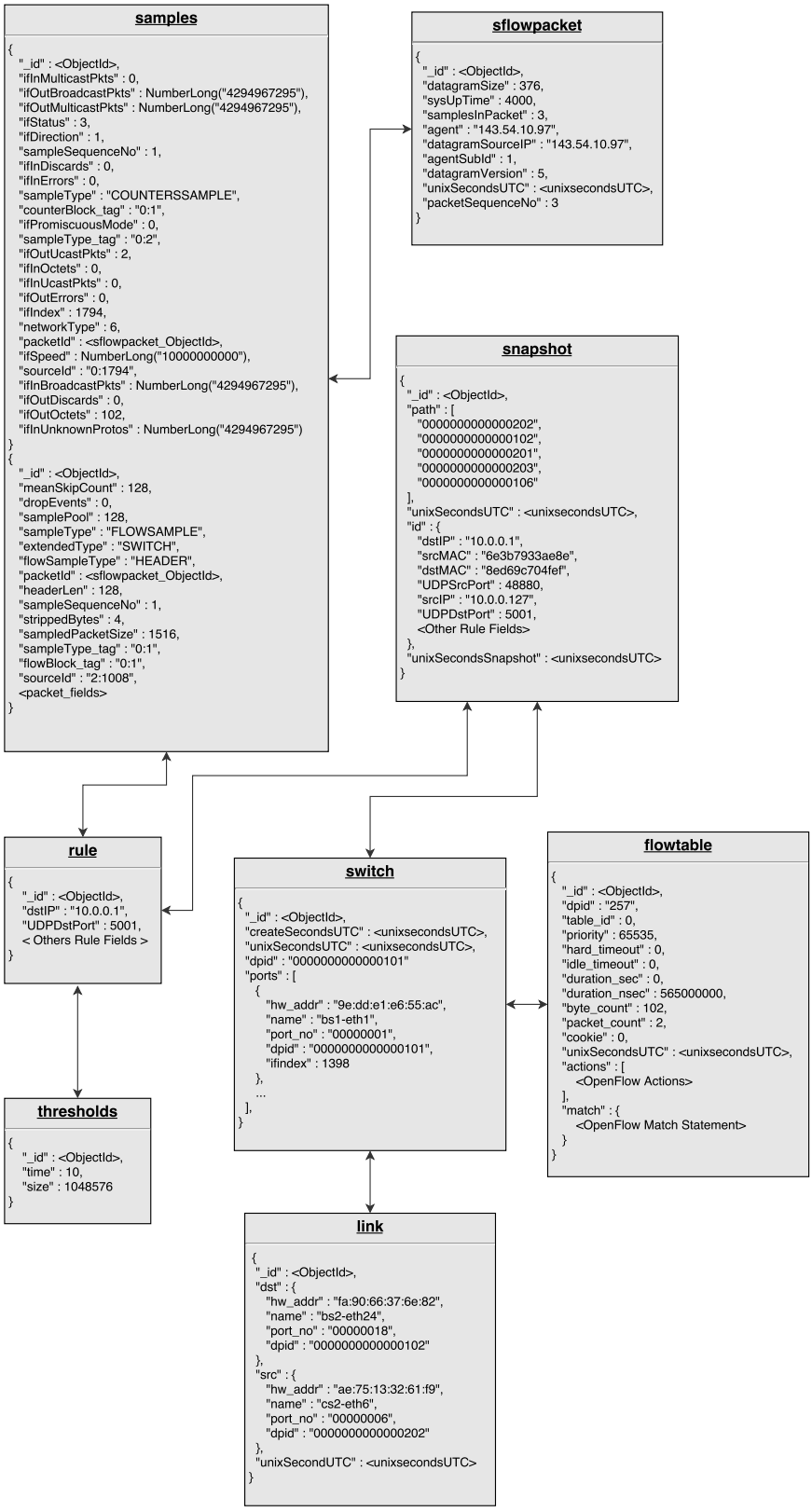
4.3 Obtendo informações da Rede

Para obter as informações do estado atual da rede, três processos permanecem ativos constantemente em nosso *framework*. O primeiro, responsável pela captura das amostras do sFlow, apresentado na arquitetura como o *sFlow Collector*, é implementado de forma a receber os dados apresentados pela ferramenta criada pela inMon *sflowtool*[1], que converte os pacotes binários do sFlow para uma saída mais legível ou para o NetFlow da Cisco enviando para um outro coletor. Estes dados são então convertidos de forma a formatar os objetos presentes na coleção *sflowpacket* e *samples* presente na Figura 4.1. A Figura 4.2, apresenta um exemplo da conversão da saída da ferramenta *sflowtool* para os objetos armazenados nas coleções do mongoDB.

O segundo e o terceiro processos são responsáveis pela comunicação entre o controlador

[1] https://github.com/sflow/sflowtool

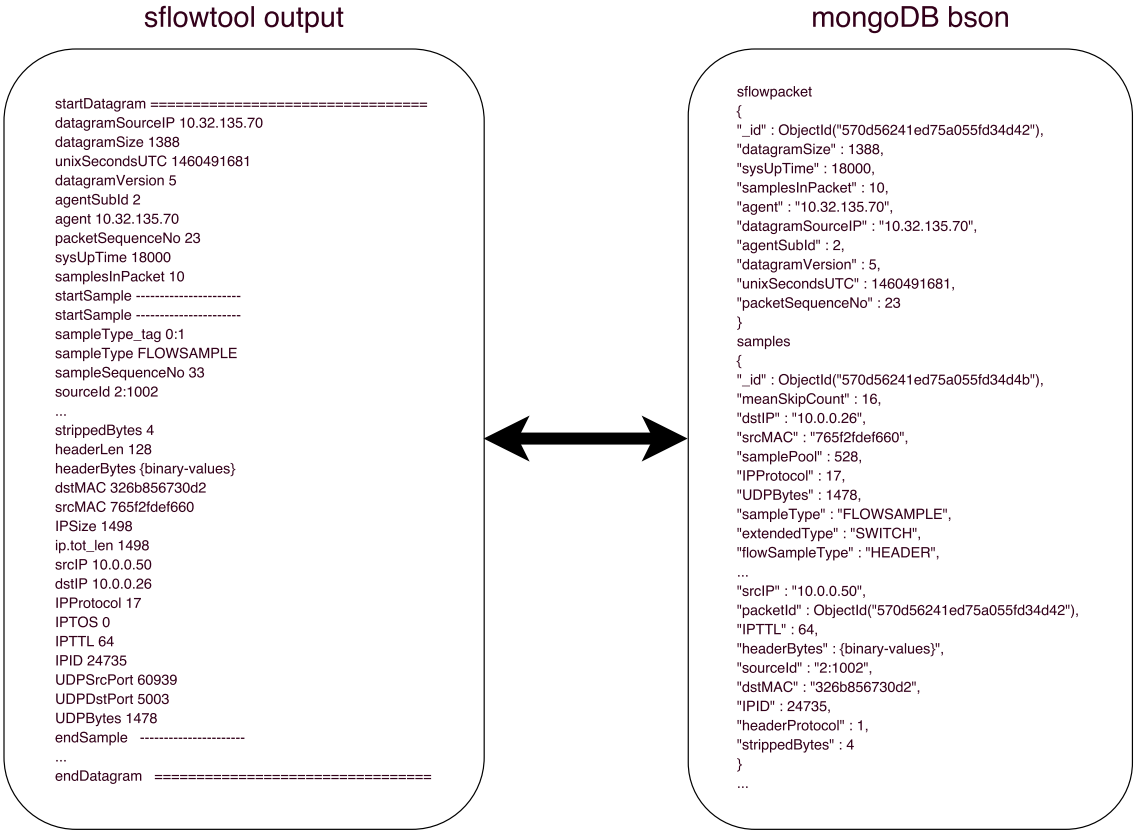
Figura 4.1: Coleções criadas no mongoDB para o SDEFIX



Fonte: do Autor (2016).

SDN e o *framework*, e possuem a incumbência de obter informações sobre as tabelas de fluxos dos *switches* OpenFlow e a topologia da rede. Apresentados na arquitetura como *Flow Table Retriever* e*Topology Retriever*, os processos se comunicam com o controlador OpenFlow via uma REST API e armazenam as informações de uma forma padronizada no mongoDB. No desenvolvimento, utilizou-se o controlador OpenFlow Ryu, que apresenta uma bem definida API, tanto para a obtenção de dados da topologia, quando para estatísticas e tabelas de fluxos. Da mesma forma, requisições de instalação de regras também são enviados via REST, através do método POST.

Figura 4.2: Conversão da ferramenra sflowtool para as coleções *sflowpacket* e *samples*



Fonte: do Autor (2016).

Embora o *framework* esteja vinculado a API REST do Ryu, pequenas alterações poderiam possibilitar a utilização de outros controladores como o OpenDayLight e o Floodlight.

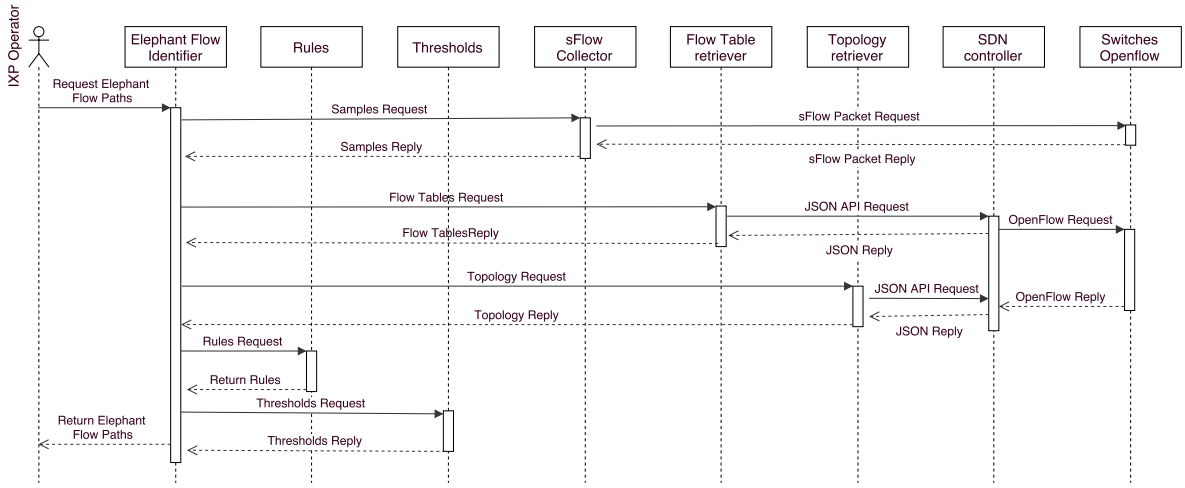
4.4 Identificação de Fluxos Elefantes

O *framework* Django foi utilizado como base para o desenvolvimento do SDEFIX. A estrutura de classes do *framework* Django é comumente definida por um conjunto de dois arquivos e uma pasta. O *models.py* é responsável pelo armazenamento do modelo de dados do módulo, comunicando com o mongoDB. O arquivo *view.py* é responsável pelo controle e encaminha-

mento das ações do módulo, que por fim, são enviados para um dos *templates* disponíveis na pasta *template* de cada módulo que, por sua vez, apresenta os dados na tela.

O principal módulo da identificação dos fluxos é o *Elephant Flow Identifier* que implementa a lógica por trás das interações entre os limiares, regras de identificação definidas pelo operador e dados recebidos tanto do controlador SDN quanto do sFlow. Realizado em etapas, primeiro o módulo identifica todos os fluxos com os limiares definidos pelo sistema em cada *switch* da rede, após une os fluxos iguais, aumentando seus contadores de tempo e taxa de transferência. Posteriormente, os fluxos são testados sobre as regras de identificação. Por último, os fluxos são validados com as tabelas de fluxos obtidas junto aos *switches*. Ao relacionar o fluxo com uma das regras existentes em cada *switch*, o sistema adiciona o *switch* em questão à lista de *switches* percorridos pelo fluxo elefante. Retornando todos os fluxos ao operador na forma de um *snapshot*.

Figura 4.3: Diagrama de Sequência: o sistema de identificação



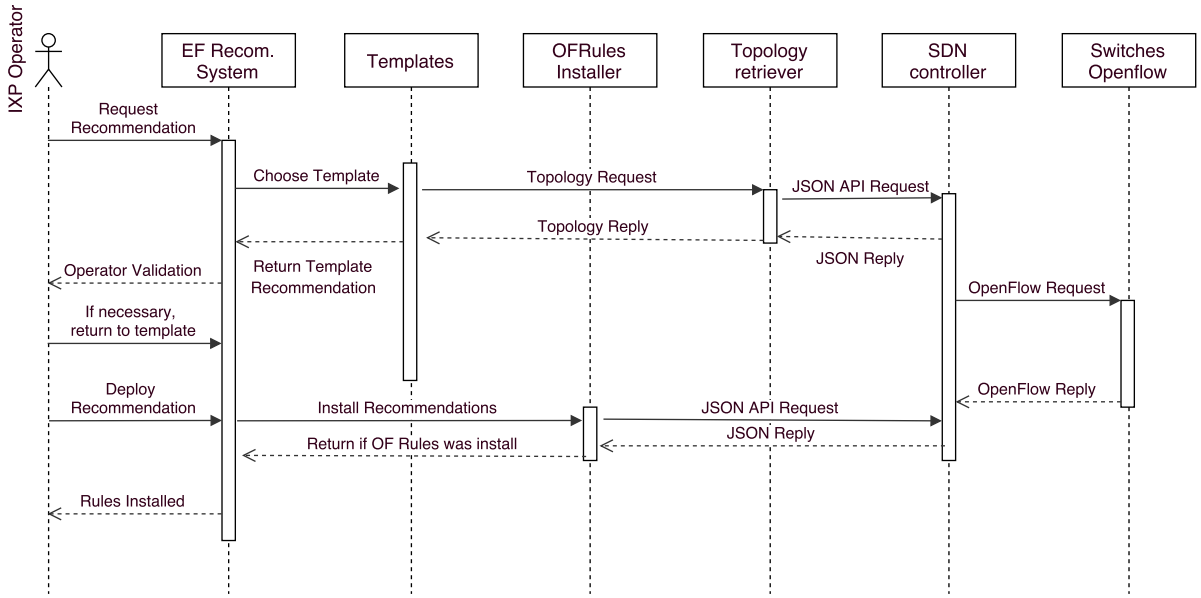
Fonte: do Autor (2016).

O diagrama de sequência apresentado na Figura 4.3, mostra as relações entre as diversas classes no sistema que alimentam o *Elephant Flow Identifier*. Após receber a requisição de identificação dos fluxos elefante, esta classe deve solicitar as amostras retiradas da rede pelo *sFlow Collector*, que comunica com os switches através do protocolo sFlow. Além disso é requisitado a topologia e as tabelas de fluxos dos módulos *Flow Table Retriever* e *Topology Retriever* que obtêm seus dados juntos ao controlador SDN. Após, estes dados são relacionados as regras de identificação definidas pelo operador no módulo *Rules* e os limiares definidos no *Thresholds*. Por fim, é retornado ao operador os caminhos percorridos pelos fluxos elefante na rede.

4.5 Implementação do Sistema de Recomendação

O sistema de recomendação trabalha de forma independente do sistema de identificação, embora utilize como entrada um *snapshot* dos fluxos elefante. Desta forma, não existe uma obrigatoriedade em sua utilização a cada identificação realizada. Além disso, embora o comportamento padrão seja realizar recomendações do último *snapshot* gerado, isto poderia ser modificado de forma a recuperar qualquer *snapshot* salvo no sistema.

Figura 4.4: Diagrama de Sequência: o sistema de recomendação



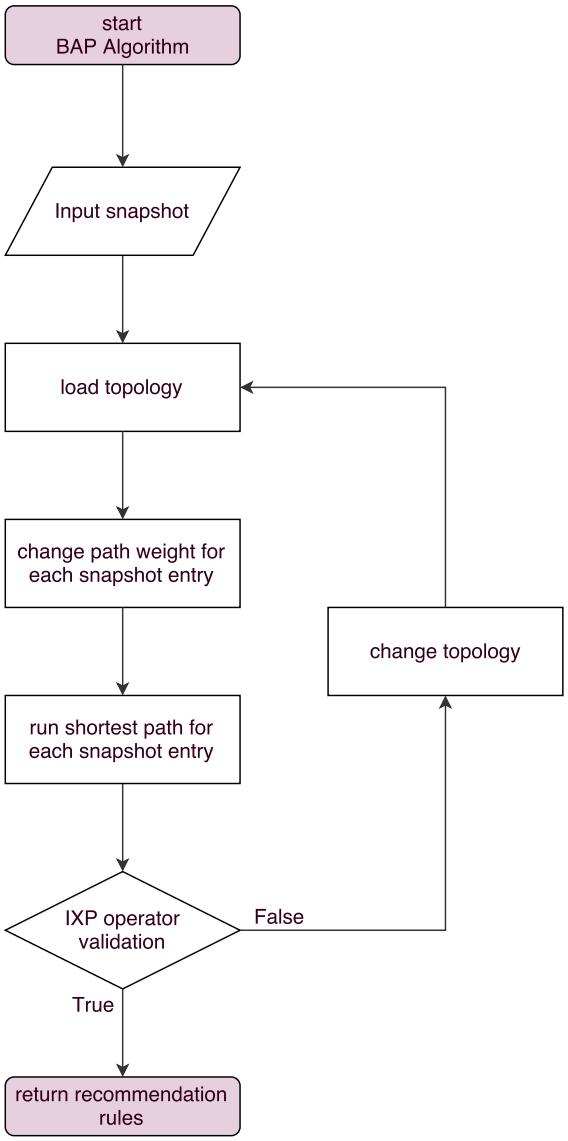
Fonte: do Autor (2016).

O diagrama apresentado na Figura 4.4, apresenta a sequência de funcionamento do módulo do sistema de recomendação para o controle e mitigação dos fluxos elefante. Após o operador realizar a solicitação de recomendação, enviando os resultados obtidos pela identificação dos fluxos elefante, este direciona os dados ao módulo *Templates* junto ao *template* a ser utilizado. Este módulo, pode buscar informações do sistema, como a topologia junto ao *Topology Retriever*. Após a execução, o *template* retorna uma recomendação que é então avaliada pelo operador e, se necessário, modificada. Após esta etapa, a recomendação finalizada é enviada para o *OFRules Installer*, que encaminha as regras para o controlador SDN através de uma API JSON. Por fim, é retornado ao operador se as regras foram instaladas com sucesso ou não.

4.5.1 Implementação dos Templates

Para validar o funcionamento do sistema de recomendação do SDEFIX foi criado um *template* que realiza a escolha da segunda melhor rota para um fluxo elefante de forma a realizar duas funções: (i) diminuir o impacto sobre os fluxos menores, liberando os enlaces padrões para

Figura 4.5: Fluxograma do algoritmo BAP



Fonte: do Autor (2016).

estes, (ii) entregar uma rota alternativa para otimizar o uso de todos os enlaces do PTT. Desta forma, o fluxograma do algoritmo apresentado na Figura 4.5, é uma representação resumida do algoritmo BAP implementado no sistema.

Podemos dividir a implementação apresentada em 3 partes: (i) O carregamento dos objetos que podem ser utilizados durante a execução do algoritmo, como a topologia da rede e o *snapshot* contendo os fluxos elefante a serem mitigados; (ii) A execução da técnica a ser utilizada, à qual corresponde a alteração dos custos dos caminhos conforme os fluxos e o cálculo do segundo melhor caminho, tanto para a ida como para a volta; e por fim (iii) a validação das recomendações geradas pelo operador do PTT, que se necessário, pode alterar a topologia, rodando novamente o algoritmo.

No próximo capítulo, será avaliada a performance tanto da identificação dos fluxos elefante em relação ao seu tempo e utilização do servidor, quanto do sistema de recomendação proposto neste capítulo.

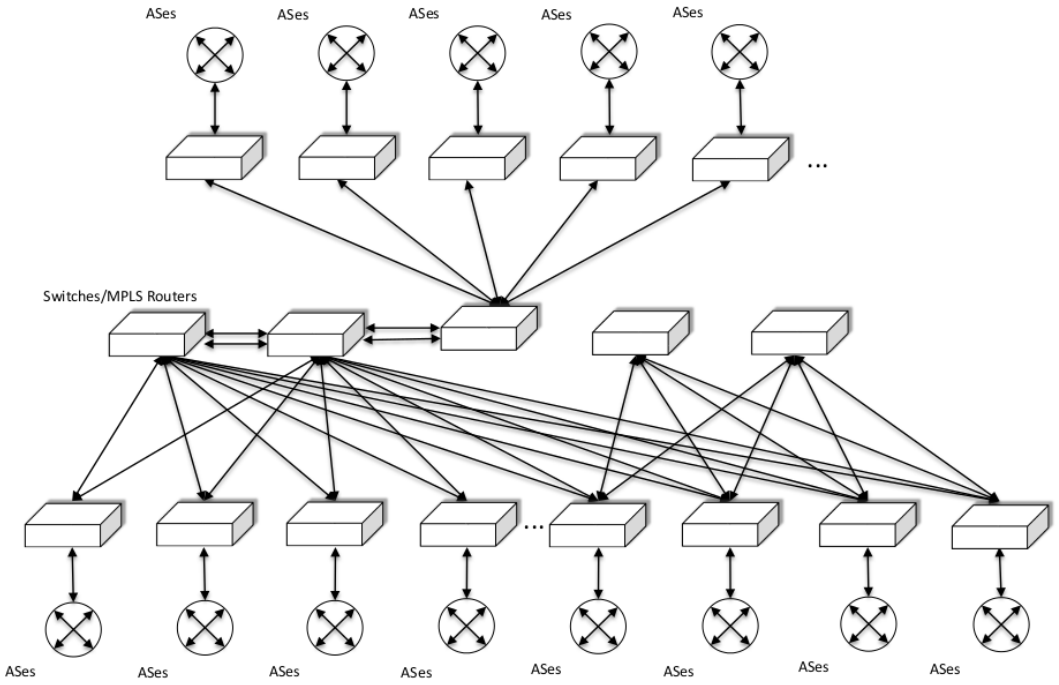
5 AVALIAÇÃO EXPERIMENTAL

5.1 Avaliação da Identificação dos Fluxos Elefantes

Nesta seção, avaliar-se-á a performance do SDEFIX na identificação de fluxos elefante. Iniciar-se-á descrevendo-se o cenário experimental e a metodologia utilizada nos testes. Após, apresentar-se-ão e se discutirão os resultados obtidos até o momento.

5.1.1 Cenário

Figura 5.1: Uma infraestrutura de PTT baseada no AMS-IX



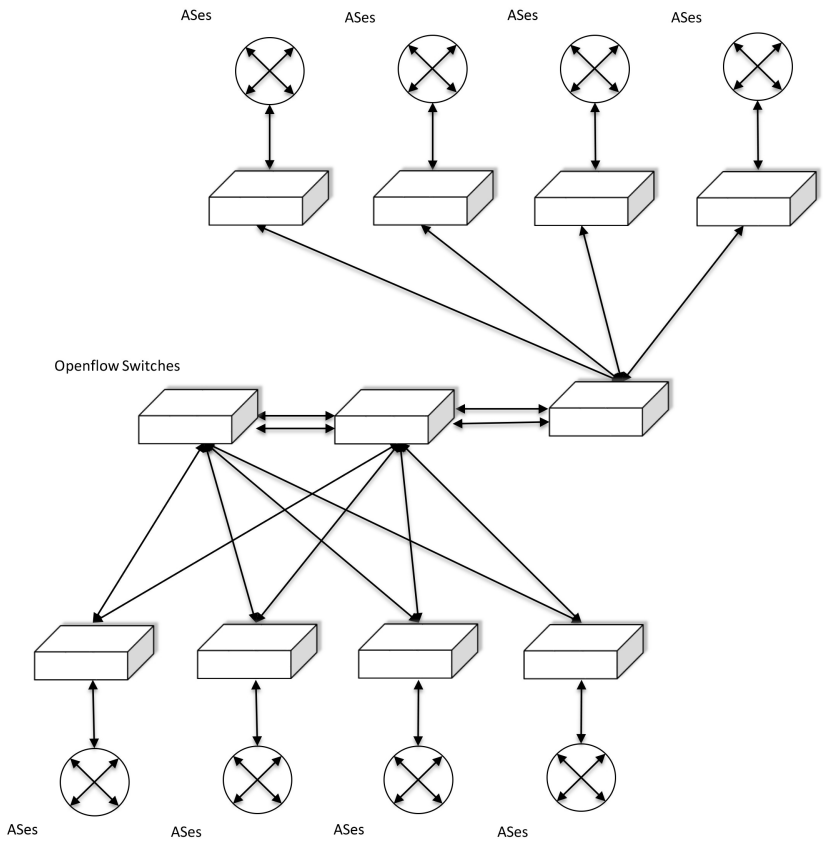
Fonte: do Autor (2016).

Conforme visto acima, a Figura 5.1 retrata uma infraestrutura baseada no AMS-IX (AMS-IX, 2015), além disso, estruturas similares podem ser encontradas em outros PTTs. Um PTT é tipicamente composto por um número de *switches* de rede camada 2 e/ou roteadores MPLS, que permitem os ASes a trocarem tráfego. Cada AS participante deve conectar um roteador de borda em um dos *switches*/roteadores pertencentes ao PTT e estabelecer uma sessão BGP de forma a realizar um *peering* com os outros ASes. (AGER et al., 2012).

O cenário experimental criado é composto por três máquinas virtuais (VMs, do inglês *Virtual Machine*) rodando numa máquina hospedeira com um Core i7 4790 com 16 GB de RAM através do VirtualBox. Cada VM possui 2GB de RAM e roda o sistema operacional Debian 8. A primeira máquina virtual roda o emulador Mininet (MININET, 2015) e contém a topologia

usada nos testes. A topologia apresentada na Figura 5.2 é uma versão simplificada da infraestrutura do AMS-IX (AMS-IX, 2015) utilizando *switches* OpenFlow (originalmente o AMS-IX utiliza MPLS routers). Na topologia, substitui-se os *core routers* do AMS-IX por 3 *switches* OpenFlow e simulamos as conexões com o PTT de 8 *switches* de borda pertencente a ASes ao invés de roteadores PE utilizados na AMS-IX. Todos os *switches* possuem como base o OpenVSwitch (OPENVSWITCH, 2015). A segunda VM hospeda o controlador SDN Ryu, que foi escolhido por permitir o desenvolvimento rápido de aplicações de gerência de rede. SDEFIX e seus componentes principais (*Elephant flow identifier, sFlow collector, Flow table retriever, Topology retriever*, a Interface Administrativa, e todos os bancos de dados MongoDB) rodam em uma terceira máquina virtual. De forma a manter a simplicidade da simulação, não foi detalhada a topologia interna de cada AS. Além disso, considerou-se que dentro de cada AS existem 32 clientes gerando tráfego.

Figura 5.2: Topologia utilizada nos experimentos



Fonte: do Autor (2016).

Nos experimentos, o *sFlow Collector* recebe as amostras de fluxos de cada *switches* a cada 10 segundos, que é o intervalo convencionalmente utilizado nas configurações padrões do sFlow. A taxa de amostragem de pacotes variou de 1 a cada 128 pacotes até 1 a cada 4096 pacotes [1]. Os limiares de tamanho e a duração usados para caracterizar os fluxos elefante foram

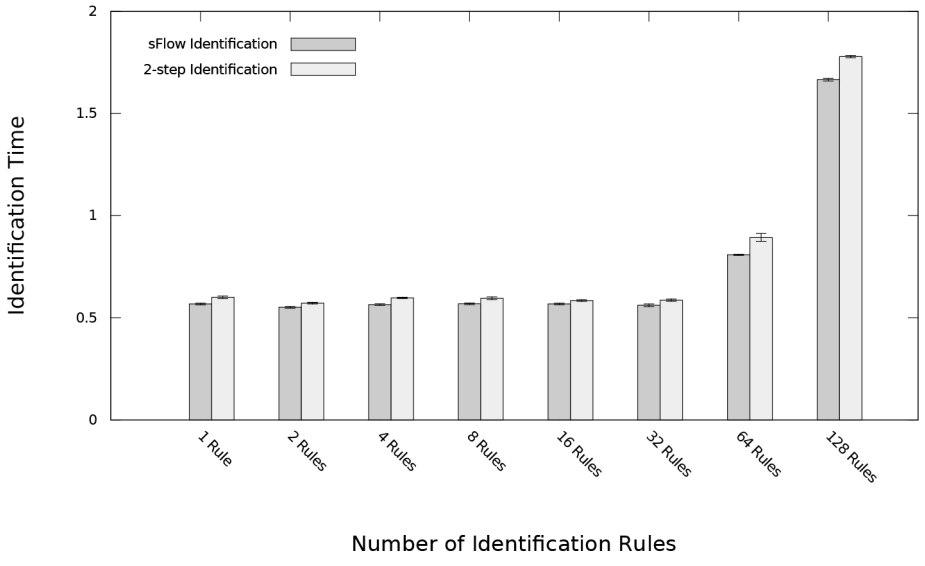
[1] A partir deste ponto utilizaremos a anotação $1/x$ para representar a taxa de amostragem de pacotes de 1 em

100MBps e 10 segundos, respectivamente. Cada cliente estabeleceu 10 conexões com outros destinos da rede, 9 conexões foram usadas para caracterizar os fluxos pequenos com transferência entre 4Mbps e 10Mbps. O restante da conexões representam um fluxo elefante com taxa de transferência de 100Mbps. Isso significa que existiram na rede um total de 128 fluxos elefante na rede PTT simulada. O número total de fluxos, entre pequenos e elefantes, foram de 1280.

Nosso interesse é analisar o desempenho do SDEFIX nos seguintes termos: (1) *tempo de identificação*, (2) *número de fluxos elefante identificados* e, (3) *utilização de recursos*. O tempo de identificação é o tempo que o sistema demora para responder uma requisição realizada pelo operador do PTT na identificação os fluxos elefante ativos na rede em um determinado momento. O número de fluxos elefante identificados corresponde àqueles que foram corretamente identificados pelo sistema em um determinado número de taxa de amostragem de pacotes. O tráfego de amostragem é o total de tráfego transferido resultante do processo de amostragem. Recursos utilizados são definidos nos termos do pico de uso de CPU, uso de memória e tamanho do banco de dados. O uso de CPU e memória refletem o total de recursos consumidos pelo SDEFIX em relação ao hospedeiro. O tamanho do banco de dados é a soma de todas as coleções do banco de dados MongoDB usadas pelo SDEFIX. Cada teste foi repetido 30 vezes com um intervalo de confiança de 95%. Na próxima subseção, apresentar-se-ão os principais resultados de nossa avaliação.

5.1.2 Resultados

Figura 5.3: Tempo de identificação pelo número de regras de identificação



Fonte: do Autor (2016).

cada *x* pacotes.

A Figura 5.3 relata o tempo de identificação em relação ao número de regras (que aumentam em cada conjunto de experimentos) utilizadas para identificar os fluxos elefante (regras de identificação). Neste caso, o taxa de amostragem de pacotes foi fixada em 1/128. Aqui, contrapomos duas abordagens para a identificação de fluxos elefante. A primeira, referenciada como *sFlow identification*, é aquela quando o fluxo elefante e seu caminho correspondente é identificado a partir de amostras do fluxo em todos os *switches* do caminho. A segunda abordagem, chamada *2-step identification*, é quando um fluxo elefante é identificado com a coleta de amostras em pelo menos um *switch*. Então o caminho correspondente pode ser calculado pela ocorrência do fluxo elefante como entrada nas tabelas de fluxos coletadas pelo sistema de acordo a estratégia descrita no Algoritmo 1. Isso é, se o fluxo elefante for encontrado na tabela de fluxo de um *switch*, este *switch* é incluído no caminho cruzado pelo fluxo elefante.

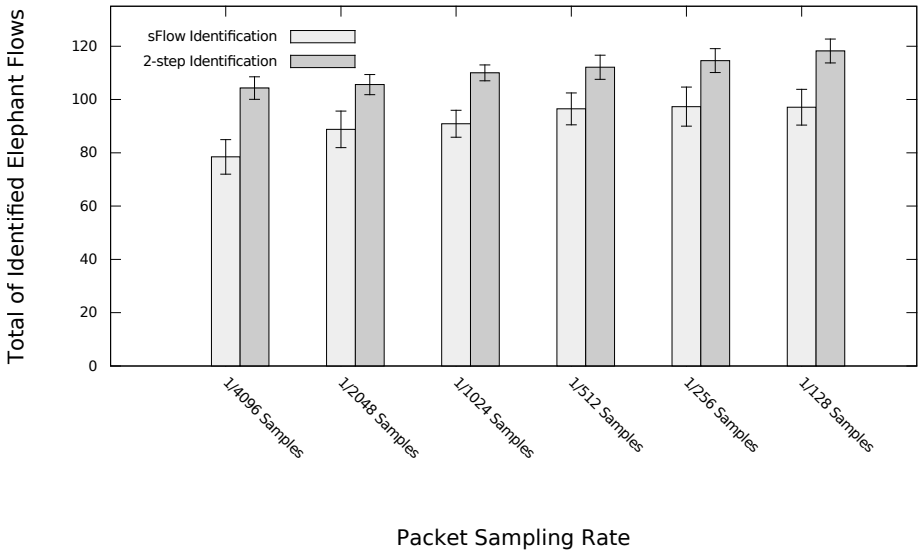
É possível observar na Figura 5.3 que a abordagem *2-step identification* realiza a identificação, em média, 6% mais lenta que a *sFlow identification* para a taxa de amostragem de 1/128. O tempo de identificação se mantêm estável próximo aos 0.6 segundos, caso o operador do PTT consulte de 1 até 32 regras de identificação. Para 64 e 128 regras de identificação, o tempo ficou em 0.9 e 1.8 segundos, respectivamente, o que acreditamos que seja aceitável em um cenário factível. Além disso, o tempo de identificação aumenta numa escala menor comparado ao número de regras de identificação. O tempo de identificação é 3 vezes maior (0.6 para 1.8) para um número de regras de identificação 100 vezes maiores (1 para 128). Isto representa uma diferença de duas ordens de magnitude.

Também medimos o tempo de identificação para diferentes taxas de amostragem de pacotes (1/4096, 1/2048, 1/512, 1/256, e 1/128 pacotes), enquanto o número de regras de identificação se mantêm fixo em 128. Observamos que o tempo de identificação não varia significantemente com a taxa de amostragem, e os resultados apresentados são basicamente os mesmos da Figura 5.3, a qual representa o pior caso, isto é, taxa de amostragem de pacotes de 1/128.

Por sua vez, a Figura 5.4 apresenta o número médio de fluxos elefante identificados quando a taxa de amostragem de pacotes varia de 1/4096 para 1/128. É observável que na Figura 5.4 o número de fluxos elefante aumenta linearmente de acordo com a taxa de amostragem de pacotes em ambas abordagens de identificação. Isto ocorre porque a chance de coletar uma amostra do fluxo que pertence ao fluxo elefante aumenta quando o número total de amostra é maior. A *2-step identification* possui uma performance em média 20% melhor que a solução pura *sFlow identification*. A diferença entre as duas abordagens pode ser explicada pelo fato de que na *sFlow identification* existem alguns fluxos elefante que não foram identificados em um ou mais *switches* da rede do PTT, tendo em vista que não foi coletada nenhuma amostra dos fluxos elefante nestes *switches* e, por consequência, não é possível determinar o caminho utilizado pelo fluxo com precisão, enquanto que na *2-step identification*, o sistema é capaz de determinar com precisão o caminho correspondente comparando a amostra do fluxo com as entradas de fluxo de cada *switch* da rede do PTT.

A Tabela 5.1 apresenta a quantidade de tráfego de amostragem para diferentes taxas de

Figura 5.4: Número de fluxos elefante identificado em relação a taxa de amostragem de pacotes



Fonte: do Autor (2016).

Tabela 5.1: Tráfego de amostragem (Mbytes)

Sampling rate	Traffic
1/128	224.4
1/256	101.2
1/512	61.6
1/1024	37.4
1/2048	19.1
1/4096	11

Fonte: do Autor (2016).

amostragem de pacotes. Há uma significativa diferença (mais de 20 vezes) em relação ao tráfego gerado, quando a taxa de amostragem é de 1/128 (224 MB), comparado com a taxa de amostragem de 1/4096, a qual gera apenas 11 MB de tráfego de rede.

Tabela 5.2: Utilização de recursos do servidor

Metric	1/128 (128R)	1/128 (1R)	1/4096 (128R)	1/4096 (1R)
CPU peak	10.2%	4.3%	8.6%	2.8%
Memory usage (MB)	155	155	155	155
Database size (MB)	215.4	215.1	38.8	38.5

Fonte: do Autor (2016).

A Tabela 5.2 resume os resultados em relação aos recursos utilizados variando o número

de regras de identificação (1 e 128) e a taxa de amostragem (1/128 e 1/4096). O uso de CPU foi primariamente dependente do número de regras de identificação e impacta no tempo de identificação conforme mostrado na Figura 5.3. A taxa de amostragem possui pouca influência no uso da CPU e o uso de memória ficou estável em torno de 155 MB. Como esperado, a taxa de amostragem está diretamente relacionada com o tamanho do banco de dados (*i.e.Flow Sample database*) usado pelo SDEFIX para armazenar as amostras coletadas.

5.1.3 Resumo dos Experimentos

Em conclusão, o tempo de identificação depende principalmente do número de regras de identificação definidas pelo operador do PTT, onde no pior caso, o sistema necessita comparar as amostras de fluxos contra todas as regras de identificação. Nossa abordagem de duas etapas para a identificação de fluxos elefante é capaz de identificar uma média de 20% mais fluxos elefante comparada a uma abordagem tradicional que utiliza apenas amostras para a identificação, isto demostra os benefícios por utilizar uma solução para a identificação de fluxos elefante em um ambiente SDN.

Como esperado, existe uma relação entre a precisão na identificação dos fluxos elefante e o total de recursos que foram utilizados para permitir esta identificação. O total do tráfego gerado pela taxa de amostragem de 1/128 é mais que 20 vezes maior do que o tráfego gerado pela taxa de 1/4096 pacotes. Entretanto, a precisão obtida pela taxa 1/128 é apenas 13% melhor que comparada a taxa de 1/4096 (Figura 5.4). Os recursos utilizados na CPU e na memória indicam que o SDEFIX é leve e pode ser implementado em *hardware* genérico. O principal gargalo do sistema de identificação proposto nesta dissertação é o tamanho do banco de dados, o qual depende principalmente da taxa de amostragem. Além disso, o operador do PTT deve avaliar a importância, tanto da acurácia na identificação, quanto do impacto gerado pelo aumento do tráfego na rede, e defini-las de acordo com a sua estrutura.

5.2 Avaliação do Sistema de Recomendação

Nesta seção, avaliar-se-á a performance do sistema de recomendação integrado ao SDEFIX para a mitigação do impacto dos fluxos elefante na rede do PTT. Iniciar-se-á com a descrição do cenário experimental e a metodologia utilizada nos testes. Após isto, serão apresentados e discutidos os principais resultados alcançados até o momento.

5.2.1 Cenário

Assim como na avaliação da identificação dos fluxos elefante, utilizamos como cenário de avaliação uma versão simplificada da infraestrutura do AMS-IX (AMS-IX, 2015) trocando os roteadores MPLS originais por *switches* SDN conforme mostrado na Figura 5.2. Emulamos a

topologia utilizando o emulador Mininet rodando em uma máquina virtual (VM) com 2Gb de RAM. O OpenVSwitch foi utilizado como *switch* tendo em vista seu suporte às últimas implementações do protocolo OpenFlow. O controlador SDN Ryu foi hospedado em uma segunda VM, enquanto o SDEFIX e o sistema de recomendação rodam em uma terceira VM. As VMs estão hospedadas em um servidor Core i7 4790 com 16GB de RAM através do VirtualBox. Em nosso cenário são avaliadas 8 ASes com 32 hosts gerando tráfego entre elas.

Avaliamos o sistema de recomendação integrado ao SDEFIX no que diz respeito a: (1) *tempo de mitigação* e; (2) *pico de tráfego*. O tempo de mitigação é o tempo total tomado pelo sistema de recomendação para gerar e aplicar uma recomendação na rede do PTT em relação aos fluxos elefante. O pico de tráfego reflete a mais alta carga em cada *switch* da rede do PTT. Cada teste foi repetido 30 vezes com um intervalo de confiança de 95%. Nas próximas subseções, apresentar-se-ão os principais resultados da nossa avaliação.

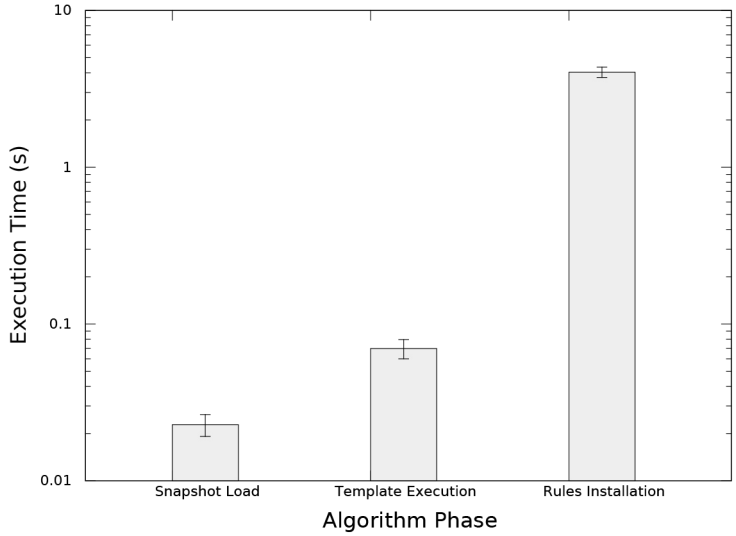
5.2.2 Resultados

Para melhor ilustrar os resultados obtidos em relação ao tempo de mitigação, fracionamos o processo de mitigação em três fases distintas. A primeira fase corresponde ao tempo de carregamento do *snapshot*, ou *Snapshot Load Phase*, onde um *snapshot* com os caminhos dos fluxos elefante identificados é carregado pelo sistema. Na fase de execução do *template*, ou *Template Execution Phase*, o *template* escolhido (BAP) é aplicado sobre o *snapshot* gerando as recomendações de regras de controle/mitigação dos fluxos elefante. Depois da validação das recomendações geradas pelo *template*, as regras são enviadas para o controlador SDN a fim de que sejam instaladas. Neste experimento, o número de fluxos elefante foi de 128 e o número total de fluxos foi 1408. Fluxos elefante possuíram um tráfego médio de 10Mb/s, enquanto os fluxos pequenos (camundongos) tiveram um tráfego variando entre 400Kb/s e 1Mb/s. A Figura 5.5 demostra o tempo de execução de cada fase separadamente.

É possível observar na Figura 5.5 que o tempo de carregamento do *snapshot* é muito menor (20 milisegundos) do que o de execução do *template* e a instalação das regras. O tempo de execução do *template* é altamente dependente da estratégia utilizada, e pode ter seu resultado alterado dependendo do método de funcionamento do *template*. Em nosso caso, o *template* BAP levou menos de 70 milisegundos para gerar a recomendação, o que pode ser considerado baixo pelo tamanho da rede do PTT. A instalação das regras ocupa o maior tempo nesta abordagem de mitigação, visto que é dependente do número de regras que devem ser instaladas. Como nosso cenário é denso no que diz respeito ao número de fluxos elefante e fluxos camundongos, era esperado que o sistema de recomendação gerasse um alto número de regras para serem instaladas pelo controlador SDN, o que ocorreu. Em nossos experimentos, 650 regras foram instaladas em cerca de 4 segundos, o que faz com que cada regra tenha sido instalada em cerca de 6 milisegundos em média.

A Figura 5.6 apresenta o pico de tráfego observado em cada *switch* da rede do PTT após a

Figura 5.5: Tempo de Mitigação da solução



Fonte: do Autor (2016).

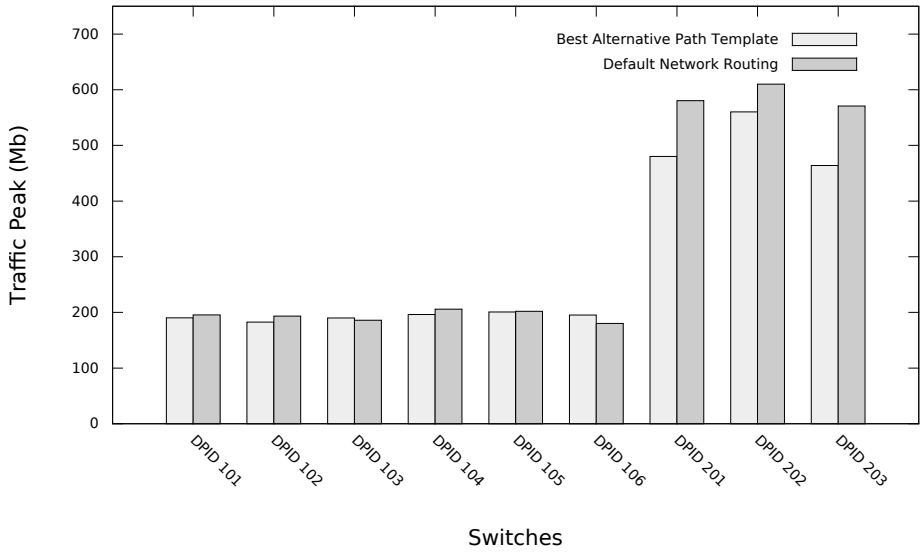
aplicação do *template* BAP comparado ao pico de tráfego da rede sem nenhuma modificação. A rede sem nenhuma modificação consiste em caminhos computados pelos mecanismos padrões de roteamento, que tentam encontrar os menores caminhos disponíveis para cada fluxo. Para uma melhor visualização do impacto do sistema de recomendação na mitigação dos fluxos elefante, nós reduzimos o número de fluxos elefante para 16 e o número total de fluxos (pequenos e elefantes) para 176. Neste experimento, os fluxos elefante possuíram um tráfego médio de 10 Mb/s enquanto os fluxos pequenos tiveram um tráfego variando entre 400Kb/s e 1 Mb/s. Os *switches* foram identificados pelos nomes DPID N, sendo N o número de cada *switch*. DPID 201, DPID 202 e DPID 203 foram os *switches* que compuseram o núcleo da rede, enquanto que os *switches* DPID 101 até DPID 106 foram os *border switches* da rede do PTT.

O resultado do experimento demonstrou que os efeitos da aplicação do *template* BAP puderam ser melhor visualizado nos *core switches*, pois estes lidam com a maior parte do tráfego da rede do PTT. A redução no pico do tráfego deste *switches* foi em média de 17%, 8%, e 19% para os *switches* DPID 201, DPID 202, DPID 203, respectivamente. A diminuição dos picos de tráfego não é maior nos *core switches*, principalmente por causa do pequeno número de enlaces alternativos de conexão entre eles. Não obstante, o sistema de recomendação do SDEFIX foi capaz de melhorar a utilização dos recursos, mesmo em um cenário limitado.

5.2.3 Resumo dos Experimentos

Em suma, o desempenho do sistema de recomendação do SDEFIX é profundamente relacionado com a forma de mitigação/controle definida no *template* utilizado pelo operador do PTT para lidar com os fluxos elefante. Podemos dividir o tempo total gasto na execução do sistema

Figura 5.6: Pico de tráfego nos *switches*



Fonte: do Autor (2016).

de recomendação em duas fases distintas: (i) o tempo de execução do *template*, que é vinculado ao *template* definido pelo operador; (ii) o tempo de carregamento do *snapshot* e o tempo de instalação individual de cada regra, que não são vinculados ao *template* definido. Verificou-se ainda que o sistema de recomendação impôs apenas uma pequena sobrecarga sobre a execução do processo de mitigação. Além disso, o tempo total de execução levou menos do que 5 segundos, mesmo em um cenário denso, com um número alto de fluxos elefante, demonstrando a escalabilidade do sistema.

Como esperado, existe um limitador entre o impacto da mitigação quando se opta pelo *template* BAP e a topologia física interna do PTT. Se a rede do PTT não oferece muitos caminhos alternativos, o efeito do *template* BAP será reduzido, o que pode ser visto nos resultados. Entretanto, diferentes *templates* projetados para alcançar outros objetivos, como balanceamento de carga ou redução de consumo de energia, por exemplo, podem alcançar uma perfomance melhor no mesmo cenário. Portanto, a topologia física do PTT é o ponto mais influente na escolha dos *templates*, e testes devem ser realizados para obter as melhores soluções para cada cenário.

6 CONCLUSÃO

Nesta dissertação foi apresentada e proposta a utilização do SDEFIX, que consiste num sistema de monitoramento para PTTs baseado em SDN, o qual permite identificar os fluxos elefante e os caminhos de rede a eles associados, além de possuir um sistema de recomendação para a gerência destes fluxos elefante na rede. O SDEFIX utiliza uma abordagem baseada em amostragem (sFlow), e aproveita a abstração das tabela de fluxos do protocolo OpenFlow para aprimorar sua precisão na identificação dos fluxos. Diferente das outras soluções para a identificação de fluxos baseada em amostragem, no SDEFIX, não é necessário que cada *switch* do caminho cruzado pelo fluxo elefante envie uma amostra do fluxo de forma a calcular o caminho deste fluxo. Ao invés disso, no SDEFIX, apenas uma amostra é necessária para permitir a identificação do fluxo elefante e seu caminho associado, não importando aonde ela tenha sido capturada.

Em conjunto com o sistema de identificação, o sistema de recomendação utiliza os *snapshots* dos caminhos dos fluxos elefante gerados pelo *framework* para auxiliar o operador do PTT na mitigação e/ou controle destes fluxos. A partir da escolha de um *template*, que pode ser adicionado ou modificado pelo operador do PTT, o sistema de recomendação pode auxiliar no encaminhamento dos fluxos elefante para rotas menos congestionadas, na diminuição do consumo de energia, ou mesmo realizar QoS nas conexões. Além disso, diferentemente de outras soluções, o sistema entrega uma recomendação que pode ser validada pelo operador antes de ser implementada na rede, garantindo a confiabilidade e a conformidade destas alterações.

6.1 Principais Contribuições e Resultados Obtidos

Pode-se destacar que os objetivos desta dissertação foram alcançados, pois o SDEFIX permite ao operador do PTT um controle em grão fino dos fluxos, principalmente na identificação e no tratamento dos fluxos elefante e na garantia de uma melhor utilização da rede. Isto pode ser verificado pelas configurações que devem ser realizadas pelo operador, como a definição das regras de identificação dos fluxos, os limiares a serem utilizados e a validação da recomendação entregue pela ferramenta. Além disso, através do monitoramento, e pela criação de *snapshots* dos fluxos da rede, o SDEFIX também pode auxiliar na gestão do PTT, relatando possíveis gargalos e enlaces que ficam constantemente congestionados de acordo com o tipo de fluxo passado por eles.

Em relação à identificação dos fluxos elefante, os resultados da avaliação mostraram que o tempo de identificação depende primariamente do número de regras de identificação definidas pelo operador do PTT e não é significativamente influenciado pela taxa de amostragem dos pacotes. A taxa de amostragem dos pacotes afeta a acurácia da identificação, isto é, o número de fluxos elefante identificados aumenta com a taxa de amostragem. Entretanto, aprimorar a precisão da identificação dos fluxos elefante, utilizando uma taxa alta de amostragem de pacotes,

resulta em um aumento da taxa de transferência de amostragem entre os *switches* e o *sFlow Collector*, o que pode afetar a performance do sistema como um todo. Apesar disso, o operador do PTT pode decidir entre as duas possibilidades de configurações e escolher a melhor relação entre taxa de amostra e total de dados transferidos para o seu cenário. Em cenários com recursos limitados pode ser importante não gerar uma quantidade excessiva de tráfego de amostragem, já que isso, indesejadamente pode afetar outros serviços. Por outro lado, se a precisão da identificação é crucial para o cenário, um aumento na taxa de amostragem se faz necessário, ainda que as consequências resultem em um aumento do tráfego gerado.

Avaliando-se o sistema de recomendação, conclui-se que a utilização de *templates* definidos pelo operador do PTT, que sugerem configurações alternativas para a rede, auxiliam a mitigação dos fluxos elefante. Além disso, ao utilizar-se o algoritmo BAP como *template*, o sistema alcançou um tempo de aplicação relativamente baixo, ainda que com grande número de fluxos elefante presentes e com a instalação das regras nos *switches* como gargalo da solução. Isso comprova que, mesmo um algoritmo simples como o BAP, pode trazer resultados significantes para o gerenciamento dos fluxos, reduzindo seu impacto na infraestrutura do PTT.

6.2 Considerações Finais e Trabalhos Futuros

Por conclusão, um sistema de monitoramento e recomendação para a identificação, mitigação e controle de fluxos elefante em um PTT baseado em redes SDN é apenas o primeiro passo para um efetivo gerenciamento dos fluxos. O estudo e a definição de diferentes *templates*, como por exemplo, *templates* focados em QoS, ou novas estratégias para roteamento em redes críticas como as redes de PTT, são necessários para disponibilizar ao operador do PTT a estratégia adequada visando o encaminhamento dos fluxos elefante e avaliar a relação destes com os fluxos pequenos na rede.

Além disso, em trabalhos futuros pretende-se verificar a viabilidade da utilização de algoritmos de aprendizagem de máquina como forma de auxiliar na identificação dos fluxos em adição às regras definidas pelo operador do PTT. Outro estudo que se pode realizar é a configuração e incorporação de outros *templates* ao sistema de recomendação, a fim de aproveitar as novas funcionalidades presentes nas últimas versões do OpenFlow, como por exemplo fila de QoS.

Por fim, pode-se concluir que um sistema que auxilie o operador do PTT na identificação e gerência de fluxos elefante é viável e factível. Portanto, a partir desta dissertação espera-se que as soluções de gerenciamento de PTTs evoluam de forma a contemplar as funcionalidades fornecidas pelo SDEFIX. Espera-se ainda que a evolução dos PTTs para redes definidas por software aconteça de forma natural e estruturada em novas funcionalidades que facilitem a administração deste cenário pelos seus operadores.

REFERÊNCIAS

AF-IX. **African Internet Exchange Association**. http://www.af-ix.net: [s.n.], 2015.

AFAQ, M.; REHMAN, S.; SONG, W.-C. Visualization of elephant flows and qos provisioning in sdn-based networks. In: **Network Operations and Management Symposium (APNOMS), 2015 17th Asia-Pacific, Proceedings...** Busan, Korea: [s.n.], 2015. p. 444–447.

AFAQ, M.; REHMAN, S. U.; SONG, W.-C. A framework for classification and visualization of elephant flows in sdn-based networks. In: **International Conference on Communications, management, and Information technology (ICCMIT'2015), Proceedings...** [S.l.: s.n.], 2015. v. 65, p. 672–681.

AFEK, Y. et al. Sampling and large flow detection in sdn. **SIGCOMM Comput. Commun. Rev.**, ACM, New York, NY, USA, v. 45, n. 5, p. 345–346, aug. 2015. ISSN 0146-4833. Available from Internet: <http://doi.acm.org/10.1145/2829988.2790009>.

AGER, B. et al. Anatomy of a large european ixp. In: **ACM SIGCOMM 2012 Conference on Applications, Technologies, Architectures, and Protocols for Computer Communication, Proceedings...** New York, NY, USA: ACM, 2012. (SIGCOMM '12), p. 163–174.

AHMAD, M.; GUHA, R. A tale of nine internet exchange points: Studying path latencies through major regional ixps. In: **Local Computer Networks (LCN), 2012 IEEE 37th Conference on, Proceedings...** [S.l.: s.n.], 2012. p. 618–625. ISSN 0742-1303.

AMS-IX. **Amsterdam Internet Exchange Infrastructure**. https://ams-ix.net/technical/ams-ix-infrastructure: [s.n.], 2015.

AMS-IX.NET. **Amsterdam Internet Exchange Home**. https://ams-ix.net/, 2016. Available from Internet: <https://ams-ix.net/>.

APIX. **Asia Pacific Internet Exchange Association**. http://apix.asia: [s.n.], 2015.

AUGUSTIN, B.; KRISHNAMURTHY, B.; WILLINGER, W. Ixps: Mapped? In: **9th ACM SIGCOMM Conference on Internet Measurement Conference, Proceedings...** New York, NY, USA: ACM, 2009. (IMC '09), p. 336–349.

BENNESBY, R. et al. An inter-as routing component for software-defined networks. In: **Network Operations and Management Symposium (NOMS), 2012 IEEE, Proceedings...** Muai, Hawaii, USA: IEEE, 2012. p. 138–145.

BI, C. et al. On precision and scalability of elephant flow detection in data center with sdn. In: **Globecom Workshops (GC Wkshps), 2013 IEEE, Proceedings...** Atlanta, USA: IEEE, 2013. p. 1227–1232.

BOSTOCK, M. **D3.js - Data-Driven Documents**. 2016. Available from Internet: <https://d3js.org/>.

BRITO, S. B. et al. Anatomy of public internet exchange points ecosystem in brazil. In: **Computer Networks and Distributed Systems (SBRC), 2015 XXXIII Brazilian Symposium on, Proceedings...** Vitoria, Brazil: SBC, 2015. p. 110–119.

CASADO, M.; PETTIT, J. **Of Mice and Elephants**. http://networkheresy.com/2013/11/01/of-mice-and-elephants/: [s.n.], 2013.

CHATZIS, N. et al. On the benefits of using a large ixp as an internet vantage point. In: **2013 Conference on Internet Measurement Conference (IMC), Proceedings...** Barcelona, Spain: ACM, 2013. (IMC '13), p. 333–346.

CHATZIS, N.; SMARAGDAKIS, G.; FELDMANN, A. On the importance of internet exchange points for today's internet ecosystem. **CoRR**, abs/1307.5264, 2013. Available from Internet: <http://arxiv.org/abs/1307.5264>.

CHATZIS, N. et al. There is more to ixps than meets the eye. **SIGCOMM Comput. Commun. Rev.**, ACM, New York, NY, USA, v. 43, n. 5, p. 19–28, nov. 2013. ISSN 0146-4833. Available from Internet: <http://doi.acm.org/10.1145/2541468.2541473>.

CHATZIS, N. et al. Quo vadis open-ix? **SIGCOMM Comput. Commun. Rev.**, ACM, New York, NY, USA, v. 45, n. 1, p. 12–18, jan. 2015. ISSN 0146-4833. Available from Internet: <http://doi.acm.org/10.1145/2717646.2717650>.

CURTIS, A.; KIM, W.; YALAGANDULA, P. Mahout: Low-overhead datacenter traffic management using end-host-based elephant detection. In: **IEEE INFOCOM, 2011 Proceedings...** Shanghai, China: IEEE, 2011. p. 1629–1637.

CURTIS, A. R. et al. Devoflow: Scaling flow management for high-performance networks. **SIGCOMM Comput. Commun. Rev.**, ACM, New York, NY, USA, v. 41, n. 4, p. 254–265, aug. 2011. ISSN 0146-4833. Available from Internet: <http://doi.acm.org/10.1145/2043164.2018466>.

D3JS. **D3: Data Driven Documents**. https://www.d3js.org/: [s.n.], 2014.

ESTEVES, R. P.; GRANVILLE, L. Z.; BOUTABA, R. On the Management of Virtual Networks. **IEEE Communications Magazine**, v. 51, n. 7, p. 2–10, July 2013.

EURO-IX. **European Internet Exchange Association**. https://www.euro-ix.net: [s.n.], 2015.

EURO-IX. **IXP Best Common Operational Practices**. https://www.euro-ix.net/documents/1391-euro-ix-ixp-bcops-221014-pdf, 2015.

FACEBOOK. **Peering Policy**. https://www.facebook.com/peering/: [s.n.], 2015.

FANG, W.; PETERSON, L. Inter-as traffic patterns and their implications. In: **Global Telecommunications Conference, 1999. GLOBECOM '99, Proceedings...** Rio de Janeiro, Brazil: IEEE, 1999. v. 3, p. 1859–1868 vol.3.

FEAMSTER, N.; REXFORD, J.; ZEGURA, E. The road to sdn. **Queue**, ACM, New York, NY, USA, v. 11, n. 12, p. 20:20–20:40, dec. 2013. ISSN 1542-7730. Available from Internet: <http://doi.acm.org/10.1145/2559899.2560327>.

FRAMEWORK, D. **Django: the Web framework for perfectionists with deadlines**. https://www.djangoproject.com/: [s.n.], 2015.

GIOTIS, K. et al. Combining openflow and sflow for an effective and scalable anomaly detection and mitigation mechanism on sdn environments. **Comput. Netw.**, Elsevier North-Holland, Inc., New York, NY, USA, v. 62, p. 122–136, abr. 2014. ISSN 1389-1286. Available from Internet: <http://dx.doi.org/10.1016/j.bjp.2013.10.014>.

GOOGLE. **Peering Policy**. https://peering.google.com/about/ peering_policy.html: [s.n.], 2015.

GREGORI, E. et al. The Impact of IXPs on the AS-level Topology Structure of the Internet. **Computer Communications**, v. 34, n. 1, p. 68–82, 15 January 2011.

GROVER, N.; AGARWAL, N.; KATAOKA, K. liteflow: Lightweight and distributed flow monitoring platform for sdn. In: **Network Softwarization (NetSoft), 2015 1st IEEE Conference on, Proceedings...** London, U.K: IEEE, 2015. p. 1–9.

GUO, L.; MATTA, I. The war between mice and elephants. In: **Network Protocols, 2001. Ninth International Conference on, Proceedings...** Riverside, California, USA: [s.n.], 2001. p. 180–188.

GUPTA, A. et al. Sdx: A software defined internet exchange. **SIGCOMM Comput. Commun. Rev.**, ACM, New York, NY, USA, v. 44, n. 4, p. 551–562, aug. 2014. ISSN 0146-4833. Available from Internet: <http://doi.acm.org/10.1145/2740070.2626300>.

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

ISOLANI, P. H. et al. Interactive monitoring, visualization, and configuration of openflow-based sdn. In: **Integrated Network Management (IM), 2015 IFIP/IEEE International Symposium on, Proceedings...** Montreal, Canada: IEEE, 2015. p. 207–215.

KOTRONIS, V.; GAMPERLI, A.; DIMITROPOULOS, X. Routing centralization across domains via sdn: A model and emulation framework for {BGP} evolution. **Computer Networks**, v. 92, Part 2, p. 227 – 239, 2015. ISSN 1389-1286. Software Defined Networks and Virtualization. Available from Internet: <http://www.sciencedirect.com/science/article/pii/S1389128615002480>.

LAC-IX. **LAC-IX (Latin America & Caribbean Internet Exchange Association)**. http://www.lac-ix.org: [s.n.], 2015.

LIN, C.-Y. et al. Elephant flow detection in datacenters using openflow-based hierarchical statistics pulling. In: **Global Communications Conference (GLOBECOM), 2014 IEEE, Proceedings...** Austin, Texas, USA: IEEE, 2014. p. 2264–2269.

LIN, P. et al. A west-east bridge based sdn inter-domain testbed. **Communications Magazine, IEEE**, v. 53, n. 2, p. 190–197, Feb 2015. ISSN 0163-6804.

LIU, J. et al. Sdn based load balancing mechanism for elephant flow in data center networks. In: **Wireless Personal Multimedia Communications (WPMC), 2014 International Symposium on, Proceedings...** Sydney, Australia: IEEE, 2014. p. 486–490.

MACHADO, C. C. et al. Towards sla policy refinement for qos management in software-defined networking. In: **Advanced Information Networking and Applications (AINA), 2014 IEEE 28th International Conference on, Proceedings...** Victoria, Canada: IEEE, 2014. p. 397–404. ISSN 1550-445X.

MCKEOWN, N. et al. Openflow: Enabling innovation in campus networks. **SIGCOMM Comput. Commun. Rev.**, ACM, New York, NY, USA, v. 38, n. 2, p. 69–74, mar. 2008. ISSN 0146-4833. Available from Internet: <http://doi.acm.org/10.1145/1355734.1355746>.

MININET. **Mininet: An Instant Virtual Network on your Laptop (or other PC)**. http://mininet.org/: [s.n.], 2015.

MONGODB.ORG. 2016. Available from Internet: <https://www.mongodb.org/>.

MORI, T. et al. On the characteristics of internet traffic variability: spikes and elephants. In: **Applications and the Internet, 2004 International Symposium on, Proceedings...** Tokyo, Japan: IEEE, 2004. p. 99–106.

NARAYANAN, R. et al. Macroflows and microflows: Enabling rapid network innovation through a split sdn data plane. In: **Software Defined Networking (EWSDN), 2012 European Workshop on, Proceedings...** Darmstadt, Germany: IEEE, 2012. p. 79–84.

NETFLIX. **Peering Policy**. https://www.netflix.com/openconnect/ deliveryOptions: [s.n.], 2015.

NETWORKX.GITHUB.IO. 2016. Available from Internet: <https://networkx.github.io/>.

NUNES, B. et al. A survey of software-defined networking: Past, present, and future of programmable networks. **Communications Surveys Tutorials, IEEE**, v. 16, n. 3, p. 1617–1634, Third 2014. ISSN 1553-877X.

ONF. **Open Networking Foundation**. https://www.opennetworking.org: [s.n.], 2015.

OPEN-IX. **Open-IX Association**. http://www.open-ix.org: [s.n.], 2015.

OPENDAYLIGHT.ORG. 2016. Available from Internet: <https://www.opendaylight.org/>.

OPENVSWITCH. **Open vSwitch: Production Quality, Multilayer Open Virtual Switch**. http://openvswitch.org/: [s.n.], 2015.

OTTO, M.; CONTRIBUTORS, B. **Bootstrap: The world most popular mobile first and responsive frontend framework.** 2016. Available from Internet: <http://getbootstrap.com/>.

PROJECT Floodlight. 2016. Available from Internet: <http://www.projectfloodlight.org/floodlight/>.

PTTMETRO. **Pontos de Troca de Trafego Metropolitanos**. http://ptt.br, 2015.

PYTHON.ORG. 2015. Available from Internet: <https://www.python.org/>.

RAGHAVAN, B. et al. Software-defined internet architecture: Decoupling architecture from infrastructure. In: **11th ACM Workshop on Hot Topics in Networks, Proceedings...** New York, NY, USA: ACM, 2012. (HotNets-XI), p. 43–48.

RESTREPO, J. C. C.; STANOJEVIC, R. Ixp traffic: A macroscopic view. In: **7th Latin American Networking Conference, Proceedings...** New York, NY, USA: ACM, 2012. (LANC '12), p. 1–8.

RYU. **Ryu Openflow controller**. http://osrg.github.com/ryu: [s.n.], 2015.

SEZER, S. et al. Are we ready for sdn? implementation challenges for software-defined networks. **Communications Magazine, IEEE**, v. 51, n. 7, p. 36–43, July 2013. ISSN 0163-6804.

SFLOW. **sFlow.org**. http://www.sflow.org: [s.n.], 2015.

SILVA et al. **Metropolitan Internet eXchange - MIX (in Portuguese PTTMetro)**. http://ptt.br/mix.txt, 2004.

STRINGER, J. et al. Cardigan: Sdn distributed routing fabric going live at an internet exchange. In: **Computers and Communication (ISCC), 2014 IEEE Symposium on, Proceedings...** Madeira, Portugal: IEEE, 2014. p. 1–7.

SUH, J. et al. Opensample: A low-latency, sampling-based measurement platform for commodity sdn. In: **Distributed Computing Systems (ICDCS), 2014 IEEE 34th International Conference on, Proceedings...** Madrid, Spain: IEEE, 2014. p. 228–237. ISSN 1063-6927.

TAVARES GUIMARAES, V. et al. A collaborative solution for snmp traces visualization. In: **Information Networking (ICOIN), 2014 International Conference on, Proceedings...** Phuket, Thailand: IEEE, 2014. p. 458–463.

TEAM, M. **Mininet: An Instant Virtual Network on your Laptop (or other PC) - Mininet**. 2016. Available from Internet: <http://mininet.org/>.

VANBEVER, L. **Novel Applications for a SDN-enabled Internet Exchange Point**. Berlin, Germany, SDN Research Group meeting, IETF 87: [s.n.], 2013.

WICKBOLDT, J. et al. Software-Defined Networking: Management Requirements and Challenges. **IEEE Communications Magazine**, v. 53, n. 1, p. 278–285, January 2015.

XIA, W. et al. A Survey on Software-Defined Networking. **IEEE Communications Surveys and Tutorials**, v. 17, n. 1, p. 27–51, Firstquarter 2015.

XIAO, P. et al. An efficient elephant flow detection with cost-sensitive in sdn. In: **Industrial Networks and Intelligent Systems (INISCom), 2015 1st International Conference on, Proceedings...** Tokyo, Japan: IEEE, 2015. p. 24–28.

ZHANG, Y. An adaptive flow counting method for anomaly detection in sdn. In: **Ninth ACM Conference on Emerging Networking Experiments and Technologies, Proceedings...** Santa Barbara, California, USA: ACM, 2013. (CoNEXT '13), p. 25–30.

ZHANG, Y. et al. On the characteristics and origins of internet flow rates. **SIGCOMM Comput. Commun. Rev.**, ACM, New York, NY, USA, v. 32, n. 4, p. 309–322, aug. 2002. ISSN 0146-4833. Available from Internet: <http://doi.acm.org/10.1145/964725.633055>.

ZHANG, Z.; WANG, B.; LAN, J. Identifying elephant flows in internet backbone traffic with bloom filters and lru. **Comput. Commun.**, Elsevier Science Publishers B. V., Amsterdam, The Netherlands, The Netherlands, v. 61, n. C, p. 70–78, may 2015. ISSN 0140-3664. Available from Internet: <http://dx.doi.org/10.1016/j.comcom.2014.12.003>.

ZHOU, R. Datacenter network large flow detection and scheduling from the edge. In: **Reading & Research Project**. [S.l.: s.n.], 2014.

ApêndiceA ARTIGO PUBLICADO – NOMS 2016

- **Título**
  SDEFIX - Identifying Elephant Flows in SDN-Based IXP Network
- **Conferência**
  The 15th IEEE/IFIP Network Operations and Management Symposium (NOMS 2016)
- **Trilha**
  Main track (full-paper)
- **Qualis**
  A2
- **URL**
  <http://noms2016.ieee-noms.org/>
- **Data**
  25 à 29 de abril de 2016
- **Realizado em**
  Istanbul, Turquia
- **Digital Object Identifier (DOI)**
  <http://dx.doi.org>

SDEFIX - Identifying 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
Email: {luis.knob, rpesteves, granville}@inf.ufrgs.br, liane@penta.ufrgs.br

***Abstract*—Internet Exchange Points (IXPs) are fundamental blocks of the Internet ecosystem by providing cost-effective connections among multiple autonomous systems (ASes). One of the main management challenges in IXP networks is the management of the so-called elephant flows. Elephant flows, characterized by high throughput and long duration, represent a small fraction of the total flows of a IXP network but have high impact on the overall traffic. A first step in the management of elephant flows is their identification, which becomes 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 monitoring system to identify elephant flows in an SDN-based IXP network. We demonstrate that our system can assist IXP operators to properly identify elephant flows in an efficient, scalable, and timely manner.**

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

I. Introduction

Internet Exchange Points (IXPs) connect Internet autonomous systems (ASes) and enable service providers (SPs) to directly exchange traffic to better serve SPs' customers. IXPs are distributed worldwide and can serve dozen to hundreds of participants. According to a recent study [1], IXPs account for at least 20% of all traffic exchanged between ASes. Another important number shows that the daily aggregated traffic of a large IXP like DE-CIX can be compared with the traffic of a tier-1 ISP like AT&T [2]. One of the main benefits experienced by IXP participants is the low cost of deployment and maintenance [3]. As a consequence, companies such as Google [4], Facebook [5], and Netflix [6] seek to connect their infrastructures to major IXPs, thus reducing the costs of connecting customers to their services. Helping drive the Internet expansion for decades, IXPs are a critical building block of the global network.

IXPs enable the exchange of Internet traffic typically through an internal network of layer-2 switches [1]. The simplest IXP network is composed of a single switching device, while more complex IXPs can be formed by much larger networks with intricate topologies [2]. Not surprisingly, IXPs present management challenges that need to be tackled for the adequate IXP's operation [7]. That includes, for example, monitoring ARP traffic, establishing VPNs, fulfilling SLAs negotiated with participants, and dealing with critical flows and their associated network paths, which is central to this paper.

Recently, with the development of Software-Defined Networking (SDN) technologies [8], a new set of opportunities started to be considered in the management of IXP networks. In that context, a typical SDN controller would be expected to manage the set of switches that form the IXP network. To take advantage of SDN, however, a complementary process that properly snapshots the IXP network status is required beforehand. With such snapshots, the IXP's SDN controller can tune the underlying network to meet its goals. In this paper, we are specially interested in identifying the so-called *elephant flows* traversing an IXP network. A flow is considered an elephant one if it has long duration and generates a significant amount of traffic [9]. Elephant flows also tend to be reduced in number but they account for most of the traffic in network infrastructures. In addition of identifying elephant flows in IXPs, we are also interested in discovering the paths that these flows use to traverse the IXP networks. Knowing these paths allows the IXP operator, for example, to identify links that form bottlenecks and, through the SDN controller, balance the network load by moving elephant flows from their original busier paths to more appropriate ones.

In our solution, we provide the IXP operator with a monitoring system that allows him/her to discover elephant flows and associated paths. That results in snapshots of the IXP network that can further feed, for example, an SDN controller that determines how flows are placed in the IXP network. Depending on the criteria used by the operator, he/she will be able in the future, by using an SDN controller, to decrease the blocking rate of new traffic, reduce IXP traversal delay, or improve energy savings. All these benefits, however, start with the adequate identification of elephants flows and associated paths, as addressed in this paper.

The remainder of this paper is organized as follows. In Section 2, we review related work on the management of

978-1-5090-0223-8/16/$31.00

IXP networks. In Section 3, we present our proposed solution and report on the implementation of our prototype for the identification of elephant flows and associated paths in IXP networks. Our solution is then evaluated in Section 4, based on a network infrastructure inspired by AMS-IXP, which is an European IXP located in Amsterdam. We finally close this paper presenting concluding remarks and future work in Section 5.

II. Management of Internet Exchange Points

In this section, we review concepts that motivated our work. We first start by providing an introduction to IXPs. Then, we discuss management aspects of IXPs, focusing on the identification of elephant flows. Next, we review SDN in IXP management.

A. Internet Exchange Points

Internet eXchange Points (IXPs) are networking infrastructures that allow distinct Internet autonomous systems (ASes) to exchange traffic in a more efficient and cost-effective manner. By using IXPs, Internet Service Providers (ISPs) and Content Delivery Networks (CDNs) providers can exchange traffic without relying on third-parties, thus reducing the overall communication cost between ASes. IXPs evolved from the previous Network Access Points (NAPs) structures, which were deployed to orchestrate the transition from academic, government-sponsored networks (*e.g.*, NSFNET) to today's commercial Internet. Other similar initiatives included the Commercial Internet eXchange (CIX) and the Metropolitan Area Exchanges (MAEs). All these efforts resulted in the modern IXPs that are deployed worldwide [10] [11].

The commercial operation of IXPs can be divided into two separate groups: private and community. Private or "for-profit" IXPs are managed by a company that charges participants a monthly fee or based on the volume of traffic exchanged. On the other hand, community or "non-profit" IXPs are administered by the members themselves or by a non-profit organization. Typically, the model adopted by the IXPs across Europe, Africa, and Latin America is the community model, while in the USA and Canada there is a dominance of commercial IXPs. This happens both by commercial motivations (since its creation, the major American ASes have connections among one another) and legislative reasons, where some countries impose the traffic exchange between service providers operating in their territory [10].

Today, there are more than 300 active IXPs connecting thousands of ASes around the world [10]. IXPs are typically grouped according to their geographical location or the federation that they belong to. Examples of federations include Euro-IX (Europe) [12], LAC-IX (Latin America) [13], APIX (Asia-Pacific) [14], and Af-IX (Africa) [15]. The main goal behind federations is to promote and regulate the traffic exchange in these specific regions. All these federations are part of the global IX-F (Internet eXchange Federation). Reflecting their commercial nature, most IXPs from USA and Canada do not belong to any association. Recently, major Internet players such as Google, Netflix, and Akamai have started supporting the Open-IX Association [16]. The Open-IX Association defines a set of technical and operational standards for IXPs and emerged as a non-profit organization in North America that follows the model adopted in Europe.

B. Management of IXPs and Elephant Flows

Figure 1 depicts a infrastructure based on the AMS-IX [17], other IXPs tend to have similar structures. An IXP is typically composed of a number of layer-2 network switches or MPLS routers that allows ASes to exchange traffic. Each participating AS has to connect its access router to one of the switches/routers belonging to the IXP and establish a BGP session in order to peer with other ASes [11].

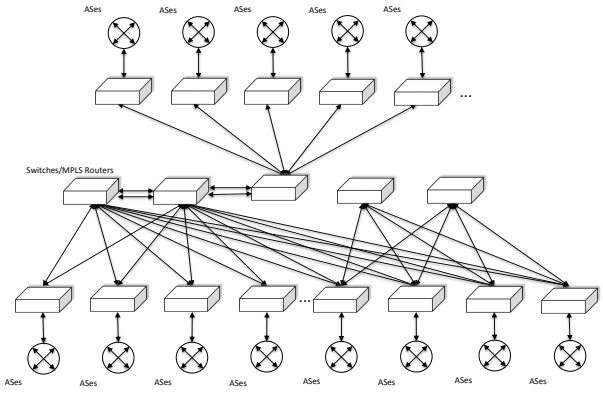


Fig. 1. A IXP infraestructure based on AMS-IX

The management of IXPs seeks the proper operation of the IXP infrastructure. For example, by using adequate monitoring tools, the IXP operator can have a better view of the IXP's network infrastructure to identify bottlenecks, handle failures, and make appropriate traffic engineering decisions to achieve better utilization of networking resources [18] [19]. Dealing with data confidentiality and promoting fair utilization of resources among ASes are other examples of IXP specific issues that need to be managed.

One IXP management aspect that has special importance is the management of *elephant flows*. In general, a flow is considered elephant if it has high throughput for a long period of time [20] [9] in comparison to other flows in the same network. Elephant flows account for a very small portion of the total flows of a network but can carry most of the traffic [9]. Managing elephant flows is important because of the potential impact of such flows in the IXP network. Elephant flows are long-lived flows that rapidly consume network buffers and impose undesirable queuing delays to short-lived ones, which are often referred to as mice flows. Therefore, mitigating the negative influence of elephant flows over mice flows and improving overall network utilization require appropriate management actions from the IXP operator.

The first step toward managing elephant flows in an IXP is to identify them. Once the IXP operator recognizes the elephant flows, he/she can take appropriate actions to optimize resource utilization and reduce the potential negative impact of elephant flows in the IXP network. For example, elephant flows can be isolated or migrated to other available network paths to make room for mice flows, use different routing strategies for elephant and mice flows, or use differentiated QoS schemes [9] [20].

C. SDN in IXPs

Software-Defined Networking (SDN) [21] [22] has been considered a viable alternative to tackle several issues in 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 [8] [23].

In the context of IXPs, SDN can overcome the difficulties inherent to the Border Gateway Protocol (BGP), namely routing based on the destination prefix, limited application of policies (direct neighbors), and indirect path selection [24]. In this regard, SDN becomes a powerful tool to improve IXP management by delivering functionalities that were hard or even impossible before. These functionalities are realized by novel applications, including application-specific peering, inbound traffic engineering, wide-area server load balancing, and upstream blocking of DoS attacks [24] [25].

In this paper, we consider the situation of an SDN-operated IXP whose SDN logically centralized controller needs to 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. Our goal is to snapshot the IXP traffic and, based on operators' requests, identify elephant flows and the path used by them to traverse the IXP network. We designed and implemented an IXP management system to achieve this goal, as presented in the next section.

D. Elephant Flows in SDN

There are several elephant flow identification strategies in SDN. A simple yet naive strategy is to define specific flow entries to capture elephant flows using an SDN standard protocol such as OpenFlow [26]. In this strategy, there are flow rules for identifying potential elephant flows in each switch of the network. These elephant flow rules are created using a flow abstraction (*e.g.*, OpenFlow). If a flow matches one or more elephant flow rules, then the flow is classified as an elephant flow. Other strategies were proposed in the context of SDN-based data center networks and include hierarchical statistics polling [27], host-based detection [28], and combination of sampling and triggering [29].

The problem with these strategies when applied to IXPs is threefold. First, they require collecting a large amount of traffic to allow the IXP operator to distinguish existing flows, which can slow down the identification process. Second, there should be as many flow entries as the number of potential elephant flows, which could exhaust the switch's flow table if the number of elephant flows to be identified is high. Finally, they can require modifications in the end-hosts or demand special hardware support that may not be possible in all IXPs.

In summary, current proposals for elephant flow identification are not well-suited for SDN-based IXPs. In the next section we introduce and detail our proposed monitoring system for SDN-based IXPs that overcomes the aforementioned limitations.

III. The SDN-based IXP Monitoring System

We developed a monitoring system named SDEFIX (Software-Defined Elephant Flow Identifier for Internet Exchange) that allows IXP operators to identify elephant flows and their associated network paths. We consider an environment where the IXP network is controlled by one logically centralized SDN controller, which runs applications that rule the underlying switches. Switches communicate with the controller through the OpenFlow protocol. Each OpenFlow switch maintains a flow table that is compared against incoming packets to determine the actions to be performed over those packets (*e.g.*, forward packet to a specific port) [26].

A. System Architecture

To overcome the scalability limitations of current elephant flow identification strategies in the context of IXPs, we designed a solution that uses sFlow [30] combined with OpenFlow. sFlow allows monitoring a large number of flows by employing flow *sampling*. Using sFlow, a sample packet is sent to a remote server, called *collector*, after a predefined number of packets is received by a network switch. In our solution, the collector stores samples in a NoSQL database (MongoDB). MongoDB provides a flexible structure to enable proper storage of packet samples. In MongoDB, the database schema may be dynamically modified according to the captured samples, which, in turn, allows storing information belonging to different network protocols. We use five MongoDB databases to store information regarding the identification rules, the collected flow samples, the flow tables of each switch, the network topology, and snapshots of identified flows, respectively.

Figure 2 depicts the architecture of our proposed system. It is composed by five main components and their respective databases: Elephant flow identifier, sFlow collector, Topology retriever, Flow Table retriever, and Administrative (Admin) interface. The Elephant flow identifier compares flow samples against identification rules and thresholds defined by the IXP operator through the Admin interface and returns the complete network path of each identified elephant flow. The sFlow collector collects flow samples in periodic intervals and sends the samples to the flow database. The Topology retriever and Flow Table retriever communicates with the SDN controller to gather the network topology and the flows tables of all

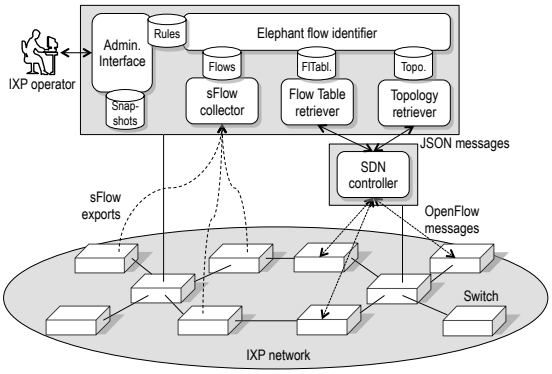


Fig. 2. SDN-based IXP Monitoring System

switches, and updates the topology and flow table databases, respectively. The Admin interface allows visualization of elephant flow paths and snapshots of previously identified flows, that may be used for historical record, or as a data source for IT infrastructure management. For each network switch, sFlow samples the traffic at specific polling intervals. We are employing the conventional interval of 10 seconds used in the sFlow's default configuration.

Although our solution is decoupled from any particular SDN controller, we are using the Ryu OpenFlow controller [31] in our implementation, since it was one of few controllers that supported latest OpenFlow versions and implements a well-defined REST API.

B. Elephant Flow Specification

In our solution, the IXP operator is responsible for defining what is an elephant flow. The IXP operator specifies identification rules to characterize elephant flows based on the following information: source/destination IP addresses, source/destination MAC addresses, transport protocol and port numbers. The IXP operator can use multiple rules to identify elephant flows and only rules strictly written are recognized by the algorithm. Additionally, it is globally defined a bandwitdh threshold (in bytes/seconds) and a duration threshold (in seconds) for all flows. Figure 3 present a list of all input fields in the system, including the fields for new identification rules, the definition of the bandwitdh and duration thresholds and the sampling configuration with polling interval and sample rate.

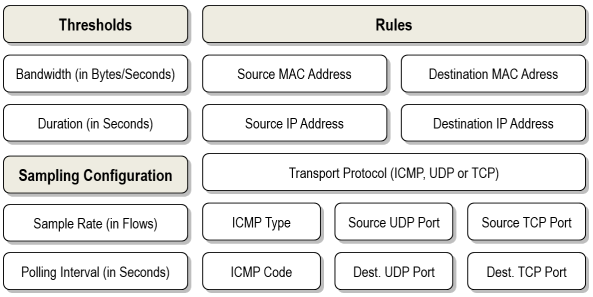


Fig. 3. SDN-based IXP Monitoring System - Input Fields

The identification rules are then used by the elephant flow identifier, which periodically compares these rules with the active flows on the network, searching for potential elephant flow candidates. A detailed overview of this module is explained in the next subsection.

C. Elephant Flow Identification

Some works presents solutions to identify elephant flows in SDN networks. The paper presented in [32] shows a solution for identifying elephant flows in composite structures with directing traditional and OpenFlow, checking via sampling packets above a dynamic threshold set by the average flow size on the network. After passing the threshold is added to the controller rules to validate if that flow is actually an elephant flow. If confirmed, the flow then receives treatment in a separate controller. While at work [33], an identification solution is presented using sFlow, capturing network samples in order to classify flows with thresholds for UDP, TCP and ICMP, using only the tool Inmon sFlow-RT [34]

Unlike other elephant flow identification approaches, our system allows the identification of the network paths associated with the elephant flows. Current solutions perform local, per switch flow identification, *i.e.*, the verification of the existence of elephant flows is performed at the switch scope, without knowledge of the other flows traversing the other switches of the infrastructure. The problem with that approach is that the IXP operator does not have a compiled list of devices through which an elephant flow passed along. Since in our system we keep track of the flow tables of all switches, we can match these flow tables with the identified elephant flows. If a switch has a flow entry corresponding to an elephant flow, we can include the switch in the path of the elephant flow. In this way, it is possible to derive the complete path for each identified elephant flow, helping the IXP operator to make appropriate management decisions with such flows.

Algorithm 1 illustrates the process of identifying elephant flows. The algorithm is triggered by the IXP operator when he/she wants to verify the elephant flows that are active in the IXP network at a given moment along with their associated network paths. In the elephant flow identification phase, the system compares the collected samples with the rules defined by the IXP operator (lines 9-11), if there is a match, the corresponding flow is added to the set of identified elephant flows (line 12). After that, the algorithm computes the network path associated with each identified elephant flow. This is performed by checking the flow table of each switch of the topology in order to find a flow entry that corresponds to the elephant flow (lines 17-21). If such an entry is found, the switch is included in the network path of the elephant flow (line 22).

As a result, the list of elephant flows matching the rules is then returned to the Admin Interface for a proper visualization by the IXP operator.

Algorithm 1 Elephant Flow Identification

R: set of rules defined by the IXP operator
S: set of collected flow samples
T: physical topology
$F(sw)$: flow table of the switch $sw \in T$
EF: set of identified elephant flows
$EFP(ef)$ network path of the elephant flow $ef \in EF$
Elephant Flow Identification:
$EF \leftarrow \varnothing$
for each $s \in S$ **do**
    for each $r \in R$ **do**
        if s matches r and s.bandwitdh $>$ *r.bandwitdh_threshold* and s.duration $>$ *r.duration_threshold* **then**
            add s to EF
        end if
    end for
end for
Elephant Flow Path Discovery:
for each $ef \in EF$ **do**
    $EFP(ef) \leftarrow \varnothing$
    for each $sw \in T$ **do**
        for each $f \in F(sw)$ **do**
            if $ef = f$ **then**
                add sw to $EFP(ef)$
            end if
        end for
    end for
end for

D. Displaying Elephant Flows Paths

Visualization and monitoring capabilities can assist IXP operators in network control and configuration tasks [35] [36]. In this perspective, SDNs require special attention, mainly because they present a high level of customization and programability. In this way, traditional visualization and monitoring solutions are not designed to cope with SDN requirements, mainly by the separation of the data and control plane and the necessity to manage both seamlessly. Therefore, an SDN-tailored monitoring/visualization solution can improve the understanding of the network behavior and simplify other management tasks [37].

After receiving the information from the Topology retriever and Elephant Flow Identifier modules, the Admin Interface returns to the user a visualization of the physical topology, as shown in Figure 4. The administrator can then visualize each identified elephant flow individually in the topology or can combine multiple elephant flows in a single view to highlight the most used network paths.

The operator can also generate a snapshot of the current network status and identified elephant flows for future analysis. Moreover, snapshots can be used to build a history of identified elephant flows and associated paths, and to assist the IXP operator in defining infrastructure update plans. When the system is initialized, the last saved snapshot is restored for the user.

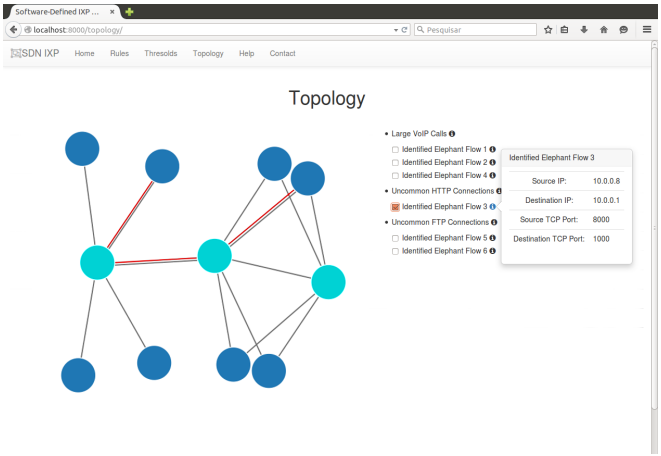


Fig. 4. Topology Visualization

In addition, the Admin interface was developed with the Django framework [38] that employs the MVT model (Model-View-Template) as a design pattern. Furthermore, the topology viewer was implemented using the D3js library [39], a powerful tool that uses JavaScript and SVG to create interactive visualizations.

IV. Evaluation

In this section, we evaluate the performance of SDEFIX in elephant flow identification. 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 composed of three virtual machines (VMs) running inside a Core i7 4790 with 16 GB of RAM through VirtualBox. Each VM has 2GB and runs Debian 8 as operating system. The first VM runs the Mininet emulator [40] and contains the topology used in the tests. The topology, shown in Figure 5 is a simplified version the AMS-IX infrastructure [17] using SDN-enabled switches (AMS-IX uses MPLS routers). In our topology, we replace the AMS-IX core routers by 3 core switches and we simulate the connection of the IXP with 8 ASes using 8 edge switches instead of the PE routers used in AMS-IX. All switches are based on the OpenVSwitch [41] implementation. The second VM hosts the Ryu SDN controller that was chosen because it allows rapid development of network management applications. SDEFIX and its main components (Elephant flow identifier, sFlow collector, Flow table retriever, Topology retriever, Admin interface, and all the MongoDB databases) runs in the third virtual machine. For the sake of simplicity, we do not detail the internal topology of each AS. We consider that there are 32 hosts inside each AS generating traffic.

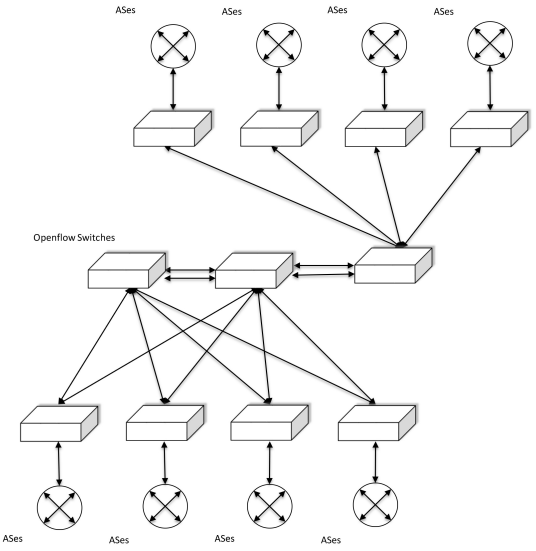


Fig. 5. Topology used in the experiments

In the experiments, the sFlow collector receives flow samples from the switches every 10 seconds, which is the conventional interval used in the sFlow's default configuration. The packet sampling rate varies from 1 in every 128 packets to 1 in every 4096 packets[1]. The size and duration thresholds used to characterize an elephant flow are 100Mbps and 10 seconds, respectively. Each host establishes 10 connections with others hosts in the network, 9 connections are used to characterize short (mice) flows with throughput between 4Mbps and 10Mbps, the remaining connection represents an elephant flow with throughput of 100Mbps. This means that there is a total of 128 elephant flows in the IXP network. The total number of flows (mice + elephant) is 1280.

We are interested in evaluating SDEFIX in terms of (1) *identification time*, (2) the *number of identified elephant flows*, (3) *resource usage*. The identification time is the time taken by the system to respond to a query executed by the IXP operator to identify the active elephant flows of the network at a given moment. The number of identified elephant flows is the number of elephant flows that were successfully identified by the system given a specific packet sampling rate. The sampling traffic is the amount of traffic resulting from the sampling process. Resource usage is defined in terms of CPU peak, memory usage, and database size. CPU peak and memory usage reflect the amount of resources consumed by SDEFIX in terms of CPU and memory, respectively. Database size is the sum of the size of all MongoDB databases used by SDEFIX. Each test was repeated 30 times with a confidence interval of 95%. In the next subsection, we present the main results of our evaluation.

[1]From this point, we use the notation $1/x$ to represent a packet sampling rate of 1 in every x packets.

B. Results

Figure 6 depicts the identification time when the number of rules used to identify elephant flows (identification rules) increases. In this case, the packet sampling rate is fixed to 1/128. Here, we compare two approaches for elephant flow identification. The first one, referred to as *sFlow identification*, is the case when the elephant flow and its corresponding path are identified when samples from that flow are collected from all switches of the path. That is, the path of an elephant flow can only be calculated if a sample of that flow is collected in all the switches of the path. In the second approach, called *2-step identification*, if samples of an elephant flow are collected from at least one switch, then the corresponding path can be calculated by matching the elephant flow with the entries in flow tables collected by the system using the strategy described in Algorithm 1. That is, if the elephant flow is found in a flow table of a switch, the switch is included in the path traversed by the elephant flow.

It is possible to observe from Figure 6 that 2-step identification performs in average 6% better than pure sFlow identification for a 1/128 sampling rate. Identification time remains stable around 0.6 seconds if the IXP operator queries 1 up to 32 elephant flows. For 64 and 128 identification rules, the identification time is 0.9 seconds and 1.8 seconds, respectively, which we believe is acceptable in realistic scenarios. Besides, the identification time increases in a slower rate compared to the number of identification rules. The identification time is three times higher (0.6 to 1.8) for a number of identification rules that is more than a hundred times higher (1 to 128), which represents a difference of two orders of magnitude.

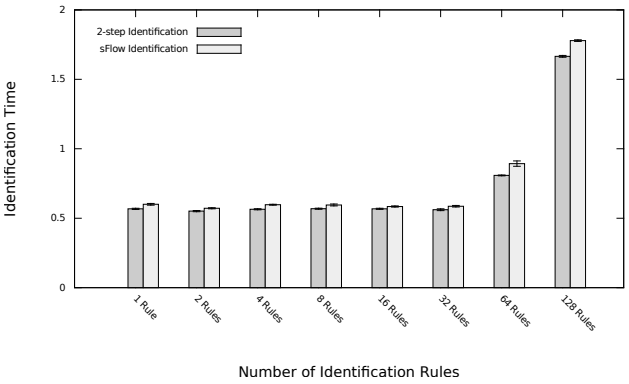


Fig. 6. Identification Time per Number of Identification Rules

We also measured the identification time for different packet sampling rates (1/4096, 1/2048, 1/512, 1/256, and 1/128 packets) when the number of identification rules is fixed in 128. We observed that the identification time does not vary significantly with the sampling rate, and the results are basically the same of Figure 6, which represents the worst case, *i.e.*, packet sampling rate of 1/128.

Figure 7 shows the average number of identified elephant flows when the packet sampling rate varies from 1/4096 to

1/128. It is possible to observe from Figure 7 that the number of identified flows increases linearly with the packet sampling rate for both identification approaches. This happens because the chances of collecting a flow sample that belongs to an elephant flow increase when the number of total flow samples is higher. The 2-step identification performs in average 20% better than pure sFlow identification. The difference between the two approaches can be explained by the fact that in sFlow identification there are some elephant flows that could not be identified in one or more switches of the IXP network, *i.e.*, there were no collected samples for those flows, and, as a consequence, the corresponding path could not be determined. On the other hand, in 2-step identification, the system needs that at least one switch sends a flow sample of an elephant flow to the sFlow collector. With a single flow sample, the system is able to determine precisely the corresponding path by comparing the flow sample with the flow entries of every switch of the IXP network.

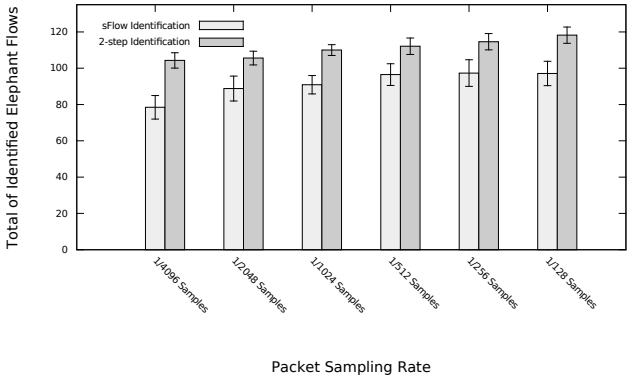


Fig. 7. Number of Identified Elephant Flows per Packet Sampling Rate

Table I shows the sampling traffic for different packet sampling rates. There is a significant difference (more than 20 times) in terms of generated traffic when a sampling rate of 1/128 (224 MB) is used, compared to a sampling rate of 1/4096, which generates only 11 MB of network traffic.

TABLE I
SAMPLING TRAFFIC (MBYTES)

Sampling rate	Traffic
1/128	224.4
1/256	101.2
1/512	61.6
1/1024	37.4
1/2048	19.1
1/4096	11

Table II summarizes the results regarding resource usage varying the number of identification rules (1 and 128) and the sampling rate (1/128 and 1/4096). CPU peak is mainly dependent on the number of identification rules and impacts the identification time as shown in Figure 6. The sampling rate has little influence in CPU usage. Memory usage remains stable around 155 MB. As expected, the sampling rate is directly related to the size of the database (*i.e.*, Flow Sample database) used by SDEFIX to store the collected samples.

TABLE II
RESOURCE USAGE

Metric	1/128 (128R)	1/128 (1R)	1/4096 (128R)	1/4096 (1R)
CPU peak	10.2%	4.3%	8.6%	2.8%
Memory usage (MB)	155	155	155	155
Database size (MB)	215.4	215.1	38.8	38.5

C. Summary

In summary, the identification time depends mainly on the number of identification rules defined by the IXP operator because, in the worst case, the system needs to compare flow samples with all the rules. Our 2-step elephant flow identification approach is capable of identifying in average 20% more elephant flows compared to the traditional pure sampling-based identification approach, which demonstrates the benefits of using an SDN-based solution for elephant flow identification.

As expected, there is a tradeoff between the precision of elephant flow identification and the amount of resources that are needed to allow such identification. The amount of generated traffic for a sampling rate of 1/128 is more than 20 times higher than the traffic generated by a sampling rate of 1/4096 packets. However, the precision gain with the 1/128 rate is only 13% better when compared to the 1/4096 rate (Figure 7). Resource usage in terms of CPU and memory indicates that SDEFIX is lightweight and can deployed in commodity hardware. The main bottleneck is the database size, which is dependent on the sampling rate. Therefore, the IXP operator has to weigh both the accuracy of identification and the impact of the generated traffic and choose accordingly.

V. CONCLUSION

In this paper we have proposed and presented SDEFIX, a monitoring system for SDN-based IXPs to allow identification of elephant flows and their associated paths. SDEFIX uses a sampling approach (sFlow) and leverages the flow table abstraction of the OpenFlow protocol to improve the precision of identification. Unlike most approaches for elephant flow identification based on sampling, SDEFIX does not require that every switch in the path traversed by an elephant flow sends a sample of that flow in order to calculate its path. Instead, in SDEFIX, only one sample is needed to allow the identification of an elephant flow and its associated path. We have evaluated SDEFIX in terms of identification time, number of identified flows, and sampling rate.

Evaluation results show that identification time depends mainly on the number of identification rules defined by the IXP operator and is not significantly influenced by the packet

sampling rate. The packet sampling rate affects the accuracy of the identification, that is, the number of identified elephant flow increases with the sampling rate. However, improving the precision of elephant flow identification by using a higher sampling rate results in increased sampling traffic, which can affect the overall performance. Therefore, the IXP operator has take both factors in consideration when choosing an adequate packet sampling rate. In resource constrained scenarios, it may be more important not to generate an excessive amount of sampling traffic, which can potentially affect other services. On the other hand, if precision of identification is crucial, an increase in sampling traffic can be tolerated.

Elephant flow identification is a first step towards the management of SDN-based IXPs. Such information can help IXP operators to define appropriate actions to handle such critical flows. As future work, we plan to carry additional experiments to evaluate SDEFIX considering different topologies and scenarios. We also intend to extend SDEFIX by taking the result of the identification as an input to adjust the network paths used by critical elephant flows.

REFERENCES

[1] J. C. C. Restrepo and R. Stanojevic, "IXP Traffic: A Macroscopic View," in *Latin American Networking Conference (LANC)*, Medellin, Colombia, 4–5 October 2012, pp. 1–8.

[2] 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.

[3] 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, 15 January 2011.

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

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

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

[7] J. Matias, B. Tornero, A. Mendiola, E. Jacob, and N. Toledo, "Implementing Layer 2 Network Virtualization Using OpenFlow: Challenges and Solutions," in *European Workshop on Software Defined Networking (EWSDN)*, Darmstadt, Germany, 25–26 October 2012, pp. 30–35.

[8] 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.

[9] 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.

[10] N. Chatzis, G. Smaragdakis, A. Feldmann, and W. Willinger, "There is More to IXPs Than Meets the Eye," *ACM SIGCOMM Computer Communication Review*, vol. 43, no. 5, pp. 19–28, November 2013.

[11] B. Ager, N. Chatzis, A. Feldmann, N. Sarrar, S. Uhlig, and W. Willinger, "Anatomy of a Large European IXP," in *ACM Conference on Applications, Technologies, Architectures, and Protocols for Computer Communication (SIGCOMM)*, Helsinki, Finland, 13–17 August 2012, pp. 163–174.

[12] Euro-IX. (2015) European Internet Exchange Association. https://www.euro-ix.net.

[13] LAC-IX. (2015) Lac-ix (latin america & caribbean internet exchange association). http://www.lac-ix.org.

[14] APIX. (2015) Asia Pacific Internet Exchange Association. http://apix.asia.

[15] Af-IX. (2015) African Internet Exchange Association. http://www.af-ix.net.

[16] Open-IX. (2015) Open-IX Association. http://www.open-ix.org.

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

[18] 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.

[19] 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.

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

[21] 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.

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

[23] 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.

[24] 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.

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

[26] 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.

[27] 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.

[28] 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.

[29] 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.

[30] sFlow. (2015) sFlow.org. http://www.sflow.org.

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

[32] C. Bi, X. Luo, T. Ye, and Y. Jin, "On precision and scalability of elephant flow detection in data center with sdn," in *Globecom Workshops (GC Wkshps), 2013 IEEE*, Dec 2013, pp. 1227–1232.

[33] M. Afaq, S. U. Rehman, and W.-C. Song, "A framework for classification and visualization of elephant flows in sdn-based networks," *Procedia Computer Science*, vol. 65, pp. 672 – 681, 2015, international Conference on Communications, management, and Information technology (ICCMIT'2015). [Online]. Available: http://www.sciencedirect.com/science/article/pii/S1877050915028410

[34] InMon. (2015) sflow-rt. www.inmon.com/products/sFlow-RT.php.

[35] C. C. Machado, G. Tavares, L. Z. Granville, A. Schaeffer-Filho, and J. A. Wickboldt, "Towards sla policy refinement for qos management in software-defined networking," in *Advanced Information Networking and Applications (AINA), 2014 IEEE 28th International Conference on*, May 2014, pp. 397–404.

[36] V. Tavares, Guimaraes, G. L. dos Santos, G. da C. Rodrigues, L. Z. Granville, and L. R. Tarouco, "A collaborative solution for snmp traces visualization," in *Information Networking (ICOIN), 2014 International Conference on*, Feb 2014, pp. 458–463.

[37] P. H. Isolani, J. A. Wickboldt, C. Both, J. Rochol, and L. Z. Granville, "Interactive monitoring, visualization, and configuration of openflow-based sdn," in *Integrated Network Management (IM), 2015 IFIP/IEEE International Symposium on*, May 2015, pp. 207–215.

[38] D. Framework. (2015) Django: the Web framework for perfectionists with deadlines. https://www.djangoproject.com/.

[39] D3js. (2014) D3: Data Driven Documents. https://www.d3js.org/.

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

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

ApêndiceB ARTIGO SUBMETIDO – GLOBECOM NOMS 2016

- **Título**
  SDEFIX - Identifying Elephant Flows in SDN-Based IXP Network
- **Conferência** IEEE GLOBECOM 2016

- **Trilha**
  Main track (full-paper)
- **Qualis**
  A1
- **URL**
  <http://globecom2016.ieee-globecom.org/>
- **Data**
  4 à 8 de dezembro de 2016
- **Realizado em**
  Washington, EUA
- **Digital Object Identifier (DOI)**
  <http://dx.doi.org>

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
Email: {luis.knob, rpesteves, granville}@inf.ufrgs.br, liane@penta.ufrgs.br

***Abstract*—Internet Exchange Points (IXPs) play a key role in the current Internet architecture enabling cost-effective connections among multiple autonomous systems (ASes). Management of IXP networks is primarily concerned with the management of the so-called elephant flows. Such flows represent a small portion of the total flows of a IXP network but usually have high impact on the overall 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 active in the network. Elephant flow management becomes 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 reconfigured to achieve a specific objective. We demonstrate that our system can help IXP operators to mitigate the impact of elephant flows in the IXP network.**

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

I. Introduction

Accounting for at least 20% of all traffic exchanged between 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 flow in IXP networks by exploiting SDN benefits.

In this paper, we introduce an SDN-based IXP management solution that enables to 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. In fact, 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 human operator, thus delivering a refined control loop.

The remainder of this paper is organized as follows. In Section 2, we 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. Related Work

In this section we review prominent proposals related to the management of SDN-based IXPs and management of elephant flows.

SDX [6] uses SDN concepts to build a software-defined IXP in order to overcome the limitations of BGP protocol, namely routing based on the destination prefix, limited application of policies, and indirect path selection. In SDX, participant ASes can 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 focus on the identification and management of elephant flows.

Mahout [17] attempts to reduce monitoring overheads in switches by detecting elephant flows at end hosts in a data center network. The solution requires the inclusion of a shim layer at the operation system of the end host, which is not applicable in IXPs. DevoFlow [18] uses OpenFlow [19] to keep track of elephant flows only and offers different statistics collection mechanism such as sampling and triggering. In DevoFlow, if a flow reaches a predefined threshold in terms of number of bytes, then the flow is classified as an elephant flow and a decreasing best-fit bin packing algorithm is applied to calculate the least congested path between the flow's endpoints. Like DevoFlow, our solution can use sampling to support the identification process. However, we add flexibility to allow the IXP operator to define the criteria used to select alternative paths for elephant flows.

ESHSP (Elephant Sensitive Hierarchical Statistics Pulling) [20] proposes a iterative process to detect elephant flows by decomposing the flow space until an elephant flow is isolated from the others. Unlike our proposal, ESHSP focus only on the elephant flow detection and does not perform further actions with such flows. Similar to our proposal, OpenSample [21] relies on sFlow for monitoring and enables traffic engineering in SDNs. The main difference is that OpenSample relies on TCP sequence numbers for accuracy and we use the switches's flow entries. 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 bandwitdh. Our proposal, on the other hand, enables the IXP operator to tune the network according to other objectives rather than bandwidth guarantees.

In summary, current proposals for elephant flow management present limitations in the context of SDN-based IXPs. 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 integrated to the SDEFIX solution [16] that allows IXP operators to mitigate the impact of elephant flows in the IXP network. We consider an environment where the IXP network is controlled by one logically centralized SDN controller that ultimately defines the physical paths used for communication inside the IXP. Switches communicate with the controller through the OpenFlow protocol [19].

A. System Architecture

We use elephant flow snapshots generated by SDEFIX as input to the recommendation system. SDEFIX is a monitoring system for SDN-based IXPs that allows identification of elephant flows and their associated paths. SDEFIX uses a sampling approach (sFlow) and leverages the flow table abstraction of the OpenFlow protocol to improve the precision of identification. SDEFIX does not require that an elephant flow is identified at every switch of the path. Instead, in SDEFIX, only one flow sample is needed to identify an elephant flow and its associated path. The elephant flow path is obtained by matching a candidate flow sample collected in a switch with the flow table entries of the other switches. The identified elephant flow paths are then grouped into a set called snapshot.

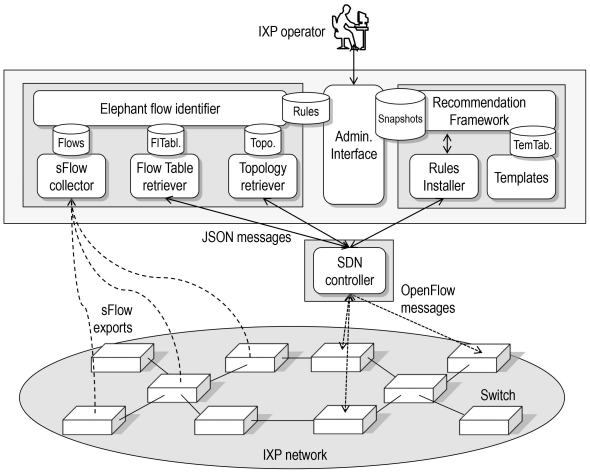


Fig. 1. SDN-based IXP Recommendation System

Figure 1 depicts the addition of the recommendation system in the existing architecture of SDEFIX. Besides adding specific functions to the *Admin Interface* module to assist the administration of the framework by the IXP operator (*e.g.*, access control to new modules), the infrastructure has 3 new modules: *Rules Installer*, *Template Management*, *Recommendation Framework*, and their associated databases, when needed. The *Recommendation Framework* module applies *templates*, algorithms used to modify existing flows, using the collected elephant flow snapshots. The IXP operator is responsible for defining and adding templates to the Recommendation Framework module. 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. The result of the application of a template over a snapshot is a recommended set of SDN rules (OpenFlow-like rules) that can be deployed in the infrastructure. The IXP operator may want to adjust a recommendation before applying it to the network, by for example, removing specific switches/links from a suggested path and running the template again with the proposed modifications. The *Template Management* module is where the IXP operator can add different templates that can be used to achieve different objectives such as load balancing, improved link utilization, bandwidth/delay guarantees, etc. Finally, the *Rules installer* module is responsible for installing the flow rules on the SDN controller after the IXP operator validates the template. The SDN controller installs the rules on the

appropriate switches using the OpenFlow protocol.

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). If the recommendation is validated by the IXP operator, the rules are installed to the SDN controller (lines 17-18). Algorithm 2 is an example of a template that can be applied to the snapshotted elephant flows. It tries to find the best alternative path (BAP) 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.

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 use the Ryu OpenFlow controller [22] 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 [23].

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.

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

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

The template used to exemplify the proposed recommendation framework is presented in algorithm 2. It computes the best alternative shortest path, *i.e.*, the second shortest path, for every elephant flow stored in SDEFIX. The ideia is to migrate an elephant flow to a new, possibly not congested path, while not significantly increasing network latency.

Templates are written in Python and use a well-defined API as shown in Figure 2. 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].

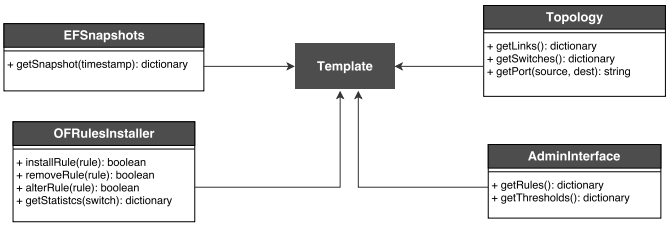


Fig. 2. Template API

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 a 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.

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

This validation may add new parameters to the template. For example, in the alternative shortest path template, the IXP operator can add weights to paths that he/she does not want to use, for any reason, such as to avoid high latencies, or to exclude specific network equipments. 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.

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 3. 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 generates the recommendation illustrated in Figure 4 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.

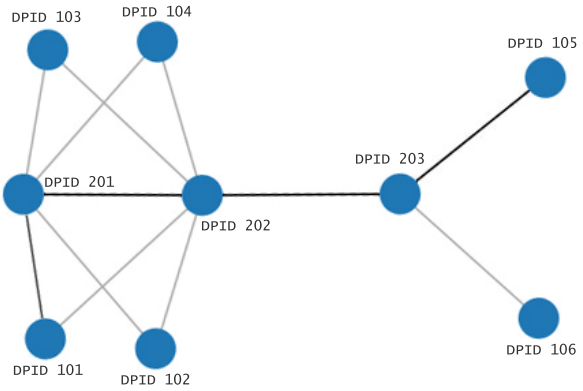


Fig. 3. Snapshot used as input

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

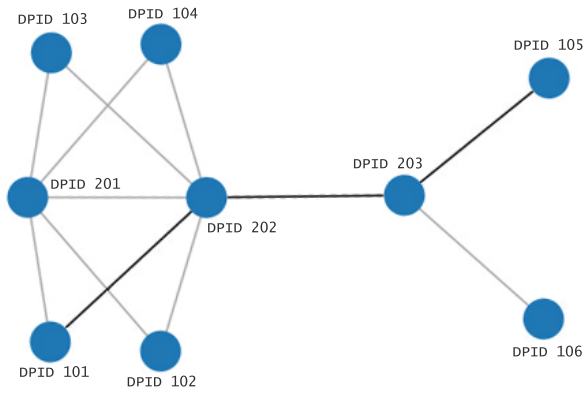


Fig. 4. Elephant flow after template validation

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 [24] replacing AMS-IX's original MPLS routers by SDN-enabled switches as shown in Figure 5. We emulate the topology using the Mininet emulator [25] running in a virtual machine (VM) with 2GB of RAM. OpenVSwitch [26] 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.

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. In 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

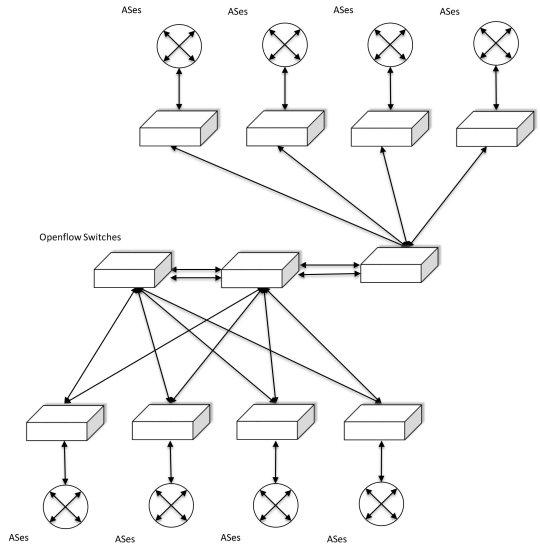


Fig. 5. Topology used in the experiments

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 6 depicts the time taken in each phase separately.

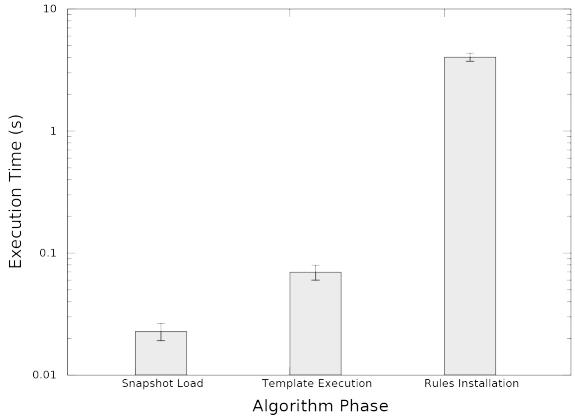


Fig. 6. Mitigation time

It is possible to observe from Figure 6 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 7 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 3 and 4) .

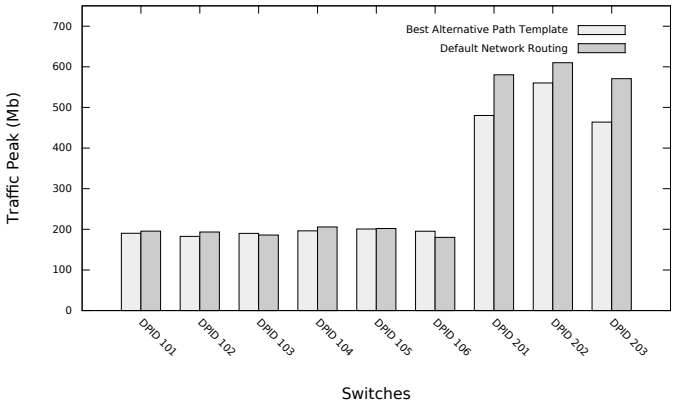


Fig. 7. Traffic peak in switches

The effect of the application of the BAP template can be better noticed in the core switches that handle most of the 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 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, different templates designed 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.

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] N. Chatzis, G. Smaragdakis, and A. Feldmann, "On the importance of internet exchange points for today's internet ecosystem," *CoRR*, vol. abs/1307.5264, 2013. [Online]. Available: http://arxiv.org/abs/1307.5264

[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. [Online]. Available: http://doi.acm.org/10.1145/2390231.2390239

[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. [Online]. Available: http://dx.doi.org/10.1016/j.bjp.2013.10.014

[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. [Online]. Available: http://XXXXX/150481.pdf

[17] 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.

[18] 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.

[19] 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.

[20] 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.

[21] 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. [Online]. Available: http://dx.doi.org/10.1109/ICDCS.2014.31

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

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

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

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

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