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

MARCUS VINICIUS BRITO DA SILVA

Prevendo e Identificando Fluxos Elefantes em Redes de Ponto de Troca de Tráfego com Suporte à Programabilidade

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

Orientador: Prof. Dr. Lisandro Zambenedetti Granville

Porto Alegre
2019

CIP – CATALOGAÇÃO NA PUBLICAÇÃO

Silva, Marcus Vinicius Brito da

Prevendo e Identificando Fluxos Elefantes em Redes de Ponto de Troca de Tráfego com Suporte à Programabilidade / Marcus Vinicius Brito da Silva. – Porto Alegre: PPGC da UFRGS, 2019.

96 f.: il.

Dissertação (mestrado) – Universidade Federal do Rio Grande do Sul. Programa de Pós-Graduação em Computação, Porto Alegre, BR–RS, 2019. Orientador: Lisandro Zambenedetti Granville.

1. Gerenciamento de Redes. 2. Pontos de Troca de Tráfego. 3. Redes Definidas por Software. 4. Redes Programáveis. I. Granville, Lisandro Zambenedetti. II. Título.

“Se um elefante tiver o seu pé sobre a cauda de um rato e você disser que é neutro, o rato não apreciará a sua neutralidade.”

— DESMOND TUTU

AGRADECIMENTOS

"Ainda que eu conheça todos os mistérios e toda a ciência, e ainda tenha uma fé capaz de mover montanhas, se não tiver amor, nada serei." (1Cr 13,2)

Em primeiro lugar, agradeço a Deus, por todas as graças recebidas. Agradeço por me dar forças, saúde e pessoas maravilhosas que foram essenciais durante toda minha jornada até aqui.

Os mais sinceros agradecimentos aos meus pais, Antonio Raiol da Silva e Vera Lucia Brito dos Reis, por todo icentivo, confiança, lutas e esforços que me fazem constantemente aprender e continuar crescendo. Que nossos sonhos sejam regados pelas lágrimas que só nós sentimos.

Um especial e carinhoso agradecimento à minha amada namorada Josefa de Paula, pelo companheirismo, paciência e os sacrifícios que enfrentamos nessa jornada. Que mais degraus sejam construídos juntos em nossas vidas, com as bençãos de Deus.

Os meus irmãos, Marcelo Victor e Natalia, por todo carinho e por estarem fisicamente ao lado dos meus pais, durante minha ausência. Agradeço a todos meus familiares, que mesmo na distância sempre me apoiaram e permaneceram próximos.

Agradeço imensamente ao Prof. Dr. Lisandro Z. Granville, primeiro pela oportunidade e aceite de tê-lo como orientador, por ter acreditado e investido em mim. Agradeço pelos valiosos aprendizados e ensinamentos. Pelas orientações, conversas, paciência e compreensão que tornaram a experiência desses anos viável e produtiva.

Agradeço também a todos os professores do Instituto de Informática da UFRGS, pelos importantes ensinamentos transmitidos em aulas e na convivência. Os ensinamentos e oportunidades que todos me proporcionaram foram de grande importância para o desenvolvimento desse mestrado e continuarão sendo, na minha carreira profissional. Agradeço à Universidade Federal do Rio Grande do Sul pela oportunidade e acolhida. Ainda, agradeço ao Conselho Nacional de Desenvolvimento Científico e Tecnológico (CNPq) pelo incentivo financeiro indispensável para a realização desse mestrado.

Aproveito a oportunidade para agradecer a todos os amigos que fiz durante esse período, sejam dentro e ou fora do INF. Pelos bons momentos, incentivos, ajudas e colaborações. A todos, MUITO OBRIGADO!

RESUMO

Pontos de Troca de Tráfego (PTTs) são redes de alto desempenho e permitem que vários sistemas autônomos da Internet realizem troca de tráfego, com benefícios que vão desde reduções de custo à melhorias de desempenho. Como em qualquer rede, os operadores de PTTs enfrentam diários desafios de gerenciamento para promover um melhor uso dos serviços fornecidos pela rede. Um problema essencial no gerenciamento de PTTs diz respeito à identificação de fluxos elefantes, que se caracterizam por ter duração e tamanho de tráfego significativamente altos em relação a outros fluxos. Diante disso, esse trabalho apresenta um mecanismo para prever o comportamento de fluxos e identificar os fluxos elefantes, mesmo antes que excedam os limiares. Para isso, são utilizadas observações históricas de fluxos anteriores e a correlação temporal de seus comportamentos. A avaliações de desempenho mostra que o mecanismo de predição é capaz de prever o tamanho e a duração dos novos fluxos e reagir rapidamente aos fluxos elefantes, com apenas 32 amostras históricas no modelo de inferência. A acurácia do mecanismo obteve até 80% de sucesso em cenários conservadores, com aproximadamente 5% de falsos positivos. Além disso, também é apresentado o IDEAFIX, um mecanismo para identificar fluxos elefantes diretamente no plano de dados, utilizando *switches* programáveis, quando não é possível validar as predições. Essa abordagem analisa o volume e duração dos fluxos para cada pacote que entra na rede, imediatamente no *switch* de borda. As informações são armazenadas em registradores, indexados por chaves *hash*, e comparadas com limiares predefinidos para classificar os fluxos. A avaliação mostra que o IDEAFIX é significativamente mais eficiente que as abordagens do estado-da-arte implementadas em SDNs (*Software Defined Networking*) tradicionais (por exemplo, *OpenFlow*), que utilizam amostragem de fluxo com *sFlow*. Enquanto o mecanismo baseado no estado-da-arte soma até 17MB de dados de monitoramento inseridos na rede, o IDEAFIX incorre em apenas 25KB. Por fim, o protótipo implementado em linguagem P4 leva menos de 0.40ms para identificar e reagir aos fluxos elefantes, com precisão de 95% em cenários com recursos de memória escassos.

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

Predicting and Identifying Elephant Flows in Internet Exchange Point Programmable Networks

ABSTRACT

Internet Exchange Points (IXPs) are high-performance networks that allow multiple autonomous systems to exchange traffic, with benefits ranging from cost reductions to performance improvements. As in any network, IXP operators face daily management challenges to promote better usage of the services provided by the network. An essential problem in IXP management concerns the identification of elephant flows, which are characterized by having traffic size and duration significantly higher than other flows. Therefore, we present a mechanism to predict flows behavior using historical observations and, by recognizing temporal patterns, identify elephant flows even before they exceed such thresholds. The prediction mechanism is able to predict the new flows size and duration, and react to elephant flows rapidly, with up to 32 historical samples in the prediction model. The mechanism accurately predicts up to 80% of elephant flows in conservative scenarios, with approximately 5% of false positives. In addition, we present IDEAFIX, a mechanism to identify elephant flows directly in the data plane, using programmable switches, when it is not possible to validate the predictions. This approach analyzing flows features for each ingress packet immediately in the edge switch. These features are then stored in registers, indexed by hash keys, and compared to predefined thresholds for flow classification. The evaluations show that IDEAFIX is significantly more efficient than the state-of-the-art approaches implemented with sFlow and traditional Software-Defined Networking (SDN) tools (*e.g.,* OpenFlow). While state-of-the-art mechanisms add up to 17MB of monitoring data, our solution causes an overhead of only 25KB. Also, the prototype implemented in P4 language takes less than 0.40ms to identify elephant flows with a 95% accuracy in scenarios with scarce memory resources.

Keywords: Network Management, Internet Exchange Point, Software-Defined Networking, Programmable Networks.

LISTA DE ABREVIATURAS E SIGLAS

AMS-IX	*Amsterdam Internet Exchange*
ARP	*Address Resolution Protocol*
ASes	*Autonomous Systems*
BGP	*Border Gateway Protocol*
CPU	*Central Process Unit*
IP	*Internet Protocol*
IX	*Internet Exchange*
IXPs	*Internet Exchange Points*
LWR	*Locally Weighted Regression*
NSW-IX	*New South Walles Internet Exchange*
OSPF	*Open Shortest Path First*
PTTs	Pontos de Troca de Tráfego
QoS	*Quality of Service*
SDN	*Software Defined Networking*
TCP	*Transmission Control Protocol*
UDP	*User Datagram Protocol*
VIC-IX	*Victoria Internet Exchange*
VPN	*Virtual Private Network*
WA-IX	*Western Australia Internet Exchange*

LISTA DE FIGURAS

LISTA DE TABELAS

SUMÁRIO

1 INTRODUÇÃO

Pontos de Troca de Tráfego (PTTs) conectam sistemas autônomos da Internet (do inglês, *Autonomous Systems* - ASes) e permitem que os provedores de serviços realizem troca de tráfego com a finalidade de melhor atender seus clientes (AGER *et al.*, 2012). Os PTTs são distribuídos por todo o mundo e desempenham um papel essencial no ecossistema da Internet, contabilizando pelo menos 20% de todo o tráfego trocado entre ASes (RESTREPO; STANOJEVIC, 2012). Além disso, a infraestrutura de um PTT pode ser formada por várias redes, com topologias intricadas (AUGUSTIN; KRISHNAMURTHY; WILLINGER, 2009). E ainda, alguns dos principais benefícios experimentados pelos participantes de um PTT é o baixo custo de implantação e manutenção (GREGORI *et al.*, 2011). Esses fatores estimulam empresas (como *Google* e *Netflix*) a conectarem suas infraestruturas aos principais PTTs, buscando assim reduzir os custos de conexão dos clientes com seus serviços (KNOB *et al.*, 2017).

Não surpreendentemente, desafios de gerenciamento são evidenciados no contexto de redes de PTTs, tais como: monitoramento de tráfego *Address Resolution Protocol* (ARP), estabelecimento de *Virtual Private Network* (VPN), ou estratégias para lidar com fluxos críticos e seus caminhos na rede. Diante disso, trabalhos têm abordado esses desafios em redes de PTT utilizando *Software Defined Networking* (SDN). SDN, por sua vez, tem contribuído fortemente para a evolução das redes de computadores e investigações têm sido realizadas sobre os benefícios de sua adoção (MCKEOWN *et al.*, 2008a). Suas principais características são evidenciadas pelo suporte ao roteamento de pacotes baseado em regras e pela abstração da lógica de controle para um *software*, chamado de controlador (ARAUJO *et al.*, 2017).

Diante dos desafios encontrados para gerenciar os fluxos que trafegam sobre a infraestrutura de um PTT, a identificação dos fluxos chamados elefantes pode favorecer uma melhor utilização dos serviços prestados pela rede (KNOB *et al.*, 2016). Um fluxo é considerado um elefante se tiver duração e volume de tráfego significativamente altos segundo limiares predefinidos (GUO; MATTA, 2001). Apesar de representarem uma fatia pequena dentre todos os fluxos em um PTT, as características de tráfego dos fluxos elefantes tendem a exaurir rapidamente os recursos dos dispositivos de rede (MORI *et al.*, 2004a). Como consequência, estes podem impactar significativamente no tráfego de fluxos menores, aqueles cujo comportamento não excede os limiares, que estejam compartilhando um mesmo caminho na rede. Portanto, o quanto antes os fluxos elefantes que ingressam em um PTT puderem ser identificados, mais rapidamente estratégias podem ser aplicadas para mitigar seus efeitos ou gerenciá-los de acordo com as políticas de tráfego, contribuindo com a qualidade de serviço global da rede.

Em resposta ao agravante dos fluxos elefantes em redes de PTT, propostas como *SDEFIX* (KNOB *et al.*, 2016) e *OpenSample* (SUH *et al.*, 2014) apresentam mecanismos para identificação de fluxos elefantes em redes de PTT que fazem uso de *SDN/OpenFlow*. Contudo, essas propostas extraem amostras dos fluxos no plano de dados e realizam o processo de análise fora dele, no plano de controle. Esse procedimento implica diretamente em atrasos no processo de identificação, uma vez que a comunicação entre *switch* e controlador introduz atraso no processamento das estatísticas no plano de controle. Tal como será apresentado na seção de avaliação, esse atraso implica fortemente no tempo de reação para mitigar os efeitos de um fluxo elefante.

Abordagens para lidar com o atraso de comunicação entre *switch* e controlador são viabilizadas a partir das redes com planos de dados programáveis, uma vez que a programabilidade dos *switches* pode favorecer que estes realizarem o processo de análise de fluxos internamente (BASAT *et al.*, 2017). Nessa perspectiva, trabalhos têm explorado os recursos de *switches* programáveis emergentes (BOSSHART *et al.*, 2014) para analisar fluxos diretamente no plano de dados (SIVARAMAN *et al.*, 2017), realizando procedimentos sobre os pacotes além do processamento tradicional de encaminhamento. A programabilidade dos *switches* permite que mecanismos sejam implementados para que os pacotes sejam analisados logo ao ingressarem na rede. A partir disso, esse trabalho apresenta dois mecanismos, que combinam predição e identificação de fluxos elefantes em redes de PTT com plano de dados programável, buscando contribuir para o gerenciamento de fluxos nesse contexto.

O primeiro mecanismo, denominado IDEAFIX, foi desenvolvido para identificar os fluxos elefantes diretamente nos *switches* realizando a contagem do volume de tráfego e a duração dos fluxos para cada pacote que ingressa na rede. Isto é, juntamente com o processo tradicional de encaminhamento, instruções são realizadas para obter as informações de volume e duração dos fluxos para cada pacote processado. Essas informações são armazenadas a partir de indexação com chaves *hash*, para identificar os fluxos individualmente. Uma tupla formada pelos endereços IP (*Internet Protocol*) de origem e destino, protocolo da camada de transporte e portas de aplicação origem e destino, pode ser utilizada para gerar uma chave de indexação.

Embora o IDEAFIX permita a identificação de fluxos elefantes em redes de PTT, é necessário que os fluxos analisados, de fato, excedam os limiares para serem classificados como elefantes. No entanto, embora inicialmente não seja perceptível, os fluxos elefantes são elefantes desde o primeiro pacote. Considerando estas observações, para diminuir tal necessidade, este trabalho também apresenta um segundo mecanismo voltado à predizer o comportamento dos fluxos em tempo de execução, usando observações históricas e, a partir do reconhecimento de padrões temporais, identificar os fluxos elefantes mesmo antes que excedam os limiares.

Para diminuir o intervalo de tempo até que os limiares de classificação sejam excedidos, uma abordagem foi desenvolvida para prever o tamanho e a duração do fluxo a partir um modelo de regressão localmente ponderada (*Locally Weighted Regression* - LWR) (CLEVELAND; DEVLIN, 1988; ELATTAR; GOULERMAS; WU, 2010), utilizando observações históricas do comportamento de fluxos anteriores. No modelo LWR, os pesos amostrais podem ser definidos com diferentes funções, contudo a função amplamente usada é a distribuição gaussiana (SCHAAL; ATKESON, 1994). Diante disso, os pesos podem ser atribuídos a partir de uma distribuição gaussiana ajustada pelo operador de rede de acordo com a janela temporal desejada. Além disso, o operador da rede pode definir um intervalo de tolerância para validar as estimativas de acordo com o intervalo de predição calculado para cada uma delas. Se o intervalo de predição for menor que a tolerância, então os valores inferidos são considerados válidos e confrontados imediatamente com limiares para caracterizar o fluxo como um elefante ou não, logo em seu início. Quando o intervalo de predição é maior que a tolerância, a inferência é invalidada e a análise prossegue com o IDEAFIX, no plano de dados, contabilizando as informações de cada pacote e comparando-as com os limiares.

Resultados experimentais mostram que o mecanismo de predição é capaz de inferir o volume e a duração de novos fluxos e reagir aos fluxos elefantes, com aproximadamente 50,29ms, utilizando até 32 amostras (comportamento de fluxos anteriores). Mesmo quando os números das amostras são significativamente grandes, ou seja, 2048 ou 4096, o mecanismo pode predizer e reagir aos fluxos elefantes em aproximadamente 126,32ms e 174,64ms, respectivamente. Esses números são muito menores que o tempo necessário para o fluxo exceder os limiares de classificação. Além disso, a acurácia do mecanismo obteve pelo menos 80% de sucesso nas predições, mesmo com tolerância conservadora (*i.e.,* 15%). Os fluxos que as predições não puderam ser validadas, tiveram seu processo de análise realizado diretamente no plano de dados, com o IDEAFIX.

O mecanismo de identificação desenvolvido em linguagem P4 (BOSSHART *et al.*, 2014), IDEAFIX, se mostrou mais eficiente em relação ao mecanismo implementado com os protocolos *OpenFlow* (MCKEOWN *et al.*, 2008b) e *sFlow* (SFLOW.ORG, 2018), baseado nos trabalhos relacionados, *SDEFIX* e *OpenSample*, que realizam amostragem de fluxos e identificação no plano de controle. Os resultados demonstram que é possível identificar e reagir aos fluxos elefantes em menos de 0,4ms (após efetivamente excederem os limiares), com apenas 21,5KB de dados de gerência inseridos na rede. Contudo, o tempo de processamento de pacotes nos *switches* P4 emulados por *software* também influencia no tempo de reação. Em *hardware switch*, espera-se que o tempo do processamento de pacotes seja na ordem de nanos-

segundos. Por fim, como a maioria das soluções fazem, o IDEAFIX permite ajustar o espaço de memória dedicado nos *switches* ao processo de identificação de acordo com a precisão desejada. Quando o espaço para o mapeamento *hash* aumenta (*e.g.,* 100% do número de fluxos), os falsos positivos e falsos negativos diminuem (*i.e.,* menor que 10% e 1%, respectivamente) e os verdadeiros positivos e negativos aumentam, atingindo até 95% da precisão.

O restante desta dissertação está organizada como segue. No Capítulo 2, são apresentados os conceitos teóricos adotados para o desenvolvimento desse trabalho, bem como os trabalhos relacionados com a proposta aqui apresentada. No Capítulo 3 são apresentadas as propostas para predizer e identificar fluxos elefantes em PTTs. A Seção 3.1, apresenta o mecanismo para a identificação de fluxos elefantes em redes de ponto de troca de tráfego com suporte a programabilidade, IDEAFIX. Já na Seção 3.2, é apresentado o mecanismo para realizar predições do comportamento de fluxos. A avaliação de desempenho é apresentada no Capítulo 4. Por fim, no Capítulo 5 são ressaltadas as principais contribuições deste trabalho, juntamente com a conclusão dos resultados experimentais, também são apresentadas as considerações finais e possíveis investigações futuras.

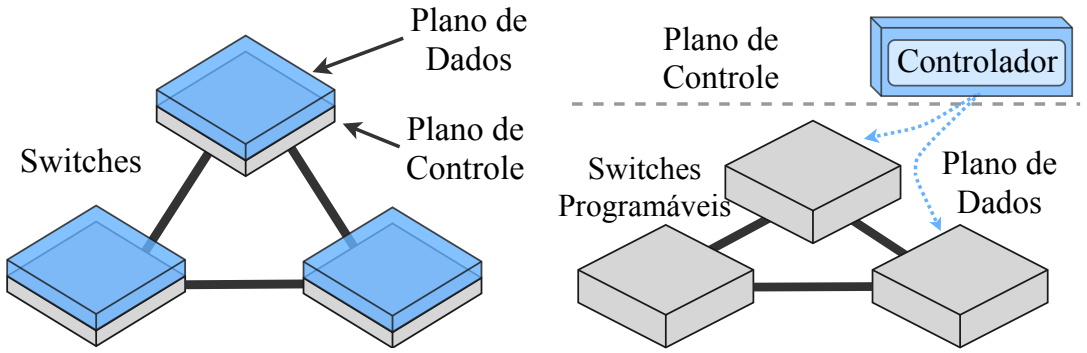
2 FUNDAMENTAÇÃO TEÓRICA E TRABALHOS RELACIONADOS

Neste capítulo, é apresentado o referencial teórico que serviu de fundamentação para o desenvolvimento desse trabalho. Na Seção 2.1, é apresentado o conceito de redes definidas por *software* tradicionais e com suporte à programabilidade. Na Seção 2.2, é brevemente destacado o conceito de fluxos elefantes e os desafios de gerência a eles associados. Na Seção 2.3, são apresentadas as características básicas e o comportamento de tráfego em redes de ponto de troca de tráfego, observando a correlação temporal existente no tráfego agregado. Na Seção 3.2.2, é apresentado o método estatístico utilizado como base no mecanismo para previsão de fluxos elefantes. Finalmente, na Seção 2.4 são apresentados os trabalhos relacionados.

2.1 Redes Definidas por *Software* - SDN

Redes Definidas por *Software* (do inglês, *Software-Defined Networking* - SDN) é uma arquitetura de rede dinâmica, que busca auxiliar e acelerar a implementação de novos serviços e responder rapidamente às mudanças na demanda de requisitos (MCKEOWN *et al.*, 2008a; XIA *et al.*, 2015). Este paradigma tem contribuído fortemente para a evolução das redes de computadores e muitas investigações têm sido realizadas sobre os possíveis benefícios de sua adoção. A Figura 2.1 abstrai a principal diferença entre as redes tradicionais e SDN.

Figura 2.1 – Arquitetura de redes tradicionais e SDN.



(a) Redes tradicionais (b) Redes Definidas por *Software*

Fonte: adaptado (DUNGAY, 2019).

SDN apresenta uma arquitetura na qual existe uma separação entre o plano de controle e o plano de dados, diferente das arquiteturas de redes tradicionais, com *switches* e roteadores (MCKEOWN *et al.*, 2008b), tal como ilustrado na Figura 2.1. Nesta separação, a tomada de decisões sobre os fluxos, bem como a atualização das tabelas de encaminhamento, fica a cargo do plano de controle, enquanto que o plano de dados administra a comutação dos pacotes

pela rede, com a premissa básica de encaminhá-los pela infraestrutura da rede (FEAMSTER; REXFORD; ZEGURA, 2014).

Em redes tradicionais, o plano de controle é executado individualmente em cada dispositivo de rede. Isso limita a visão global da rede, tornando as tomadas de decisões colaborativas entre os dispositivos, como por exemplo, para o processo de roteamento (KNOB *et al.*, 2016). Neste cenário, os dispositivos de redes colaboram para convergirem no processo de descoberta e configuração, utilizando protocolos de rede como OSPF (MOY, 1997). Em SDN, o plano de controle torna-se centralizado, permitindo que a lógica de tomada de decisão seja movida dos dispositivos para uma única entidade, chamada de controlador (ARAUJO *et al.*, 2017).

A centralização do plano de controle em SDN, além de tornar os dispositivos externos simples comutadores de pacotes, incrementa a programabilidade da rede. Assim, através de uma interface aberta como o protocolo *OpenFlow* (MCKEOWN *et al.*, 2008b), o controlador torna-se ciente de todos os elementos da rede e suas características, bem como é capaz de interagir diretamente com os *switches*. Por ter uma visão global da rede, o operador da rede pode estabelecer políticas de tráfego que se materializem em regras de encaminhamento para serem instaladas pelo controlador diretamente nos *switches*.

Além disso, mais recentemente, um novo paradigma culminou no avanço das redes SDN, até então, tradicionais. Estas são as redes com plano de dados programáveis. Nessas arquiteturas, os dispositivos de rede possuem suporte para executar códigos não primários e, consequentemente, podem ser programados para realizar procedimentos sobre os pacotes além do processamento tradicional de encaminhamento (BOSSHART *et al.*, 2014).

A programabilidade no plano de dados permite aos operadores de rede especificar - através de linguagens como P4 (BOSSHART *et al.*, 2014) - como os *switches* de uma rede devem analisar cabeçalhos (padronizados ou personalizados) e processar seus pacotes (MARQUES; GASPARY, 2018). Esse nível de flexibilidade desacopla o desenvolvimento e implantação de protocolos inovadores do projeto de *hardware* de *switches* (JEYAKUMAR *et al.*, 2014), permitindo maior agilidade no oferecimento de novos serviços.

Isso habilita que novas abordagens e mecanismos para realização de análise de fluxos (SIVARAMAN *et al.*, 2017; BASAT *et al.*, 2017) e estratégias de telemetria (MARQUES; GASPARY, 2018) sejam implementados diretamente nos *switches*, diminuindo, por exemplo, o atraso e o tráfego dados adicional na rede, gerados pela comunicação entre plano de dados e plano de controle. Conforme será discutido na Seção 3.1, esse trabalho explora os recursos disponíveis na linguagem P4 para implementar um mecanismo de análise de fluxos para identificar os fluxos elefantes diretamente no plano de dados programável.

2.2 Fluxos Elefantes

Na Internet, a maioria dos fluxos tem um tamanho e/ou tempo de vida pequeno (*i.e., mice flows or small flows*) (CURTIS; KIM; YALAGANDULA, 2011; ZHANG *et al.*, 2002), embora haja um número menor de fluxos que representa a maior parte do tráfego, tendo também uma vida útil mais longa; estes são os fluxos elefantes (do inglês, *elephant flows*) (GUO; MATTA, 2001; FANG; PETERSON, 1999). A Tabela 2.1, ilustra a classificação dos fluxos com base em seu volume de tráfego e duração. Nela é possível observar que os chamados *mice/small flows* e os *elephant flows*, se contrapõem nas extremidades das características.

Tabela 2.1 – Classes de fluxos por tamanho e duração.

Duração/Volume	**Pequeno Volume**	**Grande Volume**
Curta Duração	*Mice Flows*	Não definido
Longa Duração	Não definido	*Elephant Flows*

Fonte: do autor (2019).

A presença de fluxos elefantes em redes de computadores é comum e, como são fluxos com um longo tempo de vida e grande tamanho de tráfego, podem causar problemas de desempenho, que exigem dos operadores de rede ações de gerenciamento adequadas (KNOB *et al.*, 2017). Por exemplo, os fluxos elefantes podem afetar o desempenho de fluxos menores que ocasionalmente compartilham o mesmo caminho na rede. Além disso, eles podem exaurir recursos de memória/processamento dos dispositivos de rede, podendo ocasionar atrasos, enfileiramento e até perdas de pacotes indesejáveis (MORI *et al.*, 2004b; GUO; MATTA, 2001).

No caso das redes de PTTs, o problema passa a ser ainda mais crítico, devido à quantidade de tráfego com que estas redes devem lidar. Além disso, mitigar a influência negativa dos fluxos elefantes 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 (KNOB *et al.*, 2016). Assim, quanto mais rápido os fluxos elefantes forem identificados, mais rapidamente seus efeitos poderão ser mitigados.

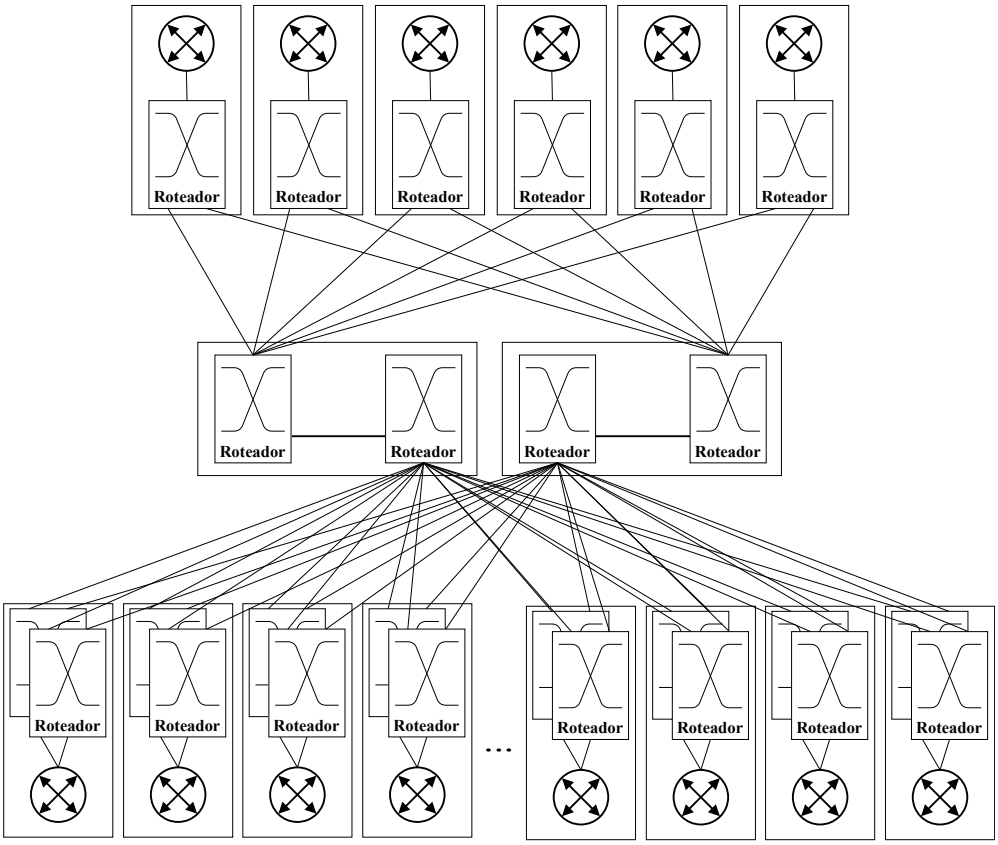
A primeira etapa para gerenciar os fluxos elefantes em uma rede de PTT é realizar a sua identificação. Uma vez que estes fluxos são identificados, o operador da rede pode realizar/definir ações apropriadas que otimizem a utilização dos recursos de rede e que diminuam o potencial negativo do impacto dos fluxos elefantes sobre a infraestrutura do PTT (KNOB *et al.*, 2017). Por exemplo, os fluxos elefantes podem ser isolados em outros caminhos disponíveis na rede,

disponibilizando o caminho padrão para os fluxos pequenos. Utilizar estratégias que definem rotas diferentes dos caminhos tradicionais ou diferenciar os fluxos por diferentes esquemas de qualidade de serviço (do inglês, *Quality of Service* - QoS) (AFAQ; REHMAN; SONG, 2015), também são alternativas para mitigar os efeitos dos fluxos elefantes (GUO; MATTA, 2001).

2.3 Pontos de Troca de Tráfego

Pontos de Troca de Tráfego (PTTs) conectam ASes e permitem que os provedores de serviços realizem troca de tráfego com a finalidade de melhor atender seus clientes (KNOB *et al.*, 2016). Os PTTs são distribuídos por todo o mundo e desempenham um papel essencial no ecossistema da internet, contabilizando pelo menos 20% de todo o tráfego trocado entre ASes (RESTREPO; STANOJEVIC, 2012). A infraestrutura de um PTT pode ser caracterizada como: simples, quando composta por um único dispositivo de comutação, ou complexa, quando formada por várias redes, com topologias intricadas (AUGUSTIN; KRISHNAMURTHY; WILLINGER, 2009). A Figura 2.2 ilustra a infraestrutura do *Amsterdam Internet Exchange* (AMS-IX), um dos maiores PTTs do mundo (AMS-IX, 2018).

Figura 2.2 – Infraestrutura do *Amsterdam Internet Exchange*.



Fonte: adaptado (AMS-IX, 2019).

O AMS-IX é um ponto de troca de tráfego distribuído, atualmente presente em múltiplas instalações independentes em Amsterdã, na Holanda. Cada instância é equipada com um ou mais dispositivos de acesso para permitir conexões à infraestrutura do AMS-IX. A implementação da plataforma de *peering* no AMS-IX usa uma infraestrutura MPLS/VPLS. Essa configuração permite uma infraestrutura resiliente e altamente escalável inerente ao MPLS, enquanto, ao mesmo tempo, a interface com os membros e clientes ainda é a plataforma comum compartilhada da Camada 2, *Ethernet* (METCALFE; BOGGS, 1976).

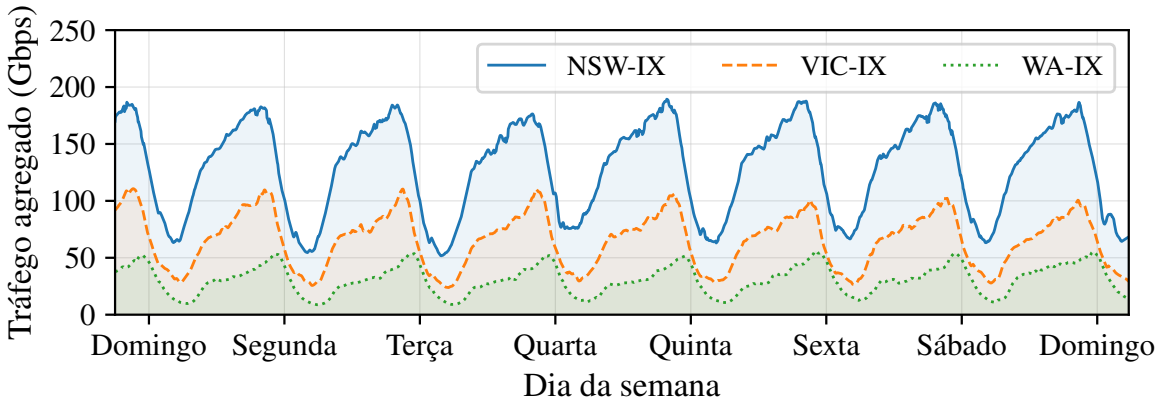
Embora estas infraestruturas pareçam complexas, um PTT pode ser resumido como um ponto centralizador de conexões entre diversas redes, sendo normalmente implementado através de soluções baseadas em tecnologia *Ethernet* (Camada 2) (KNOB *et al.*, 2016; CHATZIS *et al.*, 2013). Alguns dos principais benefícios experimentados pelos participantes de um PTT é o baixo custo de implantação e manutenção (GREGORI *et al.*, 2011). Esses fatores estimulam empresas (como *Google* e *Netflix*) a conectarem suas infraestruturas aos principais PTTs, buscando assim reduzir os custos de conexão dos seus clientes (KNOB *et al.*, 2017). Uma vez conectados fisicamente ao PTT, os membros devem promover alguma forma de *peering* com os outros ASes.

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 (KNOB *et al.*, 2016). Já os PTTs comunitários, ou "sem fins lucrativos", são administrados pelos próprios participantes ou por organizações sem fins lucrativos (CGI.BR, 2018; OPEN-IX, 2018). O modelo comunitário é tipicamente adotado pelos PTTs alocados na Europa, Africa e América Latina, enquanto que nos Estados Unidos existe domínio dos PTTs comerciais. Isto ocorre por razões comerciais (pois desde suas criações, os maiores ASes americanos possuem conexões diretas entre si), e por 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).

A Figura 2.3 apresenta o comportamento do tráfego nos PTTs da Austrália (IX-AUSTRALIA, 2018) ao longo de uma semana. O *IX-Australia* oferece *peering* com os PTT na Austrália Ocidental (*Western Australia* - WA-IX) (WA-IX, 2018), Nova Gales do Sul (*New South Wales* - NSW-IX) (NSW-IX, 2018) e *Victoria* (VIC-IX) (VIC-IX, 2018) (IX-AUSTRALIA-PEERING, 2018). Embora o comportamento expresso na Figura 2.3 seja o comportamento de tráfego agregado, é possível observar que existe um padrão periódico, com variações discretas, repetindo-se diariamente. Comportamentos de tráfego parecidos também

são observados no *Amsterdam Internet Exchange* (AMS-IX, 2018), *South African-IX* (AF-IX, 2018), *Asia Pacific-IX* (APIX, 2018), assim como em mais de 30 PTTs no Brasil (CGI.BR, 2018). Naturalmente, outros PTTs também experimentam o mesmo comportamento de tráfego.

Figura 2.3 – Tráfego agregado dos PTTs na Austrália.



Fonte: do autor (2019).

Mesmo que não seja possível identificar explicitamente os fluxos elefantes na Figura 2.3, em alguns casos, observa-se que os fluxos elefantes podem contribuir com mais de 59% do volume total de tráfego na rede (MORI *et al.*, 2004a; CURTIS; KIM; YALAGANDULA, 2011). Além disso, o tráfego de fluxos elefantes tem uma correlação substancial, mas não perfeita, com o tráfego total de fluxos (LI *et al.*, 2016). Considerando que existe uma periodicidade no tráfego de um PTT (conforme mostrado na Figura 2.3), é possível prever o tamanho e a duração de novas instâncias de fluxos observando o comportamento temporal dos fluxos anteriores.

Em outras palavras, como o padrão de tráfego exibe uma periodicidade, os eventos que anteriormente resultaram em fluxos elefantes, mais tarde, provavelmente, ocorrerão novamente. Isto é, dado que um par de *hosts* ou sub-redes realiza uma grande troca de tráfego por um longo período (*i.e.,* fluxo elefante) diariamente ou semanalmente, é razoável que fluxos começando com características semelhantes, também repitam esse padrão e repliquem o comportamento dos fluxos elefantes anteriores.

Ao utilizar o histórico de fluxos anteriores para prever o volume e a duração de novos fluxos, é necessário considerar que, em tal histórico, observações mais recentes devem possuir pesos mais significantes em relação às observações mais antigas, no cálculo de predição. Isso ocorre pela lógica intuitiva de que os novos fluxos têm maior probabilidade de repetir o comportamento de instâncias mais jovens, em relação às antigas. Ao fazer isso, é possível acompanhar o comportamento dos fluxos na rede PTT em tempo de execução e prever o comportamento dos novos fluxos de acordo com a periodicidade de eventos passados, como detalhado na Seção 3.2.

2.4 Trabalhos Relacionados

Nesta seção são apresentados os trabalhos relacionados, realizando uma breve discussão sobre suas abordagens e estratégias, bem como as limitações em relação à proposta deste trabalho. Primeiro é destacado o uso de SDN em rede de PTT para realizar identificação de fluxos elefantes. Na sequencia, são apresentados os trabalhos alinhados a proposta dessa dissertação que realizam análise de fluxos em rede com plano de dados programáveis. Finalmente, na Subseção 2.4.3, são apresentados os trabalhos voltados à predição de fluxos de rede.

2.4.1 Utilização de SDN em Redes de PTT para identificação de Fluxos Elefantes

A utilização de redes definidas por *software* (SDN) em pontos de troca de tráfego (PTTs) é apresentada em SDX (GUPTA *et al.*, 2015) como proposta aos problemas de redes tradicionais, como: limitações do protocolo BGP (*Border Gateway Protocol*) (REKHTER; LI; HARES, 2005), limitações de políticas e seleção de rotas para escoamento de tráfego. Em SDX, os ASs participantes executam aplicações SDN em um controlador virtual e as políticas geradas são combinadas em políticas de escoamento e implantadas na infraestrutura do PTT por um controlador centralizado. Contudo, ainda que permita realizar balanceamento de carga, esta abordagem não lida explicitamente com o problema dos fluxos críticos na rede. Outros trabalhos, porém, apresentam mecanismos para detectar e mitigar fluxos elefantes utilizando SDN em redes de PTT, como descrito a seguir.

Em *DevoFlow* (CURTIS *et al.*, 2011), o protocolo *OpenFlow* (MCKEOWN *et al.*, 2008b) é usado para analisar fluxos elefantes com diferentes mecanismos de monitoramento, como amostragem de pacotes e estatísticas de portas dos dispositivos de rede. *DevoFlow* implementa modificações nos *switches OpenFlow* adicionando novos recursos e contadores ao *hardware* para facilitar a monitoração de fluxos na rede. Isso tem o objetivo de otimizar o tempo de detecção dos fluxos elefantes e diminuir o número de consultas ao controlador. Quando um limiar é excedido, em termos de contagem de bytes, o fluxo é classificado como um elefante e é roteado para um caminho menos congestionado entre seus pontos finais.

Em *OpenSample* (SUH *et al.*, 2014), *sFlow* (SFLOW.ORG, 2018) é usado para realizar amostragem de fluxo em uma solução para gerenciar redes SDN. As taxas de fluxo são calculadas subtraindo números de sequência TCP (*Transmission Control Protocol*) (POSTEL, 1981) de duas amostras do mesmo fluxo e dividindo o valor pelo tempo decorrido entre elas. No entanto, com *OpenSample* as estatísticas dos *switches* não são consideradas e os fluxos pre-

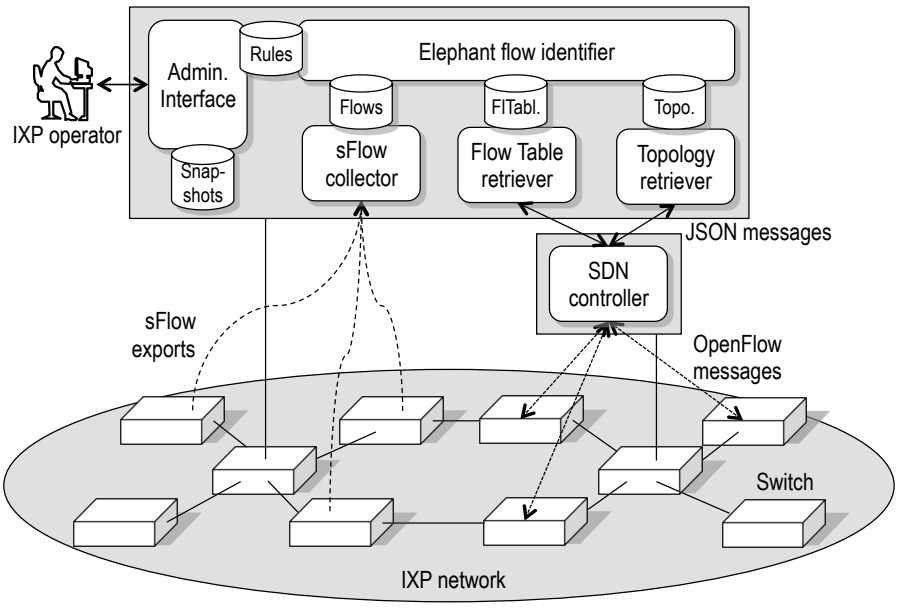
cisam ser amostrados e enviados ao plano de controle para serem processados. Isso atrasa a detecção de fluxos críticos e adiciona uma quantidade significativa de dados de amostragem na rede. Finalmente, quando um fluxo é classificado como um elefante, ele é redirecionado pelo controlador SDN para outro caminho na infraestrutura da rede.

Outras abordagens demonstram o uso de algoritmos de aprendizagem de máquina para a identificação de fluxos elefantes em uma rede OpenFlow (XIAO *et al.*, 2015; AFEK *et al.*, 2015). Utilizando algoritmos de decisão baseado em árvores, a classificação ocorre com a captura dos fluxos por um analisador de tráfego como o *Wireshark* ou *tcpdump* (COMBS, 2019).

Em SDEFIX (KNOB *et al.*, 2016), a amostragem de pacotes também ocorre utilizando *sFlow* e um módulo de identificação é alimentado para realizar a classificação dos fluxos baseado em regras predefinidas pelo operador da rede. Essa proposta realiza a identificação dos fluxos elefantes no plano de controle, a partir de amostragem de fluxos obtidas no plano de dados. Esse fator, além de implicar em atraso no processo de análise, também incorre em um volume adicional de dados de monitoração sobre a rede.

A Figura 2.4 ilustra a arquitetura do sistema proposto em SDEFIX. É possível observar que a solução envolve um módulo de análise (*Elephant flow identifer*) que consome estatísticas obtidas do plano de dados por um coletor *sFlow* para realizar o processo de identificação. Essa abordagem, semelhante às descritas anteriormente, dependem exclusivamente do envolvimento do plano de controle no processo de análise e detecção. Isso incorre, entre outros aspectos, em atrasos devido a comunicação *switch*-controlador.

Figura 2.4 – Arquitetura SDEFIX.



Fonte: (KNOB *et al.*, 2017).

Em DTL (LIU *et al.*, 2017), é apresentada uma abordagem adaptativa para detecção de fluxos elefantes, adotando o algoritmo de aprendizado de tráfego dinâmico (DTL). Primeiro, o controlador envia um conjunto de consultas (ou seja, solicitações estatísticas de tráfego) para obter as informações de tráfego nos contadores estatísticos dos switches. Em seguida, analisando a relação entre características de tráfego dos fluxos elefantes, o sistema aprende dinamicamente sobre a mudança de tráfego e define os limiares com o algoritmo DTL. Depois, o controlador configura o valor do novo limiar estimado para os *switches*, para que a classificação dos fluxos elefantes ocorra em tempo real e de forma adaptativa. Contudo, essa abordagem é aplicada no contexto de redes de *datacenter*, além de ser implementada com tradicional SDN/OpenFlow e não desenvolvida diretamente no plano de dados programável.

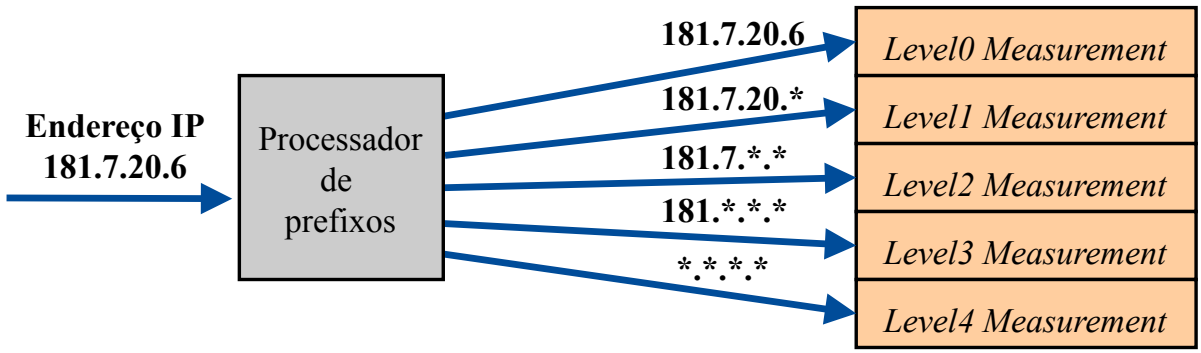
Em contraste à essas abordagens, uma das propostas apresentadas neste trabalho, denominada IDEAFIX (ver Capítulo 3.1), realiza todo o processo de análise e identificação dos fluxos elefantes diretamente no plano de dados programável. Esse mecanismo realiza o processo de identificação a partir de procedimentos implementados diretamente nos *switches* programáveis. Desse modo, buscá-se identificar e reagir aos fluxos elefantes de forma mais rápida e eficiente que as abordagens que envolvem o plano de controle no processo de identificação.

2.4.2 Monitoramento de Fluxos no Plano de Dados Programável

O monitoramento de fluxos diretamente no plano de dados programável é apresentado em *HashPipe* (SIVARAMAN *et al.*, 2017), onde esse processo é realizado em um *backbone* com taxa de transmissão superior à 100Gbps. O objetivo desse trabalho é realizar a identificação de *heavy hitters* inteiramente no plano de dados, utilizando *switches* P4 (BOSSHART *et al.*, 2014). *Heavy Hitters* podem ser considerados todos os k fluxos maiores seja em quantidade de bytes ou número de pacotes em relação aos demais fluxos em um *link* ou porta de um *switch*, em um determinado instante (SIVARAMAN *et al.*, 2017).

Em *HHH* (BASAT *et al.*, 2017), é apresentada uma proposta que realiza uma análise semelhante para identificar *heavy hitters*, usando uma estratégia de agregação de endereços IP (*Internet Protocol*), a partir de sua hierarquia, como mostra a Figura 2.5. De forma semelhante ao *HashPipe*, os fluxos são mapeados por funções *hash* para serem identificados e analisados a partir de sua chave. Os endereços IP de origem e destino, protocolo da camada de transporte e portas de aplicação de origem e destino formam uma tupla que é mapeada para um número de identificação do fluxo. A análise do fluxo é realizada contabilizando o volume de tráfego em registradores P4, indexados a partir desta chave.

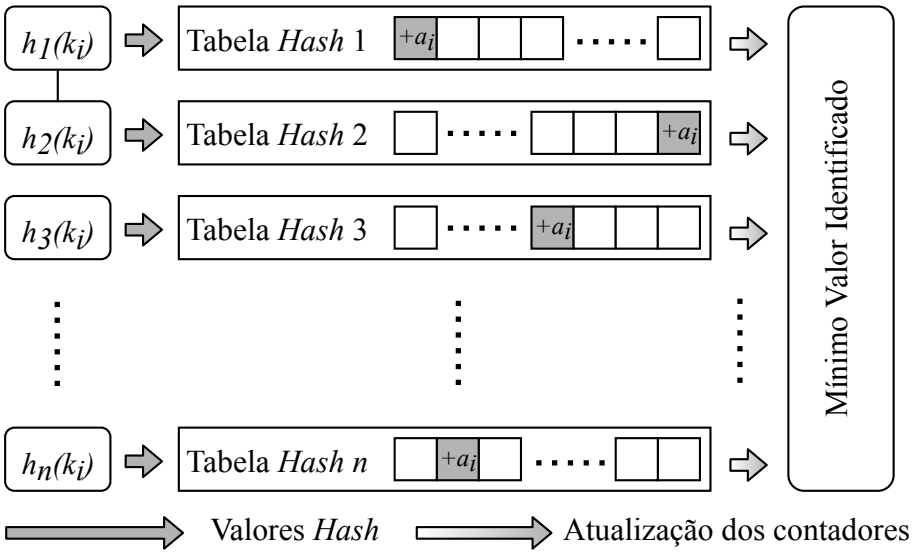
Figura 2.5 – Agregação hierárquica de endereços IP em *HHH*.



Fonte: adaptado (BASAT *et al.*, 2017).

No mapeamento *hash*, no entanto, podem ocorrer colisões de fluxos distintos para uma mesma chave. Esse fator pode incidir em falsos positivos, quando um fluxo é dito ser elefante sem de fato sê-lo, ou em falsos negativo, quando um fluxo é de fato elefante e não é identificado. Diante disso, é utilizado um mecanismo que permite realizar o mapeamento das chaves em vários níveis (TONG; PRASANNA, 2015; LAI *et al.*, 2015). Assim, a mesma tupla (k_i) é mapeada em diferentes funções *hash* (h_n) e os dados (a_i) são armazenados em diferentes posições em diferentes tabelas, para diminuir a probabilidade de caracterização indevida, como ilustra a Figura 2.6.

Figura 2.6 – Mapeamento *hash* multinível.



Fonte: adaptado (TONG; PRASANNA, 2015).

Um exemplo prático da utilização do mecanismo de mapeamento hash em vários níveis ocorre quando as chaves de um mesmo fluxo indexam valores de volume de tráfego diferentes. Isso indica que pelo menos uma das chaves colidiu com o mapeamento de outro fluxo, de modo

que seus contadores estão sendo acumulados. Nesse caso, a chave que deve ser utilizada para recuperar o melhor valor estimado para analisar o fluxo deve ser aquela que indexa o contador com o menor valor. Isso ocorre porque, no melhor caso, quando há colisões, o valor mínimo contabilizado é o valor real do fluxo analisado.

Em NHH (HARRISON *et al.*, 2018), é presentado um mecanismo para detectar *heavy hitters* em redes de longo alcance utilizando limiares adaptativos. NHH implementa um método de identificação nos *switches* de borda para contabilizar as informações de volumes dos fluxos e analisar o comportamento desses fluxos, segundo limiares locais. Cada *switch* identifica os *heavy hitters* a serem reportados ao controlador usando diferentes limiares locais, para diferentes chaves de monitoramentos, isto é, chaves *hash*.

Quando um fluxo excede os limiares para sua chave, em NHH, uma notificação é enviada ao plano de controle para reportar este fluxo. No plano de controle, o controlador combina as sinalizações recebidas dos *switches* para agregar estatísticas e identificar os *heavy hitters* dentre os fluxos reportados. A classificação ocorre de forma combinada com o plano de controle para evitar tomadas de decisões com parâmetros locais de um *switch*. Ou seja, o controlador é responsável por filtrar e julgar os fluxos reportados do plano de dados.

Além disso, o controlador realiza consultas seletivas aos *switches* para contagens adicionais e, a partir do comportamento global da rede, atualiza os limiares locais. Essa estratégia é utilizada para reduzir sobrecargas de notificações ao plano de controle geradas por limiares mal configurados. Isso contorna a limitação relacionada à visão local que os *switches* possuem sobre o estado da rede. Pois, uma vez que o controlador possui uma visão global, os limiares podem ser configurados localmente segundo os aspectos globais da rede.

Em suma, essas estratégias habilitaram a análise e monitoramento de fluxos diretamente no plano de dados programável. Contudo, não é apresentado um mecanismo voltado aos aspectos de fluxos elefantes, que envolvem tanto as características de volume quanto a duração dos fluxos. Ainda, estas propostas não são apresentadas em um contexto de redes de PTTs, onde o tempo para a identificação desses fluxos pode comprometer o bom desempenho dos serviços prestados pela rede.

Neste sentido, como será descrito na sequência, uma das propostas desse trabalho consiste em realizar o processo de identificação dos fluxos elefantes diretamente no plano de dados programável de uma rede de PTT, utilizando os recursos encontrados em *switches P4*. O mecanismo, chamado IDEAFIX, realiza todo o processo de análise e identificação dos fluxos elefantes diretamente no plano de dados. Com isso, deseja-se contribuir para o melhor gerenciamento de tráfego no contexto dessas infraestruturas.

2.4.3 Mecanismos de Predição de Fluxos

Nessa Seção são apresentados e discutidos alguns trabalhos que realizam predição do comportamento de tráfego em redes de computadores, utilizando reconhecimento de padrões. Os mecanismos de predição baseados em reconhecimento de padrões usam informações quantitativas. O termo "quantitativo" refere-se a um tipo de informação com base em dados quantificáveis (propriedades objetivas) (DALMAZO; VILELA; CURADO, 2017).

Neste sentido, as técnicas de Redes Neurais Artificiais (RNA) (CORTEZ *et al.*, 2006), Redes *Bayesianas* (DALMAZO *et al.*, 2011), Cadeias de *Markov* escondidas (DAINOTTI *et al.*, 2008) e *Machine Learning* (ERMAN; MAHANTI; ARLITT, 2006) têm recebido crescente atenção no campo dos modelos de previsão. Reconhecimento de Padrões é uma tentativa de modelar o cérebro humano, o que significa que o reconhecimento de padrões envolve basicamente o aprendizado a partir da experiência. Além disso, o modelo deve ser capaz de manter uma boa precisão, o que não é simples em pouco tempo, uma vez que a precisão dessas técnicas depende da disponibilidade de dados históricos suficientes (DALMAZO; VILELA; CURADO, 2017).

Outros trabalhos usam algoritmos genéticos para construir uma rede neural de alimentação de múltiplas camadas para predição de tráfego de rede em tempo de execução (*online*) (CHEN; YANG; MENG, 2012; HONGYING; LI, 2013). Esta abordagem foi adotada para obter uma melhor compreensão das principais características dos dados de tráfego. Além disso, o método proposto é capaz de prever pequenas medições de tráfego em escala de tempo e pode reproduzir as características estatísticas de medições reais de tráfego. No entanto, para obter resultados confiáveis, é necessária uma entrada inicial que dependa das características dos dados sob avaliação. Ou seja, é necessário um período de treinamento para realizar as predições, sempre que ocorra mudanças no comportamento da rede.

Também é encontrada abordagem baseada em estatísticas para predizer e classificar o tráfego de rede, levando em conta características como tempo entre pacotes, tamanho da carga útil e sua correlação temporal (HONGYING; LI, 2013). A solução proposta envolve uma classificação do tráfego com base em um modelo de cadeia de *Markov* escondida. Contudo, esta abordagem não é apresentada no contexto de redes de PTTs, para a identificação de fluxos elefantes.

Utilizando métodos de aprendizado de máquina (do inglês, *Machine Learning*) (ERMAN; MAHANTI; ARLITT, 2006), que envolvem o desenvolvimento de sistemas com a capacidade de aprender a partir de reconhecimento de padrão anteriores, outras abordagens são empregadas para predizer o tráfego de rede. Contudo, a aplicação de *Machine Learning* carece de

várias etapas (NGUYEN; ARMITAGE, 2008). Primeiro, os recursos (*traffic features*) devem ser selecionados para alimentar o algoritmo. Depois disso, esses recursos são atribuídos a fluxos calculados em vários pacotes. O classificador de aprendizado cria regras vinculando os recursos ao comportamento de tráfego conhecido. Finalmente, o algoritmo é capaz de prever o tráfego de rede com base nas regras aprendidas anteriormente. Nesse processo, o tempo de aprendizado compromete a adaptação do modelo de predição para acompanhar mudanças no tráfego da rede em tempo de execução.

Em trabalhos anteriores, é evidenciada a utilização de técnicas de análise Bayesiana. Em Moore (MOORE; ZUEV, 2005), os fluxos são classificados com base no conteúdo de dados e são categorizados de acordo com sua aplicação de Internet. O tráfego selecionado é agrupado por aplicação, por exemplo, transferência de dados em massa, banco de dados, correio eletrônico, serviços *web*, P2P, entre outros. No entanto, essa abordagem precisa inspecionar o conteúdo dos fluxos da rede para realizar o agrupamento. De modo diferente, neste trabalho os pacotes são analisados apenas em termos das características do fluxo, como volume, duração, origem, destino, e não ao seu conteúdo.

Outro trabalho (NGUYEN; ARMITAGE, 2006), apresenta um método para predizer o comportamento de fluxos utilizando janela deslizante de classificação *"on the fly"* (BERNAILLE *et al.*, 2006). Nesse trabalho, o uso de um pequeno número de pacotes garante a pontualidade da classificação e reduz o espaço de *buffer* necessário para armazenar informações dos pacotes para o processo de classificação. A abordagem não exige que o classificador capture o início de cada fluxo e permite que a classificação seja iniciada em qualquer momento no qual os fluxos de tráfego já estejam em andamento (NGUYEN; ARMITAGE, 2008). Contudo, no contexto de redes de PTTs, como já mencionado, o quanto antes os fluxos elefantes forem identificados, mais rapidamente seus efeitos podem ser mitigados (KNOB *et al.*, 2016; GUPTA *et al.*, 2015).

Em geral, as abordagens de reconhecimento de padrões são realizadas em dois estágios: treinamento e previsão. As tarefas de treinamento e previsão devem ser executadas em diferentes escalas de tempo. Embora o treinamento do modelo seja uma operação *off-line*, ele deve ser feito periodicamente para que um rigoroso grau de precisão seja mantido, enquanto a previsão deve ser feita continuamente e online (FRANK, 1994). Se esses dois estágios se sobrepuserem, haverá um aumento na carga de trabalho, tornando-o inadequado para a predição de tráfego em tempo de execução, em ambientes dinâmicos (DALMAZO; VILELA; CURADO, 2017).

O problema de obter um modelo de predição confiável e que aprenda em tempo de execução é um dos maiores desafios encontrados nas abordagens investigadas. Diante disso, a abordagem de previsão adotada nesse trabalho utiliza um modelo baseado em regressão localmente

ponderada (LWR), que permite ao operador da rede definir um subconjunto de amostras para influenciarem localmente no modelo. Ou seja, mesmo que exista um conjunto amostral muito grande, que demandaria custos para seu processamento, o LWR é capaz de criar um modelo de inferência baseado apenas por amostras mais próximas do fluxo em análise, de acordo com a correlação temporal entre eles, conforme mostra os resultados deste trabalho, na Seção 4.4. Ainda, uma vez que não precisa de prévio treinamento, o mecanismo é capaz de aprender e acompanhar as mudanças no comportamento do tráfego da rede em tempo de execução.

3 MECANISMO DE PREDIÇÃO E IDENTIFICAÇÃO DE FLUXOS ELEFANTES

Neste capítulo são apresentados os dois mecanismos utilizados para identificar fluxos elefantes em redes de ponto de troca de tráfego com suporte a programabilidade. O primeiro, denominado IDEAFIX, é dedicado a contabilizar as características dos fluxos e identificar os fluxos elefantes diretamente no plano de dados. O segundo mecanismo, implementado no plano de controle, utiliza uma base histórica para predizer o comportamento dos novos fluxos, a partir da correlação temporal com fluxos anteriores. Quando as predições são validadas, é possível julgar e identificar um fluxo elefante antes mesmo que os limiares sejam excedidos. De modo diferente, quando as predições não são validadas, o processo de identificação ocorre diretamente nos *switches*, com o IDEAFIX. Além disso, as estatísticas dos fluxos contabilizadas com IDEAFIX alimentam a base amostral usada no mecanismo de predição.

3.1 Identificação de Fluxos Elefantes em Redes Programáveis

Com o objetivo de identificar fluxos elefantes em redes de PTT com suporte a programabilidade, nesta seção é apresentado um mecanismo denominado IDEAFIX. O mecanismo utiliza os recursos da linguagem P4 (BOSSHART *et al.*, 2014) para extrair e armazenar informações sobre o tamanho e a duração dos fluxos que ingressam na rede. Uma tupla-5 (*i.e.,* endereços IP origem e destino, portas de origem e destino, e protocolo de transporte) é usada para criar uma chave *hash* a partir de cada pacote que chega em um *switch* de borda.

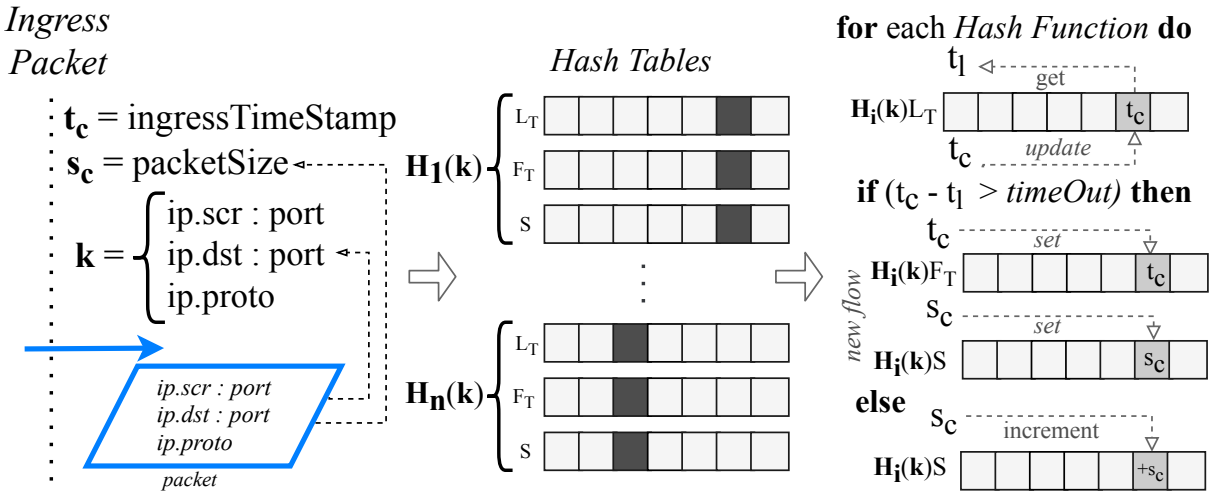
O processo de identificação implementado no IDEAFIX possui quatro etapas principais: *extração*, *consolidação*, *classificação* e *notificação*. Primeiro, quando um pacote chega em um *switch* de borda, a tupla-5 é extraída do pacote e mapeada através de uma função hash, para gerar uma chave de indexação para o fluxo. Em seguida, a etapa de consolidação identifica a qual fluxo de rede o pacote pertence, para atualizar o tamanho e a duração desse fluxo nos registradores do *switch*. A etapa de classificação compara as informações consolidadas/armazenadas aos limiares para identificar se o fluxo é um elefante ou não. Finalmente, a etapa de notificação atua para informar o plano de controle sobre os fluxos elefantes detectados.

3.1.1 Extração e Consolidação

A etapa de extração (Figura 3.1) coleta dos pacotes as informações necessárias para identificar o fluxo a qual este pertence e as características que serão contabilizadas. Quando um pacote chega ao *switch*, seu tempo de ingresso (*ingressTimeStamp*) é salvo nos metadados do pacote,

na variável local t_c, o tamanho do pacote é armazenado na variável local s_c e a identificação do fluxo (tupla-5) é armazenado na variável local k.

Figura 3.1 – Etapas de extração e consolidação.



Fonte: do autor (2018).

Como o fluxo é identificado pelo metadado k, utilizando pelo menos duas funções *hash*, é possível mapear k para indexar os registradores que armazenarão os dados do fluxo. Um registrador, por sua vez, é uma memória de estado, disponível na linguagem P4, cujos valores podem ser lidos/escritos a partir de posições indexadas. Utilizando várias funções *hash* em vez de uma única, é possível reduzir a possibilidade de perda/sobreposição de informações causada por colisões *hash* (TONG; PRASANNA, 2015).

No processo de mapeamento *hash*, para cada função *hash*, três registradores são utilizados para armazenar as informações dos fluxos, sendo eles: L_T, F_T e S. Cada *slot* no registrador L_T (48 *bits*) armazena o tempo de ingresso do último pacote de um determinado fluxo (assumindo que nenhuma colisão aconteceu). Por sua vez, cada *slot* em F_T (48 *bits*) armazena o tempo de ingresso do primeiro pacote de um fluxo. Finalmente, cada *slot* no registrador S (32 *bits*) armazena a soma acumulativa do tamanho de cada pacote de um determinado fluxo (novamente, assumindo que não há colisões). Em caso de colisão, as informações podem ser falsamente sobrescritas ou acumuladas. Não há como garantir 100% que não haverá colisões de chaves *hash*, porém a abordagem de utilizar múltiplas *hashs* reduz as chances de colisões (TONG; PRASANNA, 2015).

Para mitigar o impacto de colisões *hash*, primeiro é verificado se o pacote pertence a um fluxo já existente/ativo ou se é um novo fluxo iniciando. Isso é feito calculando o intervalo de tempo entre a chegada do último pacote e o tempo de ingresso do pacote atual ($t_c - t_l$) e com-

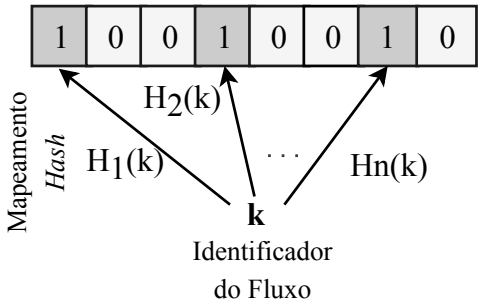
parando a diferença com um tempo limite (*timeout*) pré-definido (Figura 3.1). Se o *timeout* for excedido, então o pacote representa um novo fluxo e os registradores precisam ser reiniciados, passando a contabilizar as informações do pacote atual. Em seguida, F_T e S são atualizados para os valores do tempo de ingresso t_c e o tamanho s_c do pacote atual, respectivamente. Se o *timeout* não for excedido, significa que, ou o pacote pertence a um fluxo já ativo, ou ocorreu uma colisão de chaves *hash*. Em ambos os casos, apenas o tamanho s_c é incrementado no registrador S. Finalmente, em todas as situações, o registrador L_T é atualizado para o tempo de ingresso t_c do pacote atual. Isso permite superestimar os fluxos, a fim de evitar falsos negativos.

É importante ressaltar que essa abordagem de atualização dos registradores pode gerar falsos positivos. Ou seja, quando um fluxo não elefante é caracterizado erroneamente como sendo um elefante. Todavia, outra estratégia de atualização dos registradores impulsionaria a ocorrência de falsos negativos. Isto é, quando um fluxo que, de fato, é um elefante não é identificado. Diante disso, considerando os impactos que os fluxos elefantes podem causar na rede, é preferível tratar com cautela fluxos que não são elefantes e que tenham sido erroneamente classificados o sendo (isto é, falsos positivos), do que admitir falsos negativos. Contudo, o mecanismo apresentado pode ser adaptado segundo às necessidades do operador da rede.

3.1.2 Classificação e Notificação

O objetivo dessa etapa é caracterizar um fluxo como elefante ou não. O primeiro passo da análise é verificar se o fluxo referente ao pacote já foi previamente identificado como um elefante. Nesse sentido, a abordagem utiliza um mecanismo baseado em *bloom filter* (GERAVAND; AHMADI, 2013) para marcar quais fluxos já foram identificados como elefantes. Inicialmente, todos os *slots* de memória no *bloom filter* são configurados para valor igual à zero. Quando um fluxo é identificado como elefante, um número fixo de *slots* apontados pelas funções *hash* são setados para valor igual à um, tal como ilustrado na Figura 3.2.

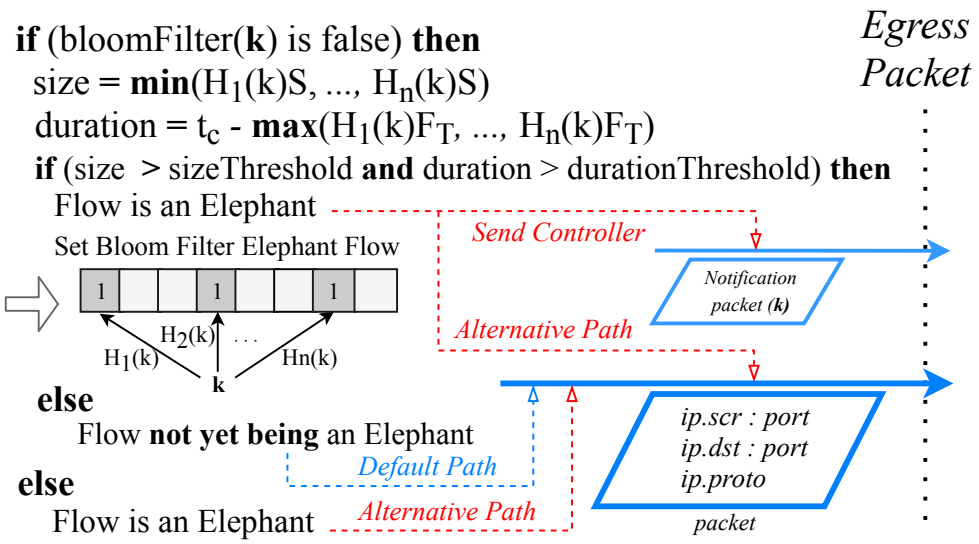
Figura 3.2 – Sinalização com *bloom filter*.



Fonte: do autor (2018).

Se o fluxo não tiver sido identificado como um elefante anteriormente ($bloom\_filter = false$), seu tamanho e duração serão estimados considerando todas as funções *hash*. Para o tamanho (*size*, na Figura 3.3), uma estratégia de *count-min* ($min(H_1(k)S, ..., H_n(k)S)$) é suficiente para determinar o tamanho mais próximo do tamanho real do fluxo. Isso ocorre porque o valor mínimo está contido no *slot* com o menor número de colisões. A duração do fluxo (*time*, na Figura 3.3), é estimada pela diferença entre o tempo do pacote atual t_c e o tempo de ingresso do primeiro pacote desse fluxo, que é está armazenado nos registradores F_T. Nesse caso, em contraste com a estimativa do tamanho do fluxo, o valor máximo registrado entre todos os registradores ($max(H_1(k)F_T, ..., H_n(k)F_T$) é usado como uma aproximação do tempo de ingresso do primeiro pacote.

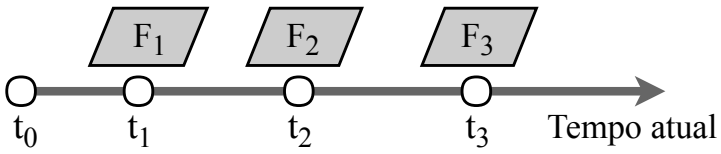
Figura 3.3 – Etapas de classificação e notificação.



Fonte: do autor (2018).

Como um exemplo do processo mencionado acima, considere que um fluxo F_3, cujo primeiro pacote chegou ao *switch* no tempo t_3 (Figura 3.4), teve colisões em todas suas chaves *hash*, com os fluxos ativos F_1 e F_2, cujos primeiros pacotes chegaram ao *switch* nos tempos t_1 e t_2, respectivamente. A estratégia de atualização dos registradores, descritos na Subseção 3.1.1, prevê que estes não serão reiniciados quando ocorrer uma colisão de chave *hash*. Portanto, F_3 assumirá as informações de tempo inicial já armazenadas para F_1 e F_2. Ao recuperar o valor máximo entre os registradores F_T, nas posições indexadas pelas chaves *hash*, é possível obter o tempo mais próximo do instante inicial do fluxo F_3. Neste exemplo, esse seria o tempo de ingresso do primeiro pacote do fluxo F_2 (t_2), pois, este é maior que o tempo de ingresso do primeiro pacote de F_1 (t_1) e, é o tempo mais próximo tempo inicial de F_3 (t_3).

Figura 3.4 – Simulação para colisão.



Fonte: do autor (2019).

Em seguida, o tamanho e a duração do fluxo são comparadas com aos limiares (*sizeThreshold* e *durationThreshold*), predefinidos pelo operador da rede. Se o tamanho e a duração não excederem os limiares, o fluxo será classificado como não sendo um elefante e o pacote será encaminhado, normalmente, por seu caminho padrão na rede. De modo diferente, se os limiares forem excedidos, então o fluxo é classificado como um elefante e uma notificação é enviada ao controlador SDN para reportar a identificação. Além disso, os pacotes deste fluxo serão encaminhados através de um caminho alternativo, como estratégia para mitigar seus efeitos sobre os demais fluxos. Várias políticas de mitigação de fluxos elefantes podem ser aplicadas; porém, como não é o foco deste trabalho gerenciar os fluxos elefantes, foi adotada uma estratégia que consiste em isolar os fluxos elefantes, encaminhando-os por caminhos alternativos na rede, baseada nos trabalhos relacionados (KNOB *et al.*, 2017; CURTIS *et al.*, 2011).

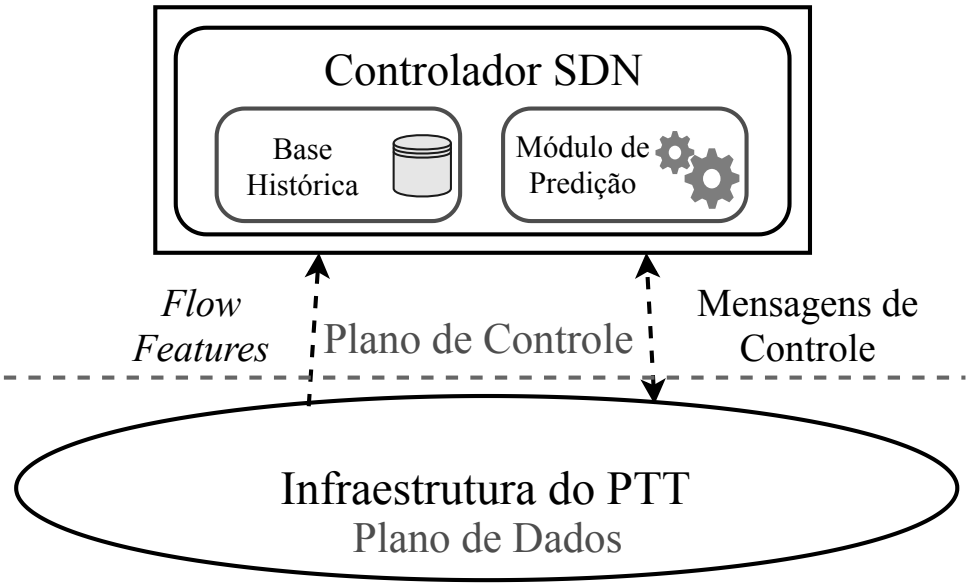
Por fim, como já mencionado, as estatísticas contabilizadas para cada fluxo são enviadas ao plano de controle para alimentar a base amostral utilizada pelo mecanismo de predição, que será descrito na sequência. Ou seja, quando um fluxo é finalizado (*TCP FIN FLAG* ou estouro de *timeout*), o *switch* recupera as informações armazenadas para quele fluxo e as envia ao plano de controle para serem inseridas na base amostral. No entanto, é importante ressaltar que a base amostral pode ser alimentada com outro mecanismo, por exemplo, utilizando um coletor *sFlow*. Porém, como demostrado no Capítulo 4, o IDEAFIX reduz significativamente o tráfego adicionado na rede com o processo de monitoração, quando comparado com uma abordagem que realiza amostragem de fluxos utilizando, por exemplo, *sFLow*.

3.2 Mecanismo de Predição

Nesta seção é descrito o mecanismo para predizer e identificar fluxos elefantes em redes de PTT, utilizando observações históricas e correlação temporal. A Figura 3.5 mostra a arquitetura proposta. É ilustrada uma infraestrutura de rede de PTT com um plano de dados programável, um banco de dados histórico e um módulo de predição, no plano de controle, conectados ao controlador da rede.

A interação entre plano de dados e plano de controle é realizada pelo controlador SDN, o qual recebe dos *switches* as estatísticas contabilizadas sobre fluxos e envia mensagens de controle para configurar o tráfego na rede, ou por exemplo, mitigar um fluxo identificado como elefante.

Figura 3.5 – Arquitetura do mecanismo de predição.



Fonte: do autor (2019).

O fluxo de execução do mecanismo inicia com o ingresso do primeiro pacote do novo fluxo, no plano de dados, e segue com o processo de inferência, no plano de controle. Quando um switch de borda recebe o primeiro pacote de um fluxo, este envia ao plano de controle uma notificação para calcular o caminho pelo qual o fluxo será roteado na rede do PTT. Em seguida, ainda no plano de controle, o mecanismo de predição, baseado no modelo LWR, é utilizado para predizer o tamanho e a duração do novo fluxo, seguindo o descrito na Subseção 3.2.2. Quando as predições são consideradas válidas, a partir do intervalo de predição tolerado, o fluxo é caracterizado como um elefante ou não, de acordo com os limiares predefinidos pelo operador da rede, como é discutido nas seções a seguir.

3.2.1 Base Histórica e Seleção Amostral

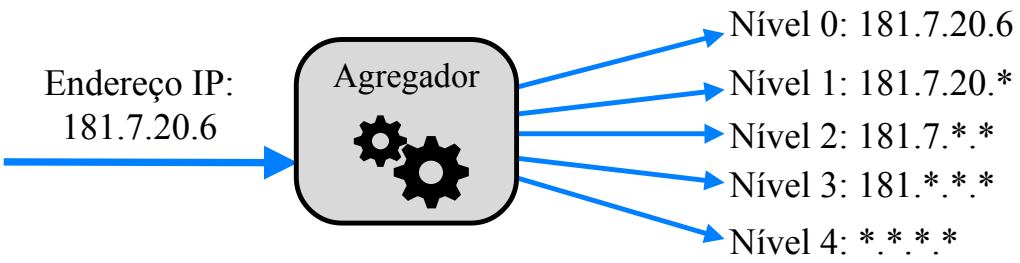
O mecanismo de predição depende de um banco de dados histórico que armazena informações sobre o comportamento de instâncias de fluxo anteriores. Essas informações são extraídas diretamente do plano de dados, no qual o mecanismo denominado IDEAFIX (Subseção 3.1) usa os recursos da linguagem P4 (BOSSHART *et al.*, 2014) para contabilizar, em cada switch de borda, o volume e a duração de cada fluxo que ingressa na rede do PTT.

Para identificar cada fluxo individualmente, uma tupla-5 (*i.e.,* endereços IP e portas de origem e destino e tipo de protocolo de transporte) é mapeada por funções *hash* para gerar as chaves de indexação dos registradores. Quando um fluxo é finalizado (*e.g., TCP FIN Flag* é válido ou estouro de *timeout*), uma notificação é enviada ao plano de controle (em um segmento UDP) para reportar o tamanho e duração do fluxo, juntamente com seu instante inicial (*i.e., ingressTimeStamp*) e a tupla-5. Dessa forma, a base amostral é alimentada com o comportamento dos fluxos que serão usados para predizer o comportamento dos novos fluxos, de acordo com a correlação temporal entre eles.

A seleção amostral é baseada na tupla-5 que define o novo fluxo. Inicialmente, todos os registros com o mesmo endereço IP de origem e destino, protocolo de transporte e pelo menos uma das portas de aplicação são selecionados. As amostras são selecionadas em uma janela temporal definida pelo operador de rede, para delimitar o tamanho da base histórica. Por exemplo, é possível definir um intervalo de amostragem em dias, semanas, meses e assim por diante.

Além disso, a seleção pode ocorrer a partir da agregação hierárquica de endereços IP, como ilustrado na Figura 3.6, baseado em HHH (BASAT *et al.*, 2017). Nessa estratégia, a seleção é dividida em níveis e a agregação ocorre a partir do prefixo do endereço IP das amostras. Isso permite administrar o conjunto amostral no processo de inferência para analisar, por exemplo, o comportamento de sub-redes e não apenas um único endereço IP. Após concluir a seleção amostral, o mecanismo executa a etapa de predição, descrita a seguir.

Figura 3.6 – Agregação hierárquica de endereço IP.



Fonte: adaptado (BASAT *et al.*, 2017).

3.2.2 Predição e Modelo LWR

Para predizer o comportamento dos fluxos de rede, foi adotado o método estatístico de regressão localmente ponderada, *Locally Weighted Regression* (LWR) (CLEVELAND; DEVLIN, 1988; SCHAAL; ATKESON, 1994). O modelo LWR permite inferir o valor de variáveis dependentes a partir de um conjunto de variáveis independentes, utilizando ponderação localizada. O LWR baseia-se no pressuposto de que os valores vizinhos do ponto desejado são os melhores indicadores para a predição, em um intervalo de amostras (CLEVELAND; DEVLIN, 1988). No contexto deste trabalho, o tamanho e a duração das instâncias de fluxo anteriores (amostras no modelo) são as variáveis dependentes no modelo. Já o *timeStamp* inicial de cada fluxo, estes são as variáveis independentes. Por fim, as predições são o tamanho e a duração do novo fluxo em seu *timeStamp* inicial (ponto desejado). Os vizinhos são as instâncias de fluxos cujo o *timeStamp* inicial está mais próximo do *timeStamp* do novo fluxo.

LWR cria uma superfície "local" para os pontos desejados usando uma regressão ponderada pela sua distância em relação às amostras no modelo (ELATTAR; GOULERMAS; WU, 2010). Este método é derivado da regressão linear padrão (JAIN, 1990), conforme mostrado na Equação 3.1. A proposta consiste em definir os parâmetros do modelo linear β_i, minimizando o quadrado dos erros entre a curva de inferência e cada amostra $(y_i - f(x_i))^2$, ponderadas localmente pelo peso w_i, definidos de acordo com a distância em relação ao ponto desejado.

$$F(\beta_0, \beta_1, ..., \beta_m) = \sum_{i=1}^{n} w_i (y_i - f(x_i))^2 \tag{3.1}$$

Após a derivação da Equação 3.1 (CLEVELAND; DEVLIN, 1988; SCHAAL; ATKESON, 1994), os parâmetros do modelo linear β_i, usados para prever os valores do ponto desejado, são obtidos pela equação normal:

$$\vec{\beta} = \left(X^T W^T W X\right)^{-1} X^T W^T W \vec{y} \tag{3.2}$$

onde X é uma matriz $n \times (m+1)$, consistindo em n pontos amostrais, cada um representado por suas dimensões de entrada m e um "1" na última coluna; y é um vetor das observações correspondentes para cada amostra; W é uma matriz diagonal de ordem n com os pesos w_i de cada ponto amostral; e $\vec{\beta}$ é o vetor $m+1$ com os parâmetros desconhecidos da regressão (*i.e.,* os coeficientes da curva). Assim, a inferência para um ponto desejado (x_q, y_q) é obtida por:

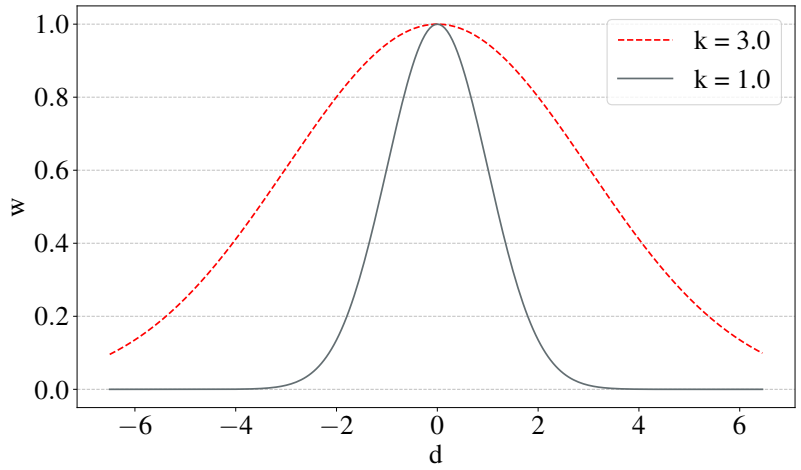
$$y_q = x_q^T \vec{\beta} \tag{3.3}$$

Os pesos permitem estabelecer a influência das amostras no modelo de predição de acordo com a sua distância (temporal) em relação ao ponto desejado. Várias funções de ponderação são propostas na literatura (CLEVELAND; DEVLIN, 1988; SIMONOFF, 2012; WAND; JONES, 1994). Contudo, a função de ponderação amplamente utilizada é baseada no *Kernel Gaussiano* (WAND; SCHUCANY, 1990), que pode ser definida por:

$$w_i = exp\left(\frac{-d_i^2}{2k^2}\right) \tag{3.4}$$

A partir da Equação 3.4, cada amostra (x_i, y_i) no modelo receberá um peso w_i no intervalo $[0, 1]$, em função da sua distância $d_i = \sqrt{(x_q - x_i)^2}$ para o ponto desejado x_q, como mostrado na Figura 3.7. Ou seja, quanto menor a distância do ponto (x_i, y_i) ao ponto desejado (x_q, y_q) (*i.e.,* $d_i \approx 0$), maior será a influência de (x_i, y_i) no modelo de predição, porque seu peso estará mais próximo de 100% (*i.e.,* $w_i \approx 1$).

Figura 3.7 – Magnitude do *Kernel Gaussiano*.



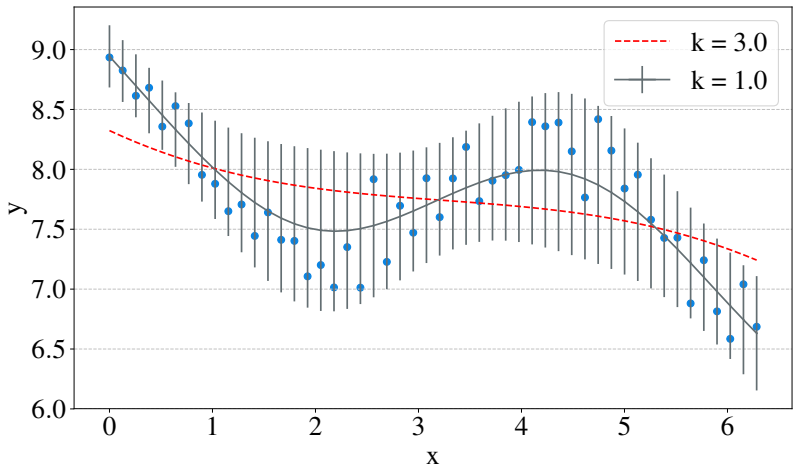
Fonte: do autor (2019).

O parâmetro k dimensiona a abrangência do *kernel*, para determinar o quão local será a regressão. A Figura 3.7 mostra o *kernel gaussiano* com $k = 1.0$ e $k = 3.0$. Ou seja, quanto maior o valor de k, maior o intervalo de amostras significativas no modelo. No entanto, se k for pequeno, o modelo será influenciado apenas por amostras muito próximas.

A Figura 3.8 mostra o comportamento do modelo de regressão, influenciado pelo ajuste da magnitude do *kernel gaussiano* (*i.e.,* ajuste de k). Devido à pequena variação amostral no modelo[1] $(6.5 \leqslant y \leqslant 9.0)$, quando $k = 1.0$, o resultado do modelo LWR é bem comportado e suave, acompanhando as mudanças no comportamento das amostras.

[1] As amostras na Figura 3.8 foram obtidas pela função: $f(x) = -(\sin(x) + (0, 4x) + 0, 26\eta) + 9$; onde η é um ruído amostral randômico e $0 \leqslant x \leqslant 2\pi$.

Figura 3.8 – Influência do parâmetro k no modelo LWR e intervalos de predição.



Fonte: do autor (2019).

De modo diferente, quando $k = 3.0$, o resultado do modelo LWR é semelhante ao resultado de uma regressão linear tradicional, que não permite acompanhar as mudanças no comportamento das amostras. Portanto, o ajuste fino do parâmetro k é essencial para o bom resultado no modelo LWR. Neste trabalho, como será apresentado na sequência, o parâmetro k é definido pelo desvio padrão amostral das variáveis independentes x_i. Isso permite adaptar dinamicamente o *kernel gaussiano* de acordo com as mudanças no comportamento do tráfego da rede.

Para avaliar a qualidade das predições é utilizado o intervalo de predição (SCHAAL; ATKESON, 1994), gerado pelo respectivo erro de cada inferência. Os intervalos de predição I_q, são limites esperados do erro de predição em um ponto de consulta x_q e, são definidos por:

$$I_q = x_q^T \vec{\beta} \pm t_{\alpha/2,n'-p'} s \sqrt{1 + x_q^T (X^T W^T W X)^{-1} x_q} \tag{3.5}$$

onde $t_{\alpha/2,n'-p'}$ é o valor do teste t de *Student* com $n' - p'$ graus de liberdade (Equação 3.6), para um nível de precisão em $100(1-\alpha)\%$.

$$n' = \sum_{i=1}^{n} w_i^2 \quad \textit{and} \quad p' = \frac{n'}{n} m \tag{3.6}$$

A Figura 3.8 mostra o comportamento do intervalo de predição para as inferências (x_q, y_q) com $k = 1.0$. Isso mostra que o intervalo de predição aumenta quando há uma grande variação nas amostras (*e.g.*, $x \approx 2$). Quando o conjunto amostral no modelo é bem comportado (*e.g.*, $x \approx 0$), o intervalo de predição é menor. Isso permite observar a eficácia do método de inferência mesmo com grande variação amostral.

Algumas adaptações foram realizadas para a aplicação do modelo LWR no contexto desse trabalho, para predizer o comportamento de fluxos em rede de PTT, admitindo a correlação temporal no tráfego da rede. O tamanho e a duração de um novo fluxo são adotados como variáveis dependentes y_q na Equação 3.3, permitindo suas predições individualmente. Em ambos os casos, as variáveis independentes x_i são os instantes iniciais (*ingressTimeStamps*) dos fluxos anteriores (*i.e.,* amostras), e x_q é o instante inicial do novo fluxo, em análise.

Para calcular a correlação temporal e a periodicidade entre o comportamento dos fluxos, foi definida uma distância temporal diária (*i.e.,* dentro de 24 horas), entre as amostras d_i e o instante inicial do novo fluxo, de acordo com a Equação 3.7. Ou seja, a maior distância temporal entre uma amostra e o novo fluxo será de até 12 horas. Assim, é possível correlacionar as ocorrências periódicas de fluxos elefantes diariamente, ainda que as amostras tenham iniciado em dias diferentes. Por exemplo, é possível reconhecer o padrão de periodicidade dos fluxos elefantes gerados por atualizações de sistema, *backup* de banco de dados, chamadas de vídeo periódicas e assim por diante. Para fins de exemplificação, nesse trabalho, o *timeStamp* é utilizado em horas.

$$d_i = |x_q - x_i| \, mod \, 24 = \begin{cases} d_i & \textbf{if } d_i \leq 12 \\ 24 - d_i & \textbf{if } d_i > 12 \end{cases} \tag{3.7}$$

A ponderação das amostras no modelo LWR é definida de acordo com a Equação 3.4. Nesse trabalho, uma modificação é realizada na função gaussiana, como mostrado na Equação 3.8. A partir disso, o operador de rede pode estabelecer um peso mínimo para um determinado intervalo de amostras. Ou seja, as distâncias no intervalo $[0, k]$ podem receber um peso mínimo igual a l (*e.g.,* 30% ou 40%), quando d_i (Equation 3.7) é igual a k. Isso permite ao operador determinar o quão conservador será o processo de inferência. Por exemplo, em cenários em que a rede está saturada, o operador pode ser mais rigoroso na ponderação das amostras e definir qual distância temporal terá influência mais significativa nas predições.

$$w_i = exp\left(\frac{d_i^2}{k^2} \cdot \ln(l)\right) \tag{3.8}$$

Para ajustar a magnitude do *kernel gaussiano*, é proposto que k seja definido pelo desvio padrão amostral das variáveis independentes x_i. Isso permite adaptar o *kernel* dinamicamente, de acordo com as mudanças no comportamento da rede. Finalmente, quando os pesos das amostras forem definidos, é possível determinar os parâmetros $\vec{\beta}$ do modelo de inferência (Equation 3.2) e, em seguida, prever o volume e duração do novo fluxo (Equação 3.3) de acordo com seu *ingressTimeStamp* (*i.e.,* x_q).

3.2.3 Validação das Predições e Classificação dos Fluxos

Depois de predizer os parâmetros de um fluxos, o mecanismo precisa executar a validação dos valores inferidos. Para essa validação, o operador da rede deve definir uma tolerância de aceitação para o intervalo de predição calculado sobre cada inferência. O intervalo de predição determina a margem de flutuação para o valor inferido, como mostrado na Figura 3.8. Assim, o operador da rede pode determinar a tolerância de acordo com, por exemplo, a demanda utilização da rede. Ou seja, quando a rede é subutilizada, o operador pode definir uma tolerância mais suave; caso contrário, quando a rede estiver saturada, a tolerância pode ser mais conservadora.

Quando o intervalo de predição é calculado (Equação 3.5), é possível validar ou invalidar os valores inferidos. Uma inferência pode ser considerada inválida principalmente quando há falta de amostras no modelo de predição, ou quando há baixa correlação temporal e periodicidade entre elas. Esse fatores resultam em aumento significativo no intervalo de predição, fazendo-o exceder a tolerância aceitável para as predições.

Todavia, uma inferência (*e.g.,* $30s$) só será considerada válida se o intervalo de predição (*e.g.,* $pred\_interv = 2s$) for menor ou igual ao valor inferido multiplicado pela tolerância (*e.g.,* $tol = 30s \times 10\% = 3s$). Neste caso (*i.e.,* quando o $pred\_interv \leq tol$), os valores inferidos são confrontados com limiares predefinidos para classificar o fluxo como um elefante ou não. Se os limiares forem excedidos, então o fluxo é considerado um elefante e pode ser mitigado imediatamente. Caso contrário, se os limiares não forem excedidos, o fluxo não será considerado um elefante e, portanto, será tratado como um fluxo regular.

Neste último caso, o processo de análise continuará ocorrendo diretamente no plano de dados. O mesmo acontece quando as inferências são invalidadas (*i.e.,* $pred\_interv > tol$), pois não é possível predizer o comportamento do fluxo e, consequentemente, não é possível classificá-lo como um elefante. Desse modo, a análise e a possível identificação são realizadas pelo mecanismo IDEAFIX, implementado nos *switches*, conforme o apresentado na Seção 3.1. No IDEAFIX, os pacotes de cada fluxo são analisados diretamente no plano de dados programável e a identificação de um fluxo elefante ocorre sempre que o volume e a duração de um fluxo excederem os limiares predefinidos.

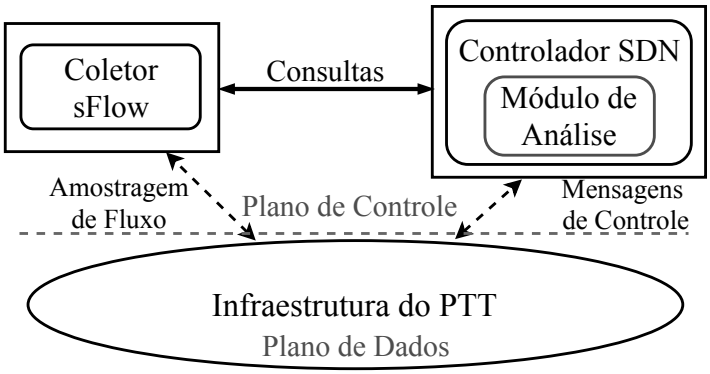
4 AVALIAÇÃO DE DESEMPENHO

Neste capítulo é realizada a avaliação de desempenho do mecanismo para predizer e identificar fluxos elefantes em redes de PTT. Primeiro, é descrito o mecanismo de comparação desenvolvido para identificação de fluxos elefantes em redes SDN tradicionais com os protocolos *OpenFlow* e *sFlow*, baseado nos trabalhos relacionados (KNOB *et al.*, 2016; CURTIS *et al.*, 2011). Em seguida, é apresentado o cenário experimental utilizado, os parâmetros de sistema, métricas utilizadas e a metodologia de avaliação. Por fim, são apresentados e discutidos os principais resultados obtidos com os experimentos.

4.1 Abordagem de Comparação

Para efeito de comparação, foi desenvolvida uma aplicação para a análise e identificação de fluxos elefantes em redes de PTT, com *SDN/OpenFlow*, baseada nos trabalhos relacionados (KNOB *et al.*, 2016; CURTIS *et al.*, 2011). O mecanismo do estado-da-arte foi desenvolvido utilizando os protocolos *OpenFlow* (MCKEOWN *et al.*, 2008b) e *sFlow* (SFLOW.ORG, 2018). Nesse mecanismo, um coletor *sFlow* realiza amostragem de tráfego nos switches de borda e estima as taxas de transmissão de cada fluxo. Foram adotadas taxas de amostragem de 1/128 e 1/1024 pacotes, para analisar os resultados nessa faixa, dado que o coletor *sFlow* estima a largura de banda em taxas de amostragem distintas. Além disso, um módulo de análise solicita ao coletor *sFlow* as estimativas de fluxo com intervalo de 0,01s, 0,1s e 0,5s entre as consultas. A arquitetura desse mecanismo é ilustrada na Figura 4.1. Quando um fluxo elefante é identificado, o controlador é notificado para inserir uma regra no *switch* de borda para marcar os pacotes do fluxo identificado (*e.g.,* setar uma *flag* no cabeçalho IP) e encaminhá-los por um caminho alternativo.

Figura 4.1 – Arquitetura do mecanismo de comparação.

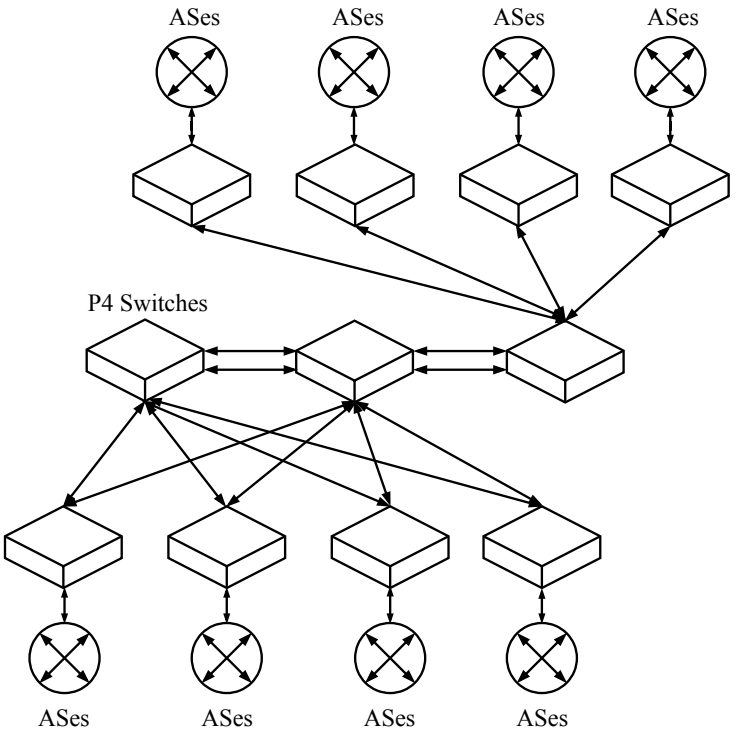


Fonte: do autor (2018).

4.2 Ambiente de avaliação

A Figura 4.2 mostra a topologia usada nos experimentos, baseada na infraestrutura do PTT de Amsterdam (AMS-IX, 2018), também utilizada em trabalhos relacionados (KNOB *et al.*, 2016). A topologia apresenta uma infraestrutura com 8 ASes conectados à rede de PTT com pelos menos oito *switches* de borda. Foram criados fluxos TCP (*Transmission Control Protocol*) entre os ASes seguindo um modelo cliente-servidor. Os fluxos foram gerados considerando dois fatores: largura de banda e duração. A largura de banda para cada fluxo foi estabelecida em 10Mbps, e suas durações foram obtidas a partir de uma distribuição normal com média de 8 segundos e desvio padrão de 5 segundos. Os limiares de classificação de fluxo elefante foram definidos em 10MB e 10 segundos (KNOB *et al.*, 2017). Para cada rodada de teste, foram gerados 2048 fluxos, dos quais aproximadamente 12% eram fluxos elefantes. Cada experimento durou 10 minutos e foi repetido pelo menos 32 vezes.

Figura 4.2 – Topologia da Rede de PTT.



Fonte: adaptado (KNOB *et al.*, 2016).

A fim de priorizar os requisitos de tempo no contexto de uma rede de PTT, na qual decisões/ações precisam ser tomadas rapidamente, uma estratégia proativa foi desenvolvida em ambas abordagens. Ou seja, quando um fluxo é iniciado, o controlador insere regras nos *switches* para que o novo fluxo seja encaminhado por um caminho padrão; do mesmo modo, são

inseridas regras de roteamento para um caminho alternativo. Quando um fluxo elefante é identificado, seus pacotes são marcados (por exemplo, definir um sinalizador no cabeçalho IP) para indicar a identificação e são encaminhados pelo um caminho alternativo, imediatamente. Essa abordagem exige mais recursos de memória para as tabelas de repasse. No entanto, isso permite reagir a um fluxo elefante mais rapidamente, como será mostrado nos resultados a seguir.

Os experimentos foram realizados em um computador com processador Intel Core i7-4790 com 8 núcleos de 3,6 GHz cada; 16 GB de memória RAM; e sistema operacional *Linux Ubuntu* 16.04 LTS. O protótipo dos mecanismos apresentados nesse trabalho foram implementados na linguagem $P4_{16}$, utilizando o *software switch P4* de referência, BMv2 [1]. O mecanismo do estado-da-arte foi desenvolvido com o *RYU SDN Framework* versão 4.2 (RYU, 2019), utilizando o protocolo *OpenFlow* versão 1.3 (MCKEOWN *et al.*, 2008b), o coletor *InMon sFlow-RT* [2] e *Open vSwitch* versão 2.0.2 (SFLOW.ORG, 2018). A infraestrutura foi emulada utilizando *Mininet* versão 2.3 (MININET, 2019), com largura de banda de 1Gbps por link e sem atraso de propagação. A carga de trabalho foi gerada com *iPerf* versão 3.0.11 (IPERF, 2019).

4.3 Métricas de Avaliação

Na avaliação de desempenho, o mecanismo de identificação de fluxos elefantes, IDEAFIX, foi comparado ao mecanismo desenvolvido baseado no estado-da-arte (Seção 4.1), a partir das seguintes métricas: (i) tempo de reação, (ii) dados excedentes, (iii) utilização de recursos no mecanismo do estado-da-arte, (iv) dados de monitoração, (v) a acurácia do mecanismo e (vi) o tempo do processamento de pacotes no *switch* P4.

Por tempo de reação, é considerado o tempo em que o primeiro pacote do fluxo identificado como elefante sai do *switch* de borda pela rota alternativa, menos o tempo em que o pacote responsável por exceder os limiares chegou no *switch*. Por volume de dados excedentes, é considerada a quantidade de *bytes* que trafegaram na rota padrão desde que o fluxo tornou-se um elefante (chegada do pacote que excede os limiares), até que a reação tenha ocorrido. A utilização de recursos, no mecanismo do estado-da-arte, é o custo computacional demandado pelo módulo de análise, no plano de controle (ver Figura 4.1).

Por volume de monitoração, é considerado o volume inserido na rede com o mecanismo de análise e identificação. Para o IDEAFIX, este é o volume (em *bytes*) das mensagens de notificação enviadas ao controlador, para reportar os fluxos identificados. Já no mecanismo do estado-da-arte, este é o volume de dados gerado com o processo de amostragem dos fluxos pelo

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

coletor *sFlow*. Para a acurácia do IDEAFIX, considera-se o número de falsos positivos: o percentual dos fluxos identificados como elefantes indevidamente; e falsos negativos: o percentual dos fluxos que eram elefantes e não foram identificados; em função do espaço para mapeamento *hash* nos *switches*. O tempo do processamento de pacotes no *switch* P4 diz respeito à sobrecarga que o mecanismo de análise e identificação incorre sobre o *switch*, comparado com o tempo necessário para realizar, somente, o tradicional encaminhamento de pacotes. Essas métricas são consideradas melhores, ao passo que admitem valores menores.

Para o mecanismo de predição, foram avaliadas as seguintes métricas: (vii) o tempo de predição/reação, $(viii)$ dados sem identificação, (ix) utilização de recursos pelo módulo de predição e (x) acurácia do mecanismo. O tempo de predição/reação, compreende o tempo desde a chegada, no *switch* de borda, do primeiro pacote do fluxo predito como um elefante, até o momento em que seu primeiro pacote sai deste *switch* para ser encaminhado pelo caminho alternativo (reação). Esse tempo incorre todo o processo de predição apresentado no Capítulo 3.2, acrescido do tempo necessário para instalar as regras de identificação/encaminhamento no *switch* de borda.

A quantidade de dados sem identificação (métrica $viii$), corresponde a quantidade de dados que trafegaram pelo caminho padrão até que o processo de predição e avaliação tenha sido concluído, o fluxo elefante seja identificado e, por fim, a reação aconteça. Essa métrica, de modo semelhante à métrica ii, denota a quantidade de tráfego dos fluxos elefantes que consumiram os recursos dos dispositivos de rede pela rota padrão e que, indiretamente, influenciaram no escoamento de fluxos menores que estavam compartilhando o mesmo caminho.

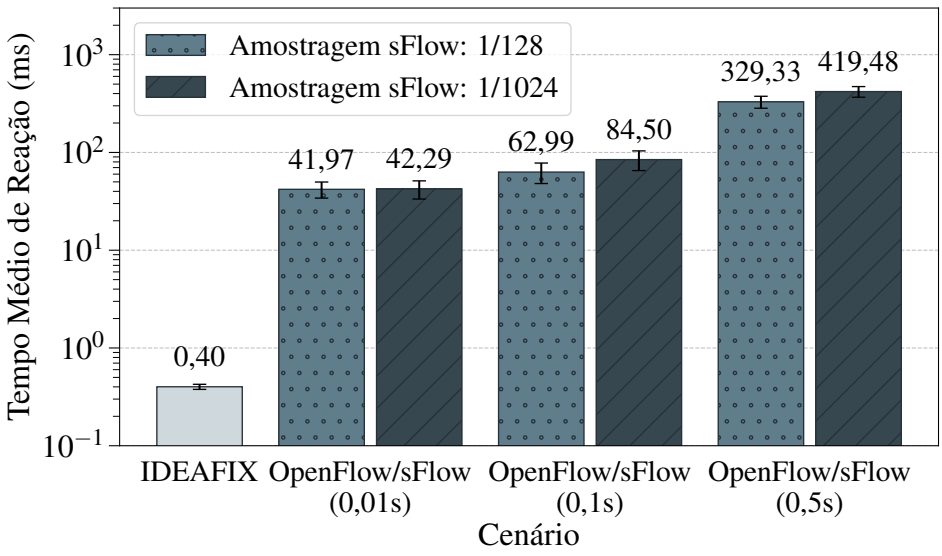
A utilização de recursos (métrica ix) expressa o custo computacional exercido, no plano de controle, pelo módulo de predição, em função da quantidade de amostras utilizadas no modelo. Por fim, a acurácia do mecanismo de predição (métrica x) foi avaliada a partir do percentual de predições validadas, em função da tolerância definida para aceitação dos erros de predição, expresso pelos intervalos de predição. As métricas vii, $viii$ e xi, são consideradas melhores, ao passo que admitem valores menores. Para a métrica x, valores maiores são considerados melhores.

4.4 Resultados

Nesta seção são apresentados os principais resultados obtidos na avaliação experimental realizada sobre os mecanismos propostos nesse trabalho. Primeiro, são apresentados os resultados obtidos para o mecanismo de identificação de fluxos elefantes diretamente no plano de dados, denominado IDEAFIX. O desempenho do IDEAFIX foi comparado com a abordagem baseada nos trabalhos relacionados, que depende do envolvimento do plano de controle no processo de identificação, como descrito na Seção 4.1. Na sequência, são apresentados os resultados obtidos na avaliação do mecanismo de predição do comportamento de fluxos, utilizando o processo de inferência baseado no modelo *Locally Weighted Regression* - LWR (ver Seção 3.2.2).

A Figura 4.3 mostra os resultados para o tempo de reação aos fluxos elefantes, quando os limiares são excedidos. As abordagens avaliadas estão descritas no eixo x. O módulo de análise no mecanismo do estado-da-arte solicita as estimativas dos fluxos para o coletor *sFlow* em uma frequência de 0,01s, 0,1s e 0,5s, e com taxa de amostragem de 1/128 e 1/1024 pacotes. No eixo y, em escala logarítmica, é apresentado o tempo médio de reação (em milissegundos) para cada uma das abordagens avaliadas. A partir disso, é possível observar que a proposta apresentada neste trabalho para identificar fluxos elefantes diretamente no plano de dados, IDEAFIX, pode identificar e reagir aos fluxos elefantes em até 0.4ms. Este tempo é basicamente o tempo do processamento de pacotes pelo *software switch P4*. Em *hardware switch*, espera-se que o tempo do processamento de pacotes seja ainda menor.

Figura 4.3 – Tempo de reação, quando os limiares são excedidos.

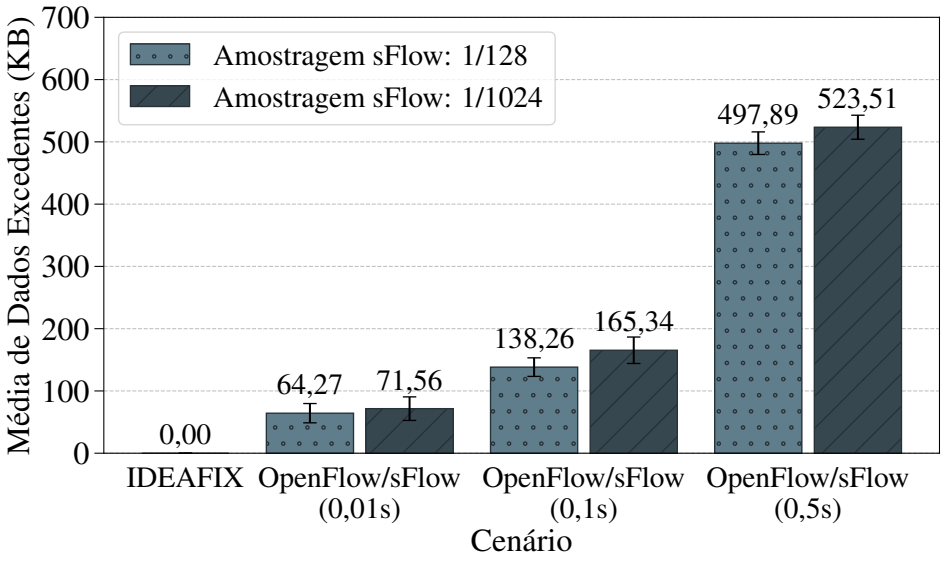


Fonte: do autor (2018).

O tempo médio de reação no mecanismo do estado-da-arte (*OpenFlow/sFlow*) varia entre 41,97ms, no melhor caso, e 419,48ms, no pior caso. Isso decorre do tempo de comunicação entre o módulo de identificação e o coletor *sFlow*. Os intervalos de confiança se sobrepõem em seus níveis correspondentes, para taxas de amostragem de pacotes em 1/128 e 1/1024. Assim, com um nível de confiança em 95%, não é possível admitir que há diferença significativa entre essas taxas de amostragem. Esse comportamento era esperado, já que o coletor *sFlow* é capaz de estimar a largura de banda dos fluxos mesmo com diferentes taxas de amostragem. Portanto, só é presenciada diferença quando há variação no intervalo de consultas ao coletor *sFlow*.

A Figura 4.4 apresenta o volume de dados que excederam os limites e continuaram sendo encaminhados pela rota padrão até a reação ocorrer. Com IDEAFIX, não há dados excedentes, pois a reação ocorre imediatamente com o processamento do pacote que caracteriza o fluxo como elefante. No entanto, no mecanismo do estado-da-arte, há uma variação entre 64,27KB, no melhor dos casos, e aproximadamente 524KB, no pior dos casos, quando o intervalo de consulta ao coletor *sFlow* é de 0,01s e 0,5s, respectivamente.

Figura 4.4 – Dados excedentes.



Fonte: do autor (2018).

Esses valores são uma consequência do tempo de comunicação entre os *switches* e o plano de controle, e o intervalo de comunicação entre o módulo de identificação e o coletor sFlow. Quando esse intervalo é curto (*e.g.,* 0,01s), o tempo de reação e os dados excedentes diminuem. No entanto, a utilização de recursos, pelo módulo de análise, no mecanismo do estado-da-arte aumenta, conforme mostra a Tabela 4.1. Quando esse intervalo é longo (*e.g.,* 0,5s), o custo computacional é relativamente pequeno, embora o tempo de reação e os dados excedentes sejam

maiores. Quando o intervalo é de 0,1s, há um custo computacional próximo ao obtido com 0,5s. Porém, os ganhos nas outras métricas aproximam-se dos obtidos com 0,01s.

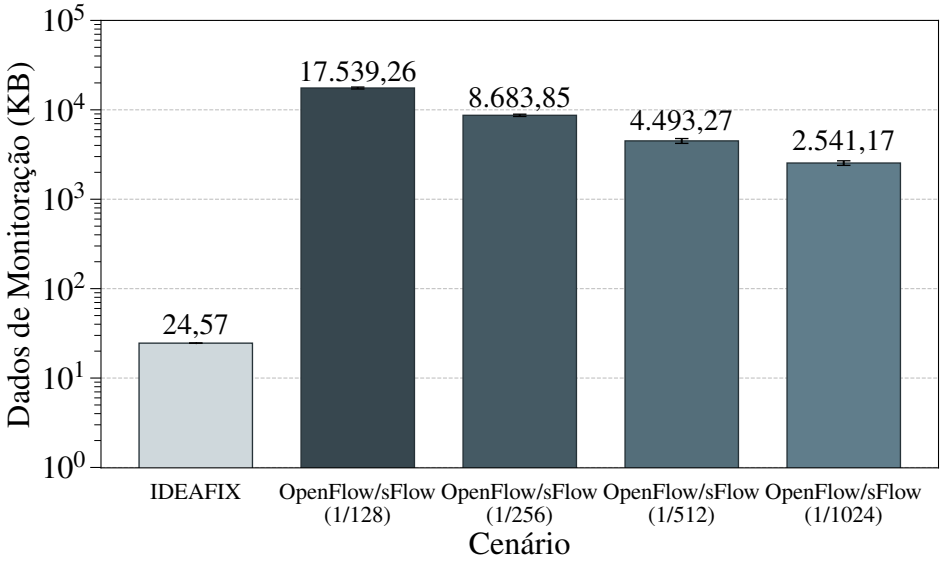
Tabela 4.1 – Utilização de recursos pelo módulo de identificação.

Intervalo de Consultas (s)	0,01	0,1	0,5
Uso de CPU	27%	7%	3%
Uso de Memória (MB)	67,1	67,1	67,1

Fonte: do autor (2018).

Os dados inseridos na rede com o mecanismo de monitoramento são mostrados na Figura 4.5. O IDEAFIX envia apenas uma notificação ao plano de controle informando o fluxo identificado, um pacote dos 96 *bytes*. Assim, a quantidade de dados de monitoramento é diretamente proporcional à quantidade de fluxos elefantes identificados. Nos experimentos, 2048 fluxos foram inseridos na rede por rodada de teste, sendo aproximadamente 256 fluxos elefantes. Portanto, os dados de monitoramento na rede foram de aproximadamente 24,57KB.

Figura 4.5 – Dados de monitoramento.

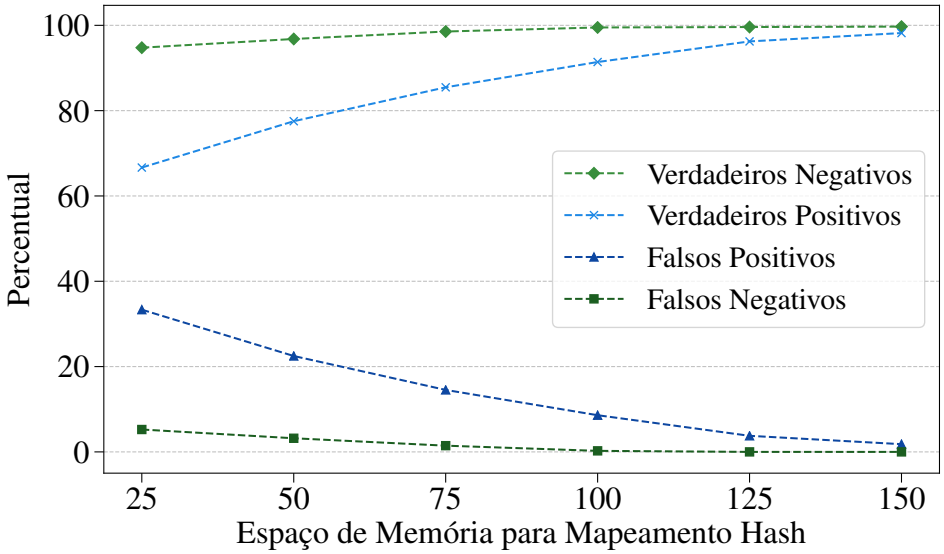


Fonte: do autor (2018).

No mecanismo do estado-da-arte, a quantidade de dados de monitoramento depende da taxa de amostragem de pacotes. Ao amostrar 1/128, há cerca de 17MB de dados de monitoramento e com uma amostragem de 1/1024, há cerca de 2,5MB. Isso representa uma diferença de aproximadamente 14,5% da última taxa de amostragem, sobre a primeira. Além disso, em taxas de amostragem nesse intervalo (*i.e.,* 1/128, 1/256, 1/512 e 1/1024), é possível observar um aumento gradual na quantidade de dados de monitoramento.

A acurácia do IDEAFIX, ilustrada na Figura 4.6, foi avaliada variando o espaço de memória nos *switches* para o mapeamento *hash* em relação ao número de fluxos inseridos na rede (*i.e.,* 2048). Com 25% de espaço disponível, há uma percentual de aproximadamente 5% de falsos negativos e 35% de falsos positivos. Esse resultado reflete a estratégia que superestima os fluxos, em que os falsos positivos são preferíveis em relação aos falsos negativos, quando ocorrem colisões de chave *hash*. Quando o espaço de mapeamento *hash* aumenta (*e.g.,* 100%), os falsos positivos e os falsos negativos diminuem (*i.e.,* menor que 10% e 1%, respectivamente) e verdadeiros positivos e verdadeiros negativos aumentam, alcançando cerca de 95% da precisão.

Figura 4.6 – Acurácia do mecanismo.



Fonte: do autor (2018).

A Tabela 4.2 mostra o tempo médio (μ), em milissegundos, do processamento de pacotes nos *switches* P4. Há um aumento de aproximadamente 30% em comparação com o processamento tradicional, que somente executa o encaminhamento de pacotes. Como mencionado anteriormente, o tempo de processamento dos pacotes pelo *software switch* P4 influencia nessa métrica. Em *hardware switch*, esse tempo seria na ordem de nanossegundos. Os intervalos de confiança (ε) não se sobrepõem a um nível de confiança de 95%.

Tabela 4.2 – Tempo do processamento de pacotes (ms).

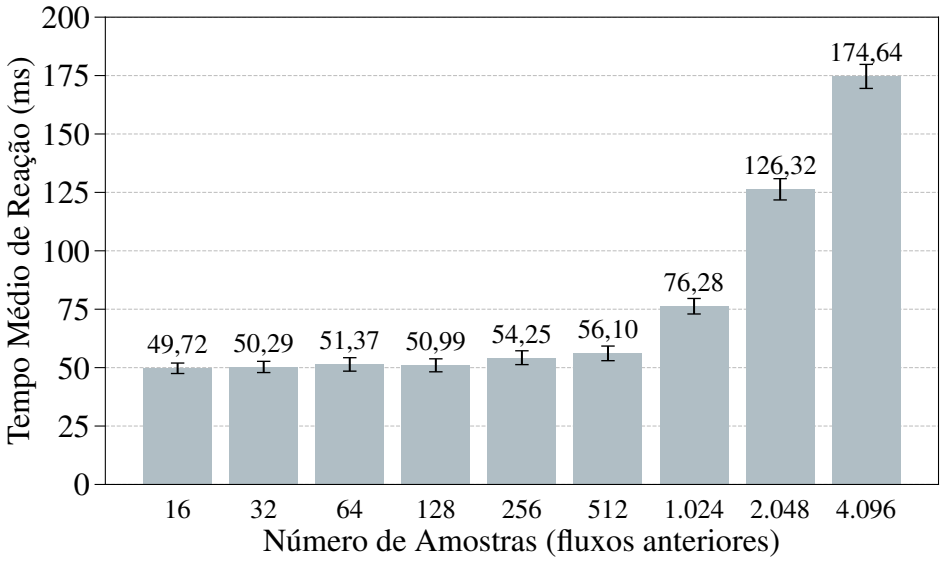
Switch P4	$\mu - \varepsilon$	μ	$\mu + \varepsilon$
Com IDEAFIX	0,256	0,279	0,302
Apenas encaminhamento	0,205	0,216	0,227

Fonte: do autor (2018).

Para a estratégia de identificação de fluxos elefantes que utiliza o mecanismo de predição do comportamento de fluxos, a avaliação obteve os resultados apresentados a seguir. Primeiro, na Figura 4.7, são mostrados os resultados para o tempo de reação aos fluxos elefantes, que já inclui o tempo utilizado na predição. Ou seja, como apresentado no Capítulo 3.2, quando um *switch* recebe o primeiro pacote de um fluxo, uma notificação é encaminhada ao plano de controle para estabelecer o caminho desse novo fluxo na rede. Nesse momento, o processo de predição é realizado para tentar identificar o fluxo como elefante ainda que no seu início, baseado no comportamento de fluxos anteriores, mesmo antes que o fluxo exceda os limiares.

O tempo de reação foi analisado em função do número de amostras utilizadas no modelo de predição. Ou seja, a quantidade de instâncias de fluxos anteriores utilizadas no processo de inferência. Isso também permite observar o tempo necessário para que uma reação ocorra com o modelo de previsão sendo gerado em tempo de execução. No eixo y, o tempo médio de reação (em milissegundos) não apresenta diferença significativa, a um nível de confiança em 95%, para um conjunto de até 512 amostras. A diferença mais significativa é evidenciada quando há pelo menos 1.024, 2.048 e 4.096 amostras (*i.e.,* mais processamento). Nesses casos, o tempo médio de reação varia entre 76,28ms, 126,32ms e 174,64ms, respectivamente.

Figura 4.7 – Tempo Médio de Reação.



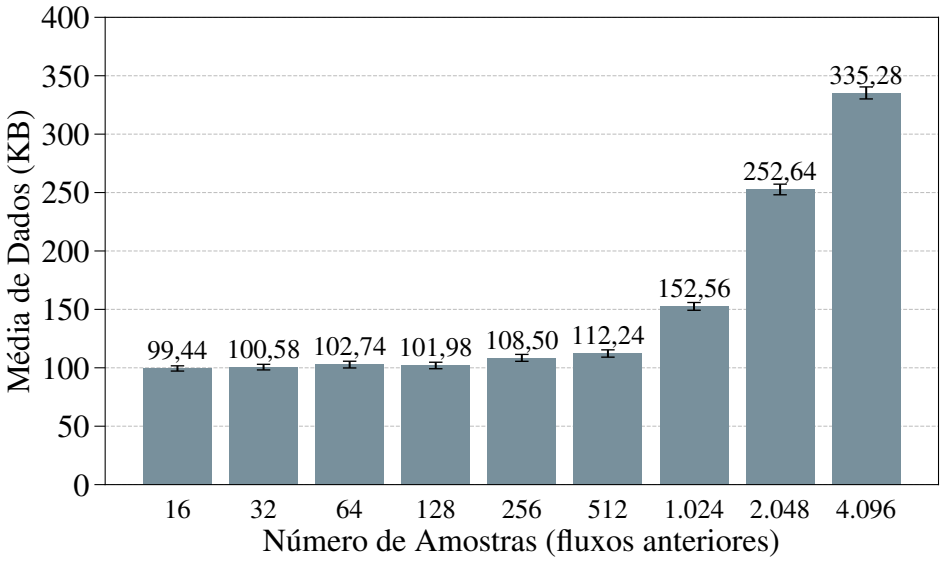
Fonte: do autor (2019).

Esses números são muito menores se comparados ao tempo que o fluxo levaria para, de fato, exceder os limiares e então ser classificado como um elefante. Ou seja, para os limiares utilizados na avaliação anterior, IDEAFIX (Figura 4.3), com base nos trabalhos relacionados (KNOB *et al.*, 2016; CURTIS *et al.*, 2011), um fluxo elefante levaria pelo menos 10s para ser

identificado, na melhor das hipóteses. Desse modo, é possível perceber que prever antecipadamente o comportamento dos fluxos permite identificar os elefantes o quanto antes. Além disso, como já mencionado, o tempo do processamento de pacotes pelo *software switch P4* também influencia o tempo de reação (ver Tabela 4.2). Isso ocorre pois o *switch* que é o responsável por notificar ao plano de controle sobre a chegada de um novo fluxo, para só então, permitir que o controlador administre o fluxo.

A Figura 4.8 mostra o volume de dados dos fluxos elefantes que percorreram o caminho padrão até ocorrer uma reação. No eixo y, a quantidade média de dados (em Kilobytes) não tem diferença significativa a um nível de confiança em 95%, para até 512 amostras. Esse resultado segue o comportamento do tempo de predição/reação, apresentado anteriormente na Figura 4.7. Como na análise anterior, a diferença mais significativa é evidenciada quando há conjuntos com pelo menos 1.024, 2.048 e 4.096 amostras de fluxos anteriores no modelo de predição. Nesses casos, é possível observar que os fluxos elefantes ainda trafegam cerca de 152,56KB, 252,64KB e 335,28KB pela rota padrão na rede, respectivamente.

Figura 4.8 – Dados sem identificação.



Fonte: do autor (2019).

A utilização de recursos pelo mecanismo de predição é apresentada na Tabela 4.3. Um baixo consumo de recursos é evidenciado quando há um pequeno número de amostras (*i.e.,* $\approx$ 64). Em contraste, à medida que esse número aumenta, ou seja, 1.024, 4.096 amostras, exige-se mais recursos, bem como é necessário mais tempo para ocorrer as reações, tal como ilustrado na Figura 4.7. Nesses casos, o uso da CPU pode atingir até 30% e 60%, respectivamente. É esperado que os recursos necessários para predizer o comportamento dos novos fluxos demandem

a um maior consumo de recursos. No entanto, considerando os impactos que os fluxos elefantes podem gerar na rede, esse custo de recursos computacionais, no plano de controle, pode ser considerado pequeno e tolerável.

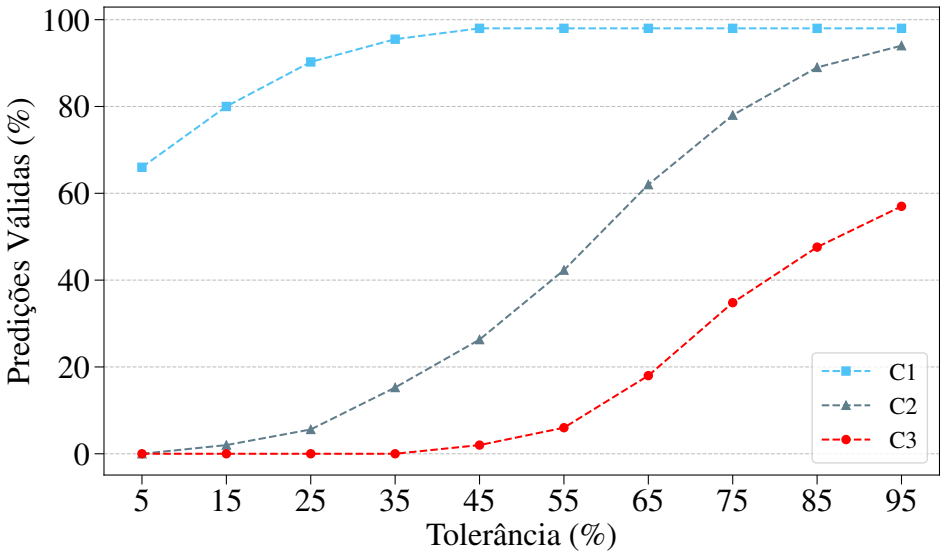
Tabela 4.3 – Uso de recursos pelo módulo de predição.

Número de amostras	64	1024	4096
Uso de CPU	12,8%	30,4%	68,7%
Uso de memória	0,3%	0,6%	2,1%

Fonte: do autor (2019).

A precisão do mecanismo de predição foi avaliada a partir do percentual de predições válidas, em função da tolerância no intervalo de predição (ver Seção 3.2.3). A Figura 4.9 mostra o percentual de previsões válidas (ou seja, fluxos elefantes corretamente previstos e identificados) em três cenários de comportamento de tráfego distintos. O primeiro cenário (curva C1) tem baixa variação no comportamento de tráfego e periodicidade regular (isto é, os fluxos seguem uma frequência bem definida). O cenário dois (curva C2) apresenta maior variação no comportamento e periodicidade dos fluxos regular. O cenário três (curva C3) tem uma alta variação no comportamento dos fluxos e baixa periodicidade (isto é, os fluxos não possuem um padrão de frequência bem definido). Assim, foi avaliada a porcentagem de predições válidas de acordo com a tolerância ao intervalo de predição, definida pelo operador de rede. O intervalo de predição foi calculado para cada inferência a um nível de confiança de 90% (ver Equação 3.5).

Figura 4.9 – Acurácia do mecanismo de predição.



Fonte: do autor (2019).

A Figura 4.9 mostra que mais de 80% das predições são validadas mesmo com uma tolerância conservadora (*e.g.,* 15%), em um cenário bem comportado (curva C1). No pior caso (curva C3), quando o comportamento do tráfego e a periodicidade não são regulares, aproximadamente 20% das previsões puderam ser validadas, com tolerância de pelo menos 65%. No caso médio (curva C2), mais de 60% das previsões foram validadas com uma tolerância de 65%. Esses resultados mostram que o método pode prever e validar valores mesmo em cenários não regulares. No entanto, isso requer mais flexibilidade na tolerância das previsões. Em cenários bem comportados, mesmo com tolerância conservadora, é possível validar pelo menos 80% das previsões.

Estes resultados representam, pelo menos, 90% de sucesso na identificação de fluxos elefantes, em relação ao número total de fluxos elefantes inseridos na rede (*i.e.,* $\approx$ 256 fluxos), com aproximadamente 5% de falsos positivos. Por fim, os fluxos elefantes cujas previsões não eram validadas, foram identificados diretamente no plano de dados, com base na abordagem anterior, IDEAFIX, imediatamente após excederem os limiares.

5 CONCLUSÃO

Um importante problema no gerenciamento de redes de PTTs diz respeito à identificação de fluxos elefantes, que são fluxos com grande volume de tráfego e longa duração. Por esses motivos, os fluxos elefantes podem impactar significativamente no tráfego de fluxos menores, que compartilham o mesmo caminho na infraestrutura dos PTTs, comprometendo a percepção geral de QoS da rede. Devido aos impactos que os fluxos elefantes têm sobre o desempenho da rede, estes devem ser identificados o quanto antes para que ações de mitigação sejam prontamente realizadas.

Diante disso, este trabalho apresenta duas abordagens para identificar fluxos elefantes em redes de ponto de troca de tráfego com suporte a programabilidade: uma utilizando predições do comportamento de fluxos e outra realizando análise diretamente no plano de dados. Em contraste com as abordagens do estado-da-arte, que exigem que os fluxos excedam os limiares para, então, classificá-los como fluxos elefantes, um mecanismo de predição é apresentado para inferir o comportamento dos fluxos logo que ingressam na rede.

Todavia, para os casos em que as predições não forem validadas, um segundo mecanismo pode ser utilizado para identificar os fluxos elefantes diretamente no plano de dados. Essa abordagem, chamada IDEAFIX, mesmo dependendo que um fluxos exceda os limiares para, então, classificá-lo como um elefante, é significativamente mais rápida e eficiente quando comparada com um mecanismo de identificação baseado nas abordagens do estado-da-arte, implementado com *SDN/OpenFlow* e utilizando *sFlow* para realizar amostragem dos fluxos.

5.1 Principais Contribuições e Resultados Obtidos

Considerando a carga de trabalho sintética e os parâmetros utilizados nos experimentos, os resultados desse trabalho mostram que a abordagem de identificação realizada diretamente no plano de dados, IDEAFIX, pode identificar e reagir aos fluxos elefantes com rapidez e eficiência, inserindo poucos dados de monitoramento na rede em comparação com uma abordagem tradicional implementada com *SDN/OpenFlow*, e que realiza amostragem de fluxo com *sFlow*. No entanto, é necessário mais espaço de memória nos dispositivos de rede para realizar o mecanismo de mapeamento *hash* e o armazenamento das informações, além de mais espaço nas tabelas de repasse, para uma estratégia de reação preventiva.

Diante disso, se o operador de rede prioriza pela identificação imediata dos fluxos elefantes, assim que ocorrer a violação dos limiares, o IDEAFIX é o mais indicado, em relação a uma abordagem que envolve o plano de controle no *loop* de identificação. Os resultados mostram

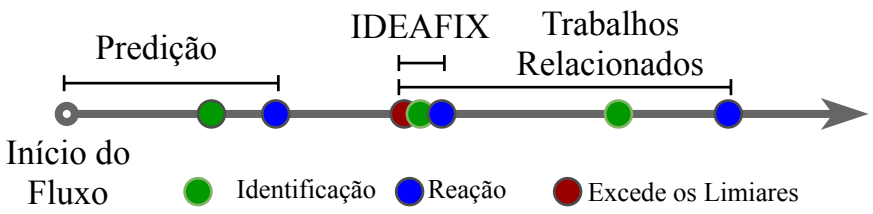
que, enquanto os mecanismos do estado-da-arte soma até 17MB de dados de monitoramento inseridos na rede, o IDEAFIX causa uma sobrecarga de apenas 25KB. Além disso, o IDEAFIX demanda de aproximadamente 0.40ms para identificar e reagir aos fluxos elefantes (em *software switch*), com precisão de até 95%, em cenários com recursos de memória escassos.

No entanto, o operador da rede pode não priorizar pelo tempo de identificação, seja por requisitos de memória ou por já possuir implantado na infraestrutura de rede um mecanismo com *sFlow*. Neste caso, os resultados deste trabalho indicam que é possível identificar e reagir aos fluxos elefantes combinando os parâmetros da taxa de amostragem e frequência de consultas ao coletor *sFlow*, para incorrer em melhor utilização dos recursos, balancear a quantidade de dados de monitoração inseridos na rede e melhorar o tempo de identificação.

Quanto ao mecanismo de predição, em contraste com as abordagens que exigem que os fluxos excedam os limiares para então classificá-los como fluxos elefantes, este se mostrou capaz de predizer e reagir aos fluxos elefantes logo após seu ingresso na rede. Os resultados experimentais mostram que o mecanismo é capaz de predizer, o volume e a duração de novos fluxos e reagir rapidamente aos fluxos elefantes, em aproximadamente 50,3ms, com apenas 32 amostras no modelo de predição. Esses números são muito menores se comparados ao tempo que os fluxos levariam para exceder os limiares e, só então, serem classificados como elefantes. Além disso, a acurácia do mecanismo obteve até 80% de previsões validadas, mesmo com uma tolerância conservadora, e aproximadamente 5% de falsos positivos.

Embora o método de predição utilizado nessa abordagem exija que os modelos de inferência sejam processados em tempo de execução, os resultados mostram que é possível predizer e reagir rapidamente aos fluxos elefantes, em um intervalo significativamente menor que o tempo necessário para que o fluxo exceda os limiares. A Figura 5.1 mostra uma comparação abstrata do tempo de reação dos mecanismos apresentados neste trabalho (reação com o mecanismo de predição e IDEAFIX) e as abordagens baseadas nos trabalhos relacionados (reação com mecanismo do estado-da-arte), em relação ao início do fluxo e tempo para exceder os limiares.

Figura 5.1 – Tempo de vida do fluxo elefante e identificação/reação.



Fonte: do autor (2019).

A Figura 5.1, baseada nos resultados obtidos com a avaliação de desempenho, abstrai o

tempo de vida de um fluxo elefante e a estimativa do tempo necessário para que as abordagens apresentadas neste trabalho realizem sua identificação e reajam para mitigar seus efeitos. É possível observar que a reação com o mecanismo de predição ocorre mais rapidamente se comparada com as outras estratégias. Isso se dá porque essa abordagem não precisa esperar que o fluxo exceda os limiares para que a identificação aconteça. No entanto, quando não é possível predizer o comportamento do fluxo, isto é, seja por falta de padrão ou correlação temporal das amostras, ou quantidade amostral insuficiente, a identificação/reação pode ocorrer diretamente no plano de dados, com o IDEAFIX, de forma mais rápida e eficiente em relação às abordagens baseadas no estado-da-arte, assim que o fluxo exceder os limiares. Isso permite que ações sejam aplicadas nas redes de PTTs para que os impactos dos fluxos elefantes sejam mitigados.

5.2 Considerações Finais e Trabalhos Futuros

A partir do apresentado neste trabalho, é possível observar duas estratégias para identificar fluxos elefantes em redes de ponto de troca de tráfego. A primeira estratégia, IDEAFIX, mostrou que é possível utilizar os recursos disponíveis em redes com suporte a programabilidade para implementar diretamente nos *switches* um mecanismo capaz de identificar e reagir ao fluxos elefantes, imediatamente após os limiares serem excedidos. Por si só, IDEAFIX tornou o tempo de identificação/reação muito mais rápido, quando comparado com as abordagens do estado-da-arte, baseadas em redes *SDN/OpenFlow*, que envolvem o plano de controle no processo de identificação. No entanto, caso o operador da rede deseje utilizar um mecanismo de identificação implementado com *OpenFlow/sFlow*, a avaliação de desempenho também aponta os parâmetros que podem ser utilizados para equilibrar os custos e ganhos no processo de identificação dos fluxos elefantes.

A segunda abordagem, por sua vez, descreve um mecanismo para prever o comportamento dos fluxos logo que estes ingressam na rede de PTT, para então classificar os fluxos elefantes antes mesmo que venham exceder os limiares. O mecanismo apresentado é capaz de correlacionar amostras do comportamento de fluxos anteriores e, a partir da correlação temporal e periodicidade entre eles, predizer o comportamento dos novos fluxos. Isso permite reagir a um fluxo elefante muito antes que os limiares sejam excedidos, possibilitando mitigar seus efeitos logo em seu início. Toda via, quando não é possível validar as predições, o processo de análise e identificação segue com a abordagem descrita anteriormente, o IDEAFIX. Além disso, o IDEAFIX opera de forma colaborativa com o mecanismo de predição, entre outros aspectos, alimentando a base histórica utilizada nas inferências.

Em trabalhos futuros, pretende-se realizar um estudo para estabelecer os limiares de forma dinâmica e autônoma. Além disso, outros métodos de inferência podem ser utilizados para prever o comportamento dos fluxos, por exemplo, baseados em aprendizado de máquina. Também desejá-se estudar estratégias para mitigar os efeitos dos fluxos elefantes identificados, segundo as demandas e políticas de operação do PTT. Deseja-se ainda, estender a compreensão sobre novos fenômenos de fluxos, a partir de características distintas de volume de tráfego e duração, para analisá-los, entender seus efeitos em diferentes tipos de redes e, então, elaborar estratégias para sua identificação. Finalmente, deseja-se implementar estes mecanismos em *hardware switch* com suporte a programabilidade, para avaliar essas abordagens em um contexto real.

REFERÊNCIAS

AF-IX. **South African Internet Exchange Association**. 2018. Disponível em: <http://www.saix.net/cgi-bin/index.pl>.

AFAQ, M.; REHMAN, S.; SONG, W.-C. Visualization of Elephant Flows and QoS Provisioning in SDN-based Networks. In: IEEE. **17th Asia-Pacific Network Operations and Management Symposium (APNOMS)**. [S.l.], 2015. p. 444–447.

AFEK, Y. *et al.* Sampling and Large Flow Detection in SDN. **SIGCOMM Comput. Commun. Rev.**, ACM, New York, NY, USA, v. 45, n. 4, p. 345–346, ago. 2015.

AGER, B. *et al.* Anatomy of a Large European IXP. **ACM SIGCOMM Conference on Internet Measurement**, ACM, New York, NY, USA, v. 42, n. 4, p. 163–174, ago. 2012.

AMS-IX. **Amsterdam Internet Exchange Infrastructure**. 2018. Disponível em: <https://ams-ix.net/technical/ams-ix-infrastructure>.

____. **AMS-IX Infrastructure** . 2019. Disponível em: <https://ams-ix.net/technical/ams-ix-infrastructure>.

APIX. **Asia Pacific Internet Exchange Association**. 2018. Disponível em: <http://apix.asia/index.php>.

ARAUJO, G. *et al.* Caracterizando estratégias de domínio espacial para gerenciamento de regras em redes definidas por software. **35o. Simpósio Brasileiro de Redes de Computadores e Sistemas Distribuídos - SBRC 2017**, 2017.

AUGUSTIN, B.; KRISHNAMURTHY, B.; WILLINGER, W. IXPs: Mapped? In: **ACM SIGCOMM Conference on Internet Measurement**. NY, USA: ACM, 2009. (IMC '09), p. 336–349. ISBN 978-1-60558-771-4.

BASAT, R. *et al.* Constant Time Updates in Hierarchical Heavy Hitter. In: ACM. **ACM SIGCOMM Conference on Internet Measurement**. NY, USA, 2017. p. 127–140.

BERNAILLE, L. *et al.* Traffic Classification on the Fly. **ACM SIGCOMM Conference on Internet Measurement**, ACM, New York, NY, USA, v. 36, n. 2, p. 23–26, abr. 2006.

BOSSHART, P. *et al.* P4: Programming Protocol-independent Packet Processors. In: **ACM SIGCOMM Conference on Internet Measurement**. New York, NY, USA: ACM, 2014. p. 87–95.

CGI.BR, C. G. da Internet no B. **Brazil Internet Exchange Points**. 2018. Disponível em: <http://ix.br/trafego/agregado/rs>.

CHATZIS, N. *et al.* There is More to IXPs Than Meets the Eye. **ACM SIGCOMM Conference on Internet Measurement**, ACM, New York, NY, USA, v. 43, n. 5, p. 19–28, nov. 2013.

CHEN, Y.; YANG, B.; MENG, Q. Small-time Scale Network Traffic Prediction Based on Flexible Neural Tree. **Applied Soft Computing**, Elsevier, v. 12, n. 1, p. 274–279, 2012.

CLEVELAND, W. S.; DEVLIN, S. J. Locally Weighted Regression: an approach to regression analysis by local fitting. In: **Journal of the American statistical association**. [S.l.]: Taylor & Francis, 1988. v. 83, n. 403, p. 596–610.

COMBS, G. **Wireshark**. 2019. Disponível em: <https://www.wireshark.org/>.

CORTEZ, P. *et al.* Internet Traffic Forecasting using Neural Networks. In: IEEE. **Neural Networks, 2006. IJCNN'06. International Joint Conference on**. [S.l.], 2006. p. 2635–2642.

CURTIS, A. R.; KIM, W.; YALAGANDULA, P. Mahout: Low-overhead Datacenter Traffic Management Using End-host-based Elephant Detection. In: IEEE. **IEEE INFOCOM Conference on Computer Communications**. [S.l.], 2011. p. 1629–1637.

CURTIS, A. R. *et al.* DevoFlow: Scaling Flow Management for High-performance Networks. In: **ACM SIGCOMM Conference on Internet Measurement**. NY, USA: ACM, 2011. v. 41, p. 254–265.

DAINOTTI, A. *et al.* Classification of Network Traffic via Packet-level Hidden Markov Models. In: IEEE. **Global Telecommunications Conference, 2008. IEEE GLOBECOM 2008. IEEE**. [S.l.], 2008. p. 1–5.

DALMAZO, B. L. *et al.* Leveraging it Project Lifecycle Data to Predict Support Costs. In: IEEE. **Integrated Network Management (IM), 2011 IFIP/IEEE International Symposium on**. [S.l.], 2011. p. 249–256.

DALMAZO, B. L.; VILELA, J. P.; CURADO, M. Performance analysis of network traffic predictors in the cloud. **Journal of Network and Systems Management**, v. 25, n. 2, p. 290–320, Apr 2017.

DUNGAY, D. **Software Defined Networking Explained**. 2019. Disponível em: <https://www.commsbusiness.co.uk/features/software-defined-networking-sdn-explained/>.

ELATTAR, E. E.; GOULERMAS, J.; WU, Q. H. Electric Load Forecasting Based on Locally Weighted Support Vector Regression. In: **IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)**. [S.l.]: IEEE, 2010. v. 40, n. 4, p. 438–447.

ERMAN, J.; MAHANTI, A.; ARLITT, M. Qrp05-4: Internet Traffic Identification using Machine Learning. In: IEEE. **Global Telecommunications Conference, 2006. GLOBECOM'06. IEEE**. [S.l.], 2006. p. 1–6.

FANG, W.; PETERSON, L. Inter-AS traffic patterns and their implications. In: IEEE. **IEEE Global Telecommunications Conference (GLOBECOM)**. [S.l.], 1999. v. 3, p. 1859–1868.

FEAMSTER, N.; REXFORD, J.; ZEGURA, E. The Road to SDN: An Intellectual History of Programmable Networks. **ACM SIGCOMM Conference on Internet Measurement**, ACM, New York, NY, USA, v. 44, n. 2, p. 87–98, abr. 2014.

FRANK, J. Artificial Intelligence and Intrusion Detection: Current and Future Directions. In: BALTIMORE, MD. **Proceedings of the 17th national computer security conference**. [S.l.], 1994. v. 10, p. 1–12.

GERAVAND, S.; AHMADI, M. Bloom Filter Applications in Network Security: A State-of-the-art Survey. **Computer Networks**, Elsevier, v. 57, n. 18, p. 4047–4064, 2013.

GREGORI, E. *et al.* The Impact of IXPs on the AS-level Topo-logy Structure of the Internet. In: **Computer Communications**. [S.l.]: Elsevier, 2011. p. 68–82.

GUO, L.; MATTA, I. The War Between Mice and Elephants. In: IEEE. **Network Protocols, 2001. Ninth International Conference on**. [S.l.], 2001. p. 180–188.

GUPTA, A. *et al.* Sdx: A Software Defined Internet Exchange. In: **ACM SIGCOMM Conference on Internet Measurement**. [S.l.]: ACM, 2015. v. 44, n. 4, p. 551–562.

HARRISON, R. *et al.* Network-Wide Heavy Hitter Detection with Commodity Switches. In: **Proceedings of the Symposium on SDN Research**. New York, NY, USA: ACM, 2018. (SOSR '18), p. 8:1–8:7.

HONGYING, J.; LI, L. Dynamic Network Traffic Flow Prediction Model Based on Modified Quantum-Behaved Particle Swarm Optimization. **Journal of Networks**, Citeseer, v. 8, n. 10, p. 2332, 2013.

IPERF. **iPerf3 - The ultimate speed test tool for TCP, UDP and SCTP**. 2019. Disponível em: <https://iperf.fr/iperf-doc.php>.

IX-AUSTRALIA. **Australia Internet Exchange Point**. 2018. Disponível em: <https://www.ix.asn.au/>.

IX-AUSTRALIA-PEERING. **Australia Internet Exchange Point Peering**. 2018. Disponível em: <https://www.ix.asn.au/peering/>.

JAIN, R. **The Art of Computer Systems Performance Analysis: Techniques for Experimental Design, Measurement, Simulation, and Modeling**. [S.l.]: John Wiley & Sons, 1990.

JEYAKUMAR, V. *et al.* Millions of Little Minions: Using Packets for Low Latency Network Programming and Visibility. In: ACM. **ACM SIGCOMM Conference on Internet Measurement**. [S.l.], 2014. v. 44, n. 4, p. 3–14.

KNOB, L. A. D. *et al.* SDEFIX—Identifying Elephant Flows in SDN-based IXP Networks. In: IEEE. **IEEE/IFIP NOMS Network Operations and Management Symposium**. [S.l.], 2016. p. 19–26.

____. Mitigating Elephant Flows in SDN-based IXP Networks. In: IEEE. **IEEE ISCC Symposium on Computers and Communication**. [S.l.], 2017. p. 1352–1359.

LAI, H. *et al.* Simultaneous Feature Learning and Hash Coding With Deep Neural networks. In: **Proceedings of the IEEE conference on computer vision and pattern recognition**. [S.l.: s.n.], 2015. p. 3270–3278.

LI, Y. *et al.* Predicting inter-data-center network traffic using elephant flow and sublink information. In: **IEEE Transactions on Network and Service Management**. [S.l.]: IEEE, 2016. v. 13, n. 4, p. 782–792.

LIU, Z. *et al.* An Adaptive Approach for Elephant Flow Detection with the Rapidly Changing Traffic in Data Center Network. **International Journal of Network Management**, Wiley Online Library, v. 27, n. 6, p. e1987, 2017.

MARQUES, J. A.; GASPARY, L. P. Explorando Estratégias de Orquestração de Telemetria em Planos de Dados Programáveis. In: **Simpósio Brasileiro de Redes de Computadores (SBRC)**. [S.l.: s.n.], 2018. v. 36.

MCKEOWN, N. *et al.* OpenFlow: Enabling Innovation in Campus Networks. **ACM SIGCOMM Conference on Internet Measurement**, ACM, New York, NY, USA, v. 38, n. 2, p. 69–74, mar. 2008.

____. OpenFlow: Enabling Innovation in Campus Networks. In: **ACM SIGCOMM Conference on Internet Measurement**. NY, USA: ACM, 2008. p. 69–74. ISSN 0146-4833.

METCALFE, R. M.; BOGGS, D. R. Ethernet: Distributed packet switching for local computer networks. **Commun. ACM**, ACM, New York, NY, USA, v. 19, n. 7, p. 395–404, jul. 1976.

MININET. **Mininet - An Instant Virtual Network on your Laptop**. 2019. Disponível em: <https://github.com/mininet/mininet>.

MOORE, A. W.; ZUEV, D. Internet Traffic Classification using Bayesian Analysis Techniques. In: ACM. **ACM SIGMETRICS Performance Evaluation Review**. [S.l.], 2005. v. 33, n. 1, p. 50–60.

MORI, T. *et al.* On the characteristics of Internet Traffic variability: Spikes and Elephants. In: **IEICE TRANSACTIONS on Information and Systems**. [S.l.: s.n.], 2004. v. 87, n. 12, p. 2644–2653.

____. Identifying Elephant Flows Through Periodically Sampled Packets. In: ACM. **ACM SIGCOMM Conference on Internet Measurement**. [S.l.], 2004. (IMC '04), p. 115–120.

MOY, J. **OSPF version 2**. [S.l.], 1997.

NGUYEN, T.; ARMITAGE, G. Synthetic Sub-flow Pairs for Timely and Stable IP Traffic Identification. In: **Proc. Australian Telecommunication Networks and Application Conference**. [S.l.: s.n.], 2006.

NGUYEN, T. T.; ARMITAGE, G. A Survey of Techniques for Internet Traffic Classification using Machine Learning. **IEEE Communications Surveys & Tutorials**, IEEE, v. 10, n. 4, p. 56–76, 2008.

NSW-IX. **New South Wales Internet Exchange**. 2018. Disponível em: <https://bgpview.io/ix/320>.

OPEN-IX. **Open-IX Association**. 2018. Disponível em: <https://www.open-ix.org/cpages/membership>.

POSTEL, J. **Transmission control protocol**. [S.l.], 1981.

REKHTER, Y.; LI, T.; HARES, S. **A border gateway protocol 4 (BGP-4)**. [S.l.], 2005.

RESTREPO, J. C. C.; STANOJEVIC, R. IXP Traffic: a Macroscopic View. In: ACM. **the 7th Latin American Networking Conference**. New York, NY, USA, 2012. p. 1–8.

RYU. **Ryu SDN Framewaork**. 2019. Disponível em: <https://osrg.github.io/ryu/>.

SCHAAL, S.; ATKESON, C. G. Robot Juggling: Implementation of Memory-based Learning. **IEEE Control Systems**, IEEE, v. 14, n. 1, p. 57–71, 1994.

SFLOW.ORG. **sFlow**. 2018. Disponível em: <http://www-cs-faculty.stanford.edu/~uno/abcde.html>.

SIMONOFF, J. S. **Smoothing methods in statistics**. [S.l.]: Springer Science & Business Media, 2012.

SIVARAMAN, V. *et al.* Heavy-hitter detection entirely in the data plane. In: ACM. **Symposium on SDN Research**. [S.l.], 2017. p. 164–176.

SUH, J. *et al.* Opensample: A Low-latency, Sampling-based Measurement Platform for Commodity SDN. In: IEEE. **34th IEEE ICDCS International Conference on Distributed Computing Systems**. [S.l.], 2014. p. 228–237.

TONG, D.; PRASANNA, V. High Throughput Hierarchical Heavy Hitter Detection in Data Streams. In: IEEE. **22nd IEEE HiPC International Conference on High Performance Computing**. [S.l.], 2015. p. 224–233.

VIC-IX. **Victorian Internet Exchange**. 2018. Disponível em: <https://bgpview.io/ix/256>.

WA-IX. **West Australian Internet Exchange**. 2018. Disponível em: <https://www.peeringdb.com/ix/21>.

WAND, M. P.; JONES, M. C. **Kernel smoothing**. [S.l.]: Chapman and Hall/CRC, 1994.

WAND, M. P.; SCHUCANY, W. R. Gaussian-based kernels. In: **Canadian Journal of Statistics**. [S.l.]: Wiley Online Library, 1990. v. 18, n. 3, p. 197–204.

XIA, W. *et al.* A survey on software-defined networking. **IEEE Communications Surveys & Tutorials**, IEEE, v. 17, n. 1, p. 27–51, 2015.

XIAO, P. *et al.* An Efficient Elephant Flow Detection with Cost-sensitive in SDN. In: IEEE. **Industrial Networks and Intelligent Systems (INISCom), 2015 1st International Conference on**. [S.l.], 2015. p. 24–28.

ZHANG, Y. *et al.* On the Characteristics and Origins of Internet Flow Rates. In: **ACM SIGCOMM Conference on Internet Measurement**. New York, NY, USA: ACM, 2002. v. 32, n. 4, p. 309–322. ISSN 0146-4833.

ANEXO A ARTIGO PUBLICADO – SBRC 2018

Este trabalho apresenta um mecanismo inicial para realizar a identificação de fluxos elefantes diretamente no plano de dados programável de redes de ponto de troca de tráfego. Cada pacote é analisado por um *switch* ao ingressar na rede e o processo de identificação ocorre de forma imediata. O protótipo desenvolvido em P4 mostrou-se significativamente mais eficiente em relação às abordagens do estado da arte implementadas com o protocolo *OpenFlow*. Os resultados demonstraram que é possível identificar os fluxos elefantes de forma rápida e eficiente, com menos de 35% de falsos positivos e menos de 10% de falsos negativos, em um cenário cujos recursos de memória eram escassos.

- **Título:**
  Identificação de Fluxos Elefantes em Redes de Ponto de Troca de Tráfego com Suporte à Programabilidade P4
- **Conferência:**
  XXXVI Simpósio Brasileiro de Redes de Computadores e Sistemas Distribuídos (SBRC)
- **Tipo:**
  Trilha principal (*full-paper*)
- **Qualis:**
  B2
- **URL:**
  <https://portaldeconteudo.sbc.org.br/index.php/sbrc/article/view/2483>
- **Data:**
  6 a 10 de maio de 2018
- **Realizado em:**
  Campos do Jordão - SP, Brasil

Identificação de Fluxos Elefantes em Redes de Ponto de Troca de Tráfego com Suporte à Programabilidade P4

Marcus Vinicius Brito da Silva[1], **Jonatas Adilson Marques**[1], **Luciano Paschoal Gaspary**[1], **Lisandro Zambenedetti Granville**[1]

[1]Instituto de Informática – Universidade Federal do Rio Grande do Sul (UFRGS)
Caixa Postal 15.064 – 91.501-970 – Porto Alegre – RS – Brasil

{mvbsilva, jamarques, paschoal, granville}@inf.ufrgs.br

Abstract. *In view of the challenges encountered in management the flows that transit over a traffic exchange network, the identification of called elephants flows can contribute to the quality of services provided to its participants. In this perspective, taking advantage of the resources found in switches with programmable support, this work presents a mechanism to realize the identification of elephant flows directly in the programmable data plane of the network. Each packet is analyzed by a switch when it ingresses the network and the identification process occurs in immediate. The prototype developed in P4 showed up significantly more efficient than the state-of-the-art approaches implemented with the OpenFlow protocol. The results showed that it is possible to identify elephant flows quickly and efficiently, with less than 35% false positives and less than 10% false negatives, in a scenario where memory resources were scarce.*

Resumo. *Diante dos desafios encontrados no gerenciamento de fluxos que trafegam sobre uma rede de ponto de troca de tráfego, a identificação dos chamados fluxos elefantes pode contribuir na qualidade dos serviços prestados aos seus participantes. Nessa perspectiva, aproveitando os recursos encontrados em switches com suporte a programabilidade, este trabalho apresenta um mecanismo para realizar a identificação de fluxos elefantes diretamente no plano de dados programável da rede. Cada pacote é analisado por um switch ao ingressar na rede e o processo de identificação ocorre de forma imediata. O protótipo desenvolvido em P4 mostrou-se significativamente mais eficiente em relação às abordagens do estado da arte implementadas com o protocolo OpenFlow. Os resultados demonstraram que é possível identificar os fluxos elefantes de forma rápida e eficiente, com menos de 35% de falsos positivos e menos de 10% de falsos negativos, em um cenário cujos recursos de memória eram escassos.*

1. Introdução

Pontos de Troca de Tráfego (PTT) conectam sistemas autônomos da internet (ASs) e permitem que os provedores de serviços realizem troca de tráfego com a finalidade de melhor atender seus clientes [Knob et al. 2016]. PTTs são distribuídos por todo o mundo e desempenham um papel essencial no ecossistema da internet, contabilizando pelo menos 20% de todo o tráfego trocado entre ASs [Cardona Restrepo and Stanojevic 2012]. A infraestrutura de um PTT pode ser caracterizada como: simples, quando composta por um único dispositivo de comutação, ou complexa, quando formada por várias redes, com topologias intricadas [Augustin et al. 2009].

Um dos principais benefícios experimentados pelos participantes de um PTT é o baixo custo de implantação e manutenção [Gregori et al. 2011]. Esses fatores estimulam empresas (como Google e Netflix) a conectarem suas infraestruturas aos principais PTTs, buscando assim reduzir os custos de conexão de seus clientes com seus serviços [Knob et al. 2017]. Não surpreendentemente, desafios de gerenciamento são evidenciados no contexto de redes de PTT, tais como: monitoramento de tráfego *Address Resolution Protocol* (ARP), estabelecimento de *Virtual Private Network* (VPN), ou estratégias para lidar com fluxos críticos e seus caminhos na rede. Diante disso, propostas têm sido apresentadas abordando esses desafios em redes de PTT utilizando *Software Defined Networking* (SDN). Este paradigma tem contribuído fortemente para a evolução das redes de computadores e muitas investigações têm sido realizadas sobre os possíveis benefícios de sua adoção [Wickboldt et al. 2015]. Suas principais características são evidenciadas no encaminhamento de pacotes baseado em fluxos e na abstração da lógica de controle para um *software*, chamado de controlador [Araujo et al. 2017].

Diante dos desafios encontrados para gerenciar os fluxos que trafegam sobre a infraestrutura, a identificação dos fluxos chamados elefantes [Guo and Matta 2001] pode favorecer uma melhor utilização dos serviços prestados em uma rede de PTT. Um fluxo é considerado um elefante se tiver duração e volume de tráfego significativamente altos segundo limiares predefinidos [Guo and Matta 2001]. Esses fluxos representam a maior parte do volume do tráfego da infraestrutura, apesar de tenderem a ser uma fatia pequena dentre todos os fluxos. Como consequência, os fluxos elefantes podem impactar significativamente no tráfego de fluxos menores, aqueles cujo comportamento não excede os limiares. Em resposta a esse agravante, propostas como *SDEFIX* [Knob et al. 2016] e *OpenSample* [Suh et al. 2014] apresentam mecanismos para identificação de fluxos elefantes em redes de PTT que fazem uso de SDN/OpenFlow. Contudo, essas abordagens extraem amostras dos fluxos no plano de dados e realizam o processo de análise fora dele, no plano de controle. Esse procedimento implica diretamente em atrasos no processo de identificação, uma vez que a comunicação entre *switch* e controlador gera atraso no processamento das estatísticas no plano de controle. Tal como será apresentado ao decorrer desse trabalho, esse atraso implica fortemente no tempo de reação para mitigar os efeitos de um fluxo elefante sobre os fluxos menores.

Abordagens para lidar com o atraso de comunicação entre *switch* e controlador são viabilizadas a partir das redes com planos de dados programáveis, uma vez que a programabilidade dos *switches* pode favorecer que estes realizarem o processo de análise de fluxos de forma interna [Basat et al. 2017]. Nessa perspectiva, trabalhos têm explorado os recursos de *switches* programáveis emergentes [Bosshart et al. 2014] para analisar fluxos diretamente no plano de dados [Sivaraman et al. 2017][Basat et al. 2017], realizando procedimentos sobre os pacotes além do processamento tradicional de encaminhamento. A programabilidade dos *switches* permite que mecanismos sejam implementados para que os pacotes sejam analisados logo ao ingressarem na rede. Nesse sentido, esse trabalho apresenta uma proposta para realizar identificação de fluxos elefantes em redes de PTT com plano de dados programável, buscando contribuir para o gerenciamento de fluxos nesse contexto.

O mecanismo desenvolvido realiza a contagem do volume de tráfego e a duração dos fluxos de forma imediata, para cada pacote que ingressa na rede. Ou seja, juntamente

com o processo tradicional de encaminhamento, instruções são realizadas para obter as informações de volume e duração para cada pacote processado. Essas informações são armazenadas a partir de indexação com chaves *hash*, para identificar os fluxos individualmente. Isso permite que cada fluxo seja analisado tendo suas informações indexadas por um conjunto de características que definem sua singularidade. Como exemplo, uma tupla formada pelos endereços IP de origem e destino, protocolo da camada de transporte e portas de aplicação origem e destino pode ser utilizada para gerar uma chave de indexação. Para aumentar a acurácia do mecanismo, a qual pode ser comprometida com o número de colisões, foi adotado um mapeamento *hash* de múltiplos níveis. Dessa forma, pelo menos duas chaves são geradas por diferentes funções *hash* para indexar os contadores que armazenam as informações sobre os fluxos, em diferentes tabelas.

Contudo, caso ocorram colisões, os contadores de diferentes fluxos podem ser acumulados, levando à caracterização indevida, incindindo em falsos positivos ou falsos negativos. Uma vez que o propósito deste trabalho está em identificar os fluxos elefante que trafegam pela rede, é preferível que ocorram falsos positivos em detrimento de falsos negativos. Ou seja, ao ocorrer colisões nas chaves *hash*, é preferível que um fluxo venha ser identificado como elefante sem este sê-lo, do que um fluxo que de fato é elefante não seja identificado. Para garantir isso, o mecanismo de análise e armazenamento das informações de fluxos admite um comportamento onde estes são superestimados e não subestimados. Esse processo é descrito em mais detalhes na seção 3.

Os resultados obtidos nos testes experimentais apontaram que o protótipo desenvolvido em P4 se mostrou significativamente mais eficiente em relação ao mecanismo implementado com o protocolo *OpenFlow*, baseado em trabalhos anteriores. No protótipo em P4, todo o processo de análise e identificação dos fluxos elefantes é realizado diretamente nos *switches*. Isso permite analisar cada pacote que chega no *switch*, além de realizar a identificação de forma imediata com o pacote que excede os limiares de volume e duração. Os resultados demonstram que é possível identificar e reagir aos fluxos elefantes de forma rápida (menos de 0.5 ms) e eficiente, com apenas 21.5 KB de dados inseridos na rede pelo mecanismo, para um cenário com até 50% de fluxos elefantes sobre o total de fluxos na rede. Além disso, o mecanismo obteve menos de 35% de falsos positivos e menos de 10% de falsos negativos para os cenários mais extremos, quando o espaço de mapeamento *hash* era significativamente menor em relação à quantidade de fluxos analisados pelos *switches*.

O restante desse trabalho está organizado da seguinte forma. A seção 2 discute os trabalhos relacionados. A seção 3 descreve o mecanismo para realizar a identificação dos fluxos elefantes. A seção 4 apresenta a avaliação e os resultados obtidos. Finalmente, a seção 5 expõe as principais conclusões do estudo e perspectivas de trabalhos futuros.

2. Trabalhos Relacionados

Nessa seção são apresentados os trabalhos relacionados, e uma breve discussão é realizada sobre suas abordagens e estratégias utilizadas, bem como as limitações em relação à proposta deste trabalho. Primeiro é destacada a aplicação de SDN no cenário de redes de PTT e na sequência são descritas as abordagens para realizar análise e monitoração de fluxos neste contexto.

2.1. Utilização de SDN em Redes de PTT

A utilização de redes definidas por *software* em pontos de troca de tráfego é apresentada em SDX [Gupta et al. 2015] como proposta aos problemas de redes tradicionais, como: limitações do protocolo BGP (*Border Gateway Protocol*), limitações de políticas e seleção de rotas para escoamento de tráfego. Em SDX, os ASs participantes executam aplicações SDN em um controlador virtual e as políticas geradas são combinadas em políticas de escoamento e implantadas na infraestrutura do PTT por um controlador centralizado. Contudo, ainda que permita realizar balanceamento de carga, esta abordagem não lida explicitamente com o problema dos fluxos elefantes na rede.

Outros trabalhos abordam o problema de fluxos elefantes em redes de PTT fazendo uso de SDN/OpenFlow. Em *DevoFlow* [Curtis et al. 2011], os fluxos são amostrados e contabilizados em termos de *bytes*. Quando um limiar predefinido é atingido, o fluxo é classificado como elefante e um algoritmo de roteamento é aplicado para calcular o caminho menos congestionado entre seus pontos finais. Em *OpenSample* [Suh et al. 2014], é utilizado *sFlow* para realizar o monitoramento dos fluxos. Nessa abordagem, as taxas de fluxo são calculadas subtraindo os números de sequência TCP (*Transmission Control Protocol*) de duas amostras do mesmo fluxo e dividindo o valor pelo tempo entre elas. Quando um fluxo é classificado como elefante, ele é redirecionado pelo controlador SDN para outro caminho usando um algoritmo de ajuste global.

Em SDEFIX [Knob et al. 2016], a amostragem de pacotes também ocorre utilizando sFlow e um módulo de identificação é alimentado para realizar a classificação dos fluxos baseado em regras predefinidas pelo operador da rede. De forma semelhante às abordagens já mencionadas, essa proposta realiza a identificação dos fluxos elefantes no plano de controle a partir de amostragens de fluxos obtidas no plano de dados. Esse fator, além de implicar em atraso no processo de análise, também incorre em um volume adicional de dados de monitoração sobre a rede. De modo diferente, a proposta apresentada neste trabalho realiza todo o processo de identificação dos fluxos diretamente no plano de dados programável, a partir de procedimentos implementados diretamente nos *switches*.

2.2. Monitoramento de Fluxos no Plano de Dados Programável

O monitoramento de fluxos diretamente no plano de dados programável é apresentado em [Sivaraman et al. 2017], onde esse processo é realizado em um *backbone* com taxa de até 100 Gbps. O objetivo desse trabalho é realizar a identificação de *Heavy Hitters* inteiramente no plano de dados, utilizando *switches P4*. *Heavy Hitters* podem ser considerados todos os fluxos maiores seja em quantidade de bytes ou número de pacotes em relação aos demais fluxos em um *link* ou porta de um *switch* [Sivaraman et al. 2017]. Em [Basat et al. 2017], é apresentada uma proposta que realiza essa análise utilizando um mecanismo de agregação de endereços IP (*Internet Protocol*) a partir de sua hierarquia. Nestes trabalhos, os fluxos são mapeados por funções *hash* para serem identificados e analisados a partir de sua chave. Os endereços IP de origem e destino, protocolo da camada de transporte e portas de aplicação origem e destino formam uma tupla que é mapeada para um número de identificação do fluxo. A análise do fluxo é realizada contabilizando o volume de tráfego, indexado a partir desta chave.

Em suma, essas estratégias habilitaram a análise e monitoramento de fluxos diretamente no plano de dados programável. Contudo, não é apresentado um mecanismo

voltado aos aspectos de fluxos elefantes, que envolvem tanto as características de volume quanto a duração dos fluxos. Ainda, estas propostas não são apresentadas em um contexto de redes de PTT, onde o tempo para a identificação desses fluxos pode comprometer o bom desempenho dos serviços prestados pela rede. Neste sentido, como será descrito na sequência, a proposta desse trabalho consiste em realizar o processo de identificação dos fluxos elefantes diretamente no plano de dados programável de uma rede PTT, utilizando os recursos encontrados em *switches P4*. Com isso, deseja-se contribuir para o melhor gerenciamento de tráfego no contexto dessas infraestruturas.

3. Identificação de Fluxos Elefantes em PTT com *Switches* Programáveis

Entre os desafios encontrados no gerenciamento de redes de PTT, a identificação dos fluxos ditos elefantes pode contribuir fortemente, entre outros aspectos, para tomadas de decisões que venham resultar no melhor funcionamento da infraestrutura. No contexto de um PTT, estes fluxos maiores podem ser administrados de modo que não venham a impactar no escoamento de fluxos menores. A qualidade de serviço prestada pela rede pode ser um dos fatores a serem garantidos a partir de um mecanismo que permita realizar a identificação destes fluxos de forma rápida e eficiente. Nesse sentido, considerando uma rede de PTT com plano de dados programável, este trabalho apresenta um mecanismo que permite realizar a análise e identificação de fluxos elefantes diretamente nos *switches*.

O cenário ilustrado na Figura 1 abstrai a infraestrutura de um PTT, formada por *switches* com suporte a programabilidade e a figura de um controlador SDN. O controlador, por sua vez, tendo uma visão global da infraestrutura, tem o papel de fazer a interface entre operador da rede e o plano de dados. Com isso, informações podem ser enviadas do plano de dados para o plano de controle informando alguma anormalidade identificada, permitindo que medidas sejam tomadas rapidamente para mitigar seus efeitos. Esses elementos habilitam o processo de análise e identificação de fluxos elefantes, tal como será descrito na sequência.

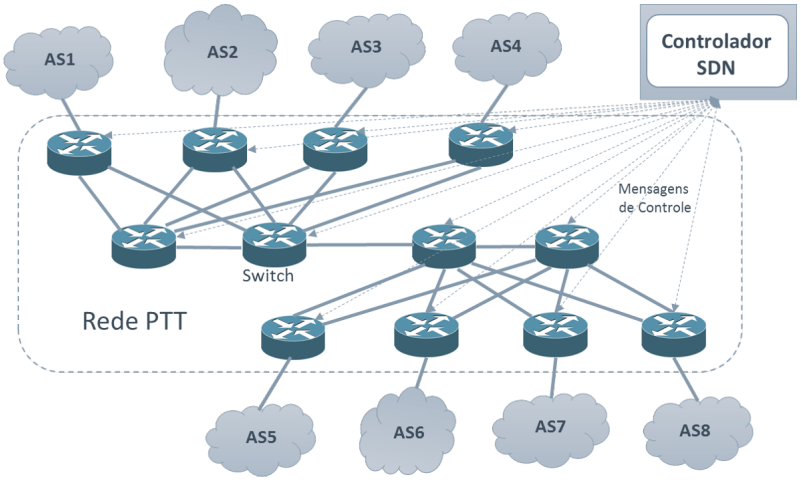


Figura 1. Cenário de uma rede de PTT com *switches* programáveis.

O mecanismo apresentado neste trabalho realiza a contagem de volume de tráfego e duração dos fluxos armazenando essas informações a partir de indexação *hash*, mapeando cada fluxo individualmente. Assim, cada fluxo pode ser analisado tendo suas

informações indexadas por um conjunto de características que definem sua singularidade. Como exemplo, uma tupla formada pelos endereços IP de origem e destino, protocolo da camada de transporte e portas de aplicação origem e destino pode ser utilizada para gerar uma chave de indexação. Logo, para cada pacote que chega em um *switch*, além do processamento tradicional de encaminhamento, suas características de volume e duração são contabilizadas e armazenadas em contadores indexados com as chaves de seu fluxo.

Para minimizar a possibilidade de agregar informações de fluxos distintos em contadores com o mesmo indexador, cenário que pode ocorrer quando há colisões no mapeamento *hash*, é adotado um mapeamento de múltiplos níveis [Tong and Prasanna 2015]. Ou seja, é utilizado mais de um indexador para um mesmo fluxo e estas chaves são geradas a partir de diferentes funções *hash*. Este mecanismo é ilustrado na Figura 2, onde k_i é mapeado por diferentes funções *hash* ($h_1, h_2, ...$). Assim, os dados a_i são acumulados em contadores com posições distintas, em diferentes tabelas. Se ocorrer uma colisão, existe a possibilidade de pelo menos um contador possuir os valores reais do fluxo desejado. Ou no caso mais extremo, onde todos os indexadores possam vir a colidir com indexadores de outros fluxos, ainda existe a possibilidade de obter o valor mais próximo do valor real.

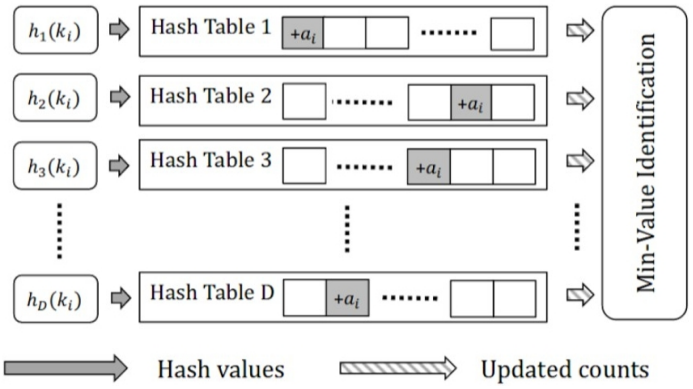


Figura 2. Mapeamento *hash* de múltiplos níveis [Tong and Prasanna 2015].

Quando um pacote chega em um *switch*, pelo menos duas chaves são geradas para indexar os contadores que armazenarão as informações sobre o fluxo. Se vier a ocorrer colisão em apenas uma das chaves com um fluxo iniciado posteriormente e seus valores vierem a ser acumulados, a outra chave estará indexando o contador que deve possuir os valores reais de cada fluxo. A partir disso, o volume de tráfego que será utilizado para julgar o fluxo será aquele cujo contador possui o menor valor registrado. Isso permite reduzir a probabilidade de ocorrer falsos positivos e falsos negativos.

No entanto, para lidar com o aspecto temporal dos fluxos é necessário armazenar o tempo de chegada do último pacote, de modo a obter a diferença com o tempo de chegada do próximo pacote. Esse procedimento permite determinar se um fluxo continua ativo ou foi reiniciado. Ou seja, é possível determinar um *timeout* e quando a diferença entre o novo pacote e seu anterior exceder esse tempo, pode-se concluir que este é um novo fluxo e só então os contadores poderão ser reiniciados. Essa estratégia também permite evitar que os valores de um fluxo que já está sendo armazenado seja reiniciado caso ocorra uma colisão com as chaves de um novo fluxo. Caso os contadores fossem reiniciados, o fluxo mais antigo (o qual possui maior possibilidade de se tornar um fluxo elefante) passaria a

assumir as informações temporais de um fluxo mais novo. Isso poderia implicar em falsos negativos, quando um fluxo elefante não é identificado.

Ainda, se ambas as chaves de um novo fluxo vierem a colidir com as chaves de um fluxo ainda ativo, os contadores não serão reiniciados e o novo fluxo passará a assumir os valores já contabilizados para o fluxo anterior. Dessa maneira os fluxos são superestimamos, mas não subestimamos. Ou seja, é aceitável que haja falsos positivos em detrimento de falsos negativos, já que o interesse está efetivamente em identificar os fluxos elefantes, uma vez que estes podem implicar fortemente no desempenho de fluxos menores. Entretanto, a possibilidade de colisões de múltiplas chaves é minimizada com o aumento do espaço para o mapeamento *hash*. Muito embora a limitação de memória seja um grande desafio no contexto dos dispositivos físicos, nesta proposta o espaço de memória dedicado nos *switches* para a utilização no mecanismo de identificação pode ser adaptado segundo a acurácia desejada.

Uma vez que os valores a serem utilizados para julgar os fluxos são definidos, estes são comparados com limiares que têm o papel de caracterizar o fluxo como elefante ou não. Estes processos, tal como o processo de coleta das informações dos fluxos, são realizados inteiramente nos *switches* programáveis. Vale ressaltar que apenas os *switches P4* que compõem a borda da infraestrutura implementam o mecanismo para identificação de fluxos elefantes. Os *switches* do núcleo da rede realizam apenas o tradicional encaminhamento de pacotes. Isso ocorre para evitar sobrecarga no núcleo e possibilita, ainda, que os fluxos sejam analisados logo ao ingressarem na rede. Os limiares, por sua vez, são definidos globalmente e podem ser atualizados dinamicamente pelo operador da rede, permitindo um gerenciamento mais flexível. Contudo, o processo de gerenciamento e definição dos valores para os limiares estão fora do escopo deste trabalho.

Quando um *switch* caracteriza um fluxo como elefante, uma notificação é enviada ao plano de controle informando a nova identificação. Um pacote contendo as informações que caracterizam aquele fluxo é enviado ao controlador para que este tome ciência do fluxo identificado. A partir disso, ações podem ser tomadas no plano de controle para tratar este fluxo seguindo as políticas definidas no PTT. Como o processo para mitigar os efeitos dos fluxos elefantes não compreende o escopo deste trabalho, foi elaborada uma política para estabelecer apenas um caminho alternativo para estes fluxos, na tentativa de reduzir seus impactos sobre os fluxos menores que estariam compartilhando uma mesma rota. Deste modo, quando um fluxo elefante é identificado, além do *switch* enviar a notificação ao plano de controle, este passa a encaminhar o fluxo identificado imediatamente por sua rota alternativa. Os detalhes da implementação e avaliação do mecanismo são apresentados na seção a seguir.

4. Avaliação

Nesta seção é apresentada a metodologia de avaliação, bem como os resultados obtidos a partir dos experimentos realizados. Para efeito de comparação foi desenvolvida uma aplicação com a mesma finalidade da proposta deste trabalho, implementada com o protocolo *OpenFlow* e baseada nos trabalhos relacionados. Ou seja, foi desenvolvido um mecanismo para analisar os fluxos e identificar aqueles que são elefantes no contexto de redes de PTT utilizando SDN/*Openflow*. Nesse mecanismo, as informações sobre os fluxos são obtidas a partir de troca de mensagens sobre os *status* das regras instaladas

nos *switches*. O controlador realiza requisições do tipo *FlowStatsRequest* para obter as informações contabilizadas para cada regra de encaminhamento instalada em um *switch*. Essa solicitação é respondida com uma mensagem do tipo *FlowStatsReply*, contendo as informações de duração, volume em bytes e quantidade de pacotes para cada regra.

Uma vez que são atribuídas regras de encaminhamento para cada fluxo, é possível obter as informações que os caracterizam individualmente. Neste trabalho, o intervalo entre dois ciclos de requisição e resposta é chamado de intervalo de amostragem. Esse recurso disponível no protocolo *OpenFlow* foi utilizado pois a contagem das informações é realizada sobre cada pacote em cada regra de encaminhamento. Assim, este mecanismo admite um comportamento semelhante à proposta desenvolvida em P4, a qual realiza o processo de análise para cada pacote que passa pelo *switch*.

O cenário de teste utilizado nos experimentos abstrai a infraestrutura de uma rede de ponto de troca de tráfego, tal como ilustrado na Figura 3. É possível observar que 8 ASs estão conectados na rede do PTT pelos *switches* de borda, e estes possuem pelo menos dois caminhos de alcance para o núcleo da rede. No cenário de comparação com *OpenFlow*, de forma semelhante ao mecanismo em P4 (que é implementado apenas nos *switches* de borda), apenas os *switches* de borda são consultados para se obter as informações sobre as regras de encaminhamento. Essa condição é suficiente para o funcionamento do mecanismo apresentado, já que os pacotes são analisados assim que ingressam na rede.

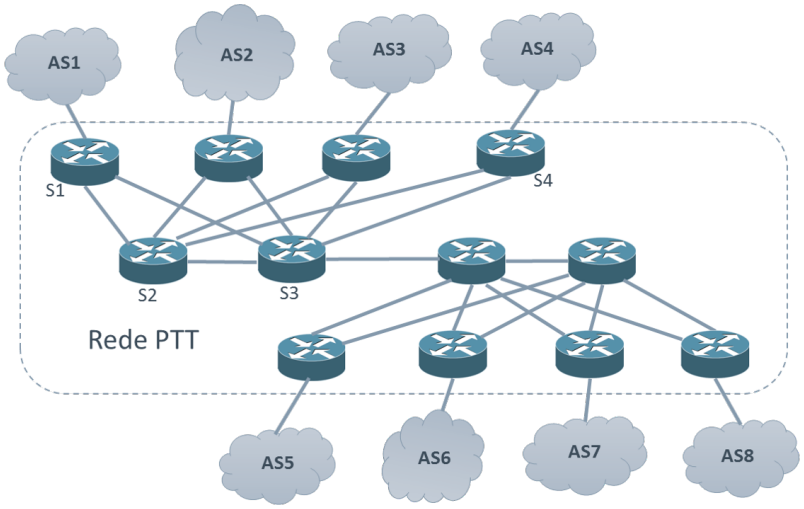


Figura 3. Cenário de Avaliação.

Para realizar os experimentos, foram criados fluxos TCP entre os ASs seguindo uma configuração cliente e servidor. Os fluxos foram gerados com base em dois fatores: volume e duração. O volume foi estabelecido a partir da largura de banda utilizada pelo fluxo, variando entorno de 5 Mbps. Já para a duração, foi determinado 13 segundos para os fluxos elefantes e 5 segundos para os demais fluxos. Os limiares para caracterizar os fluxos como elefantes foram fixados em 5 MB e 10 segundos. Esses valores são baseados em trabalhos anteriores, [Knob et al. 2016] [Knob et al. 2017]. Um total de 448 fluxos foram gerados na rede a cada rodada de teste, com 10 minutos de duração. Quando um fluxo é iniciado, uma rota padrão é estabelecida pelo controlador a partir de um algoritmo de menor caminho (por exemplo: AS1-S1-S2-S4-AS4) e quando um fluxo é identificado

como elefante, este é redirecionado por um caminho alternativo (por exemplo: AS1-S1-S3-S4-AS4), Figura 3. Os aspectos de otimização relacionados às rotas e aos limiares estão fora do escopo desse trabalho.

De modo a priorizar os requisitos de tempo no contexto de uma rede de PTT, onde as decisões/ações devem ser tomadas de forma rápida, foi desenvolvida uma abordagem preventiva. Ou seja, dado que um fluxo é iniciado, o controlador ao inserir as regras de encaminhamento para a rota padrão também insere as regras de encaminhamento para a rota alternativa. Assim, quando um fluxo é identificado como elefante, o *switch P4* pode imediatamente redirecioná-lo pelo caminho alternativo. No cenário com *OpenFlow*, essa medida preventiva foi implementada de modo que apenas uma regra de encaminhamento precise ser adicionada pelo controlador no *switch* pelo qual o fluxo entra na rede. Da mesma forma, ao receber a notificação de um novo fluxo, o controlador *OpenFlow* também instala a rota alternativa, além da rota padrão. Quando um fluxo é identificado com elefante, o controlador insere uma nova regra no *switch* de borda para marcar o fluxo como identificado (por exemplo, setar uma *flag* do cabeçalho IP) e encaminhá-lo pela rota alternativa. É evidente que esta abordagem carece de mais recursos de memória para as tabelas de encaminhamento. Contudo, é possível que exista um cenário onde esse custo é tolerável em detrimento do tempo para identificação dos fluxos elefantes, já que eles podem impactar fortemente no desempenho dos demais fluxos.

Os experimentos foram realizados em um computador equipado com: processador Intel Core i7-4790 com 8 núcleos de 3.6 GHz; 16 GB de memória RAM; sistema operacional *Linux Ubuntu 16.04 LTS*. O protótipo dessa proposta foi implementado na linguagem $P4_{16}$ e utilizando o *switch* de *software P4* versão 2[1]. O mecanismo de comparação foi desenvolvido com *RYU SDN Framework* versão 4.2, utilizando o protocolo *OpenFlow* na versão 1.3 e *Open vSwitch* versão 2.0.2. A infraestrutura foi emulada utilizando *Mininet* versão 2.3, onde os *links* foram fixados com 100 Mbps de largura de banda e sem atraso de propagação. Para gerar a cargar de trabalho, foi utilizada a ferramenta *iPerf* versão 3.0.11. Para gerar o *log* dos fluxos foi utilizada a ferramenta *Wireshark* versão 2.2.6.

O desempenho da proposta implementada em P4 foi comparado ao mecanismo desenvolvido com *OpenFlow* a partir de quatro métricas: (i) tempo de reação, (ii) volume de *bytes* excedentes, (iii) volume de monitoração e (iv) acurácia. Por tempo de reação, é considerado o tempo em que o primeiro pacote do fluxo identificado como elefante sai pela rota alternativa, menos o tempo em que o pacote responsável por exceder os limiares chegou no *switch*. Por volume de *bytes* excedentes, é considerada a quantidade de *bytes* que trafegaram na rota padrão desde que o fluxo tornou-se um elefante (chegada do pacote que excede os limiares). Por volume de monitoração, é considerado o volume inserido na rede com o mecanismo de análise e identificação. Para a proposta em P4, este é volume de notificações enviadas ao controlador. Já no mecanismo desenvolvido em *OpenFlow*, este é o volume das mensagens de requisição sobre as informações dos fluxos nos *switches* e sua reposta. Por fim, para a acurácia, considera-se o número de falsos positivos: o percentual dos fluxos identificados como elefantes indevidamente; e falsos negativos: o percentual dos fluxos que eram elefantes e não foram identificados, em função do espaço para mapeamento *hash*. Essas métricas são consideradas melhores, ao passo que admitem valores menores.

[1] https://github.com/p4lang/behavioral-model

4.1. Resultados

Os resultados obtidos nos experimentos são apresentados segundo as métricas descritas anteriormente. Para efeito de comparação, o mecanismo desenvolvido em *OpenFlow* também foi avaliado com uma estratégia reativa. Neste caso, o controlador só insere a rota alternativa para um fluxo elefante quando este é identificado. Os resultados obtidos com a proposta implementada em *switches P4* estão descritas com a abreviatura "P4", para o mecanismo desenvolvido com *OpenFlow*, é utilizada a abreviatura "OFP", para a estratégia preventiva e "OFR", para a estratégia reativa.

A Figura 4 ilustra o tempo médio de reação aos fluxos identificados como elefantes. No eixo x, estão as abordagens avaliadas, sendo que as abordagens implementadas com *OpenFlow* foram analisadas com intervalo de amostragem variando em: 0.5, 1 e 5 segundos. No eixo y, em escala logarítmica, é apresentado o tempo médio de reação (em milissegundos) para cada uma das abordagens avaliadas. É possível observar que o mecanismo em P4 consegue identificar e reagir a um fluxo elefante com tempo médio de 0.4 ms. Sendo este basicamente o tempo para o processamento do pacote pelo *switch P4*.

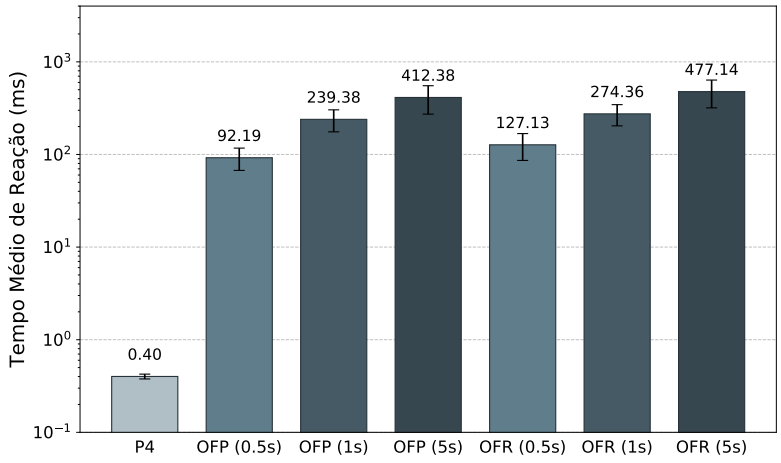


Figura 4. Tempo de reação.

De modo diferente, as abordagens que utilizam *OpenFlow* admitem uma variação entre 92.19 ms, no melhor caso, e até 477.14 ms, no pior caso. Uma vez que estas abordagens dependem do envolvimento do controlador, essa diferença pode ser justificada pelo tempo de comunicação entre os *switches* e a unidade de controle. Já que os intervalos de confiança nas abordagens proativa e reativa se sobrepõem em seus níveis correspondentes, não é possível admitir que há diferença a um nível de confiança em 95%.

A segunda métrica diz respeito ao volume de dados que excederam os limiares e continuaram a trafegar pela rota padrão, até que a reação tenha ocorrido. Na Figura 5 é possível observar que a implementação em P4 não possui dados excedentes, já que a reação ocorre de forma imediata com o processamento do pacote que caracteriza o fluxo como elefante. No entanto, nas abordagens com *OpenFlow* há uma média de quase 0.9 MB de dados excedentes no pior caso, quando o intervalo de amostragem é de 5s. Esses valores são consequência do tempo de comunicação entre o controlador *OpenFlow* e os switches para reagir a uma identificação, além do intervalo entres as mensagens

de monitoração. Ainda, é possível observar uma leve sobreposição dos intervalos de confiança entre as abordagens proativa e reativa a um nível de confiança em 95%.

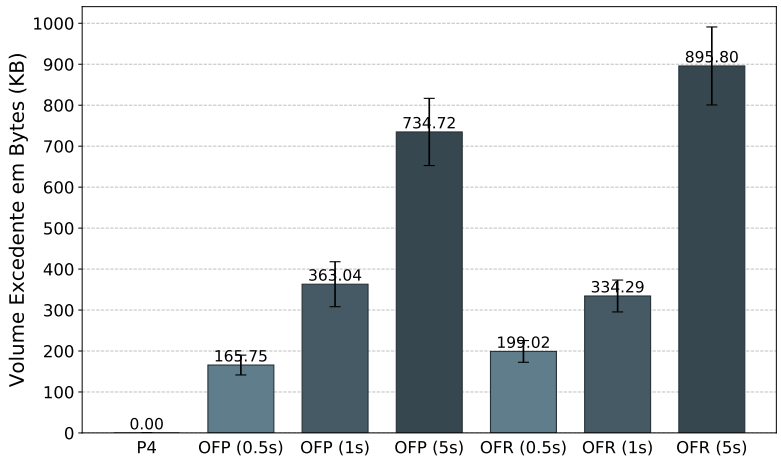
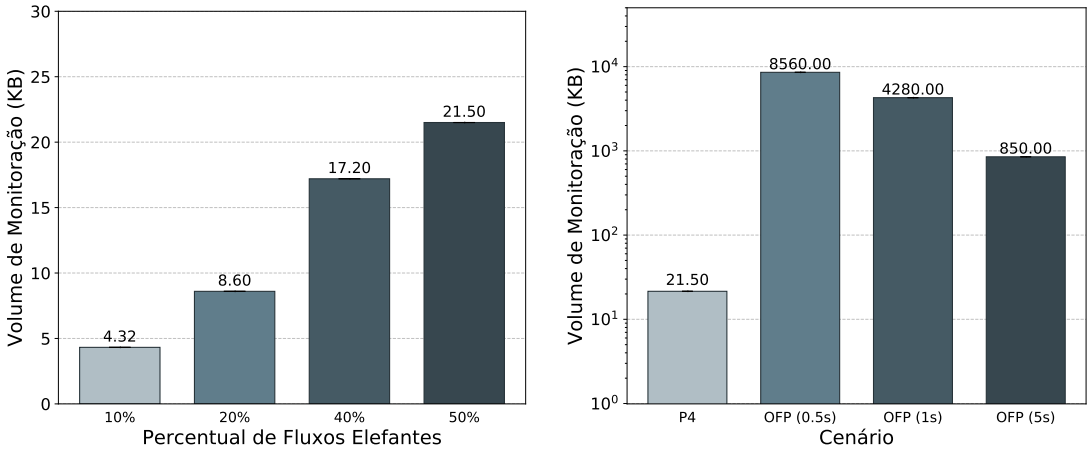


Figura 5. Volume de *bytes* excedentes.

Outra métrica significativa diz respeito ao volume de dados inserido na rede com o mecanismo de monitoração. Uma vez que a proposta em P4 envia apenas uma notificação informando o fluxo identificado ao plano de controle (pacote de 96 *bytes*), o volume de dados inseridos na rede é diretamente proporcional à quantidade de fluxos elefantes identificados. Na Figura 6(a) é possível observar a quantidade de dados inseridos na rede quando o percentual de fluxos elefantes varia entre 10, 20, 40 e 50% do número total de fluxos (cerca de 448). Esses valores crescem proporcionalmente, sendo o valor observado quando se tem 50% de fluxos elefantes cinco vezes maior se comparada quando há 10%.



(a) Implementação em P4

(b) P4 x OpenFlow

Figura 6. Volume de dados de monitoração.

Na Figura 6(b) é possível observar que existe uma diferença significativa entre a proposta em P4 e as abordagens com *OpenFlow*. É destacado o cenário com 50% de fluxos elefantes em relação ao número total de fluxos na rede, já que este permite realizar

uma análise considerando situações mais extremas da rede. Nas abordagens com *OpenFlow*, a medida em que o intervalo de amostragem aumenta (5s) o volume de monitoração diminui. Contudo, isso implica no aumento do tempo de reação (ver Figura 4), uma vez que são grandezas inversamente proporcionais. Por fim, não há diferença entre as abordagens *OpenFlow* proativa e reativa, já que a quantidade de mensagens é proporcional ao intervalo de amostragem e a duração dos experimentos, que foi fixada em 10 minutos.

A acurácia do mecanismo foi avaliada variando o espaço para o mapeamento *hash* nos *switches P4* em relação a quantidades de fluxos inseridos na rede. Na Figura 7, é possível observar que com apenas 25% de espaço, obtém-se um percentual de aproximadamente 5% de falsos negativos e pouco menos de 35% de falsos positivos. Esse comportamento é resultado da estratégia que superestima os fluxos, de modo que é preferível obter falsos positivos em relação à falsos negativos quando ocorrem colisões de chaves *hash*. É evidente que a medida em que o espaço para mapeamento *hash* aumenta (por exemplo, 75%), os falsos positivos e falsos negativos diminuem (prox. de 10% e 1%, respectivamente) ao passo que os verdadeiros positivos e verdadeiros negativos aumentam, admitindo cerca de 95% de acurácia.

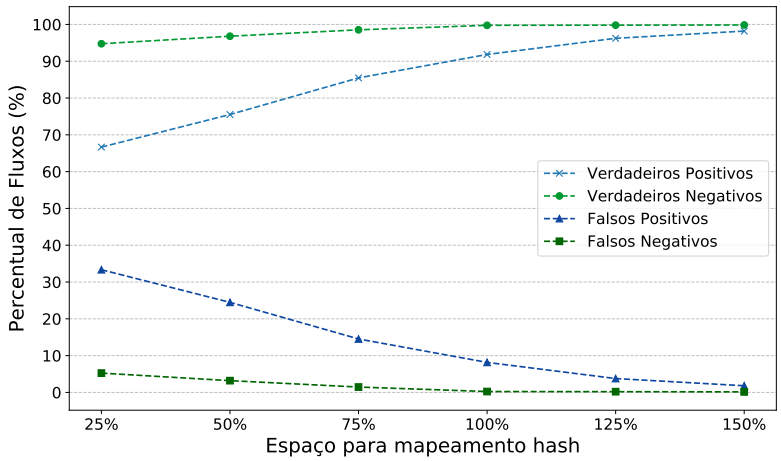


Figura 7. Acurácia do mecanismo em função do espaço de mapeamento Hash.

A Tabela 1 destaca a diferença no tempo médio de processamento dos pacotes com a implantação do mecanismo nos *switches P4*, um aumento de aproximadamente 30%. O tempo médio de processamento dos pacotes nos switches que realizam apenas o encaminhamento é 30% menor, e os intervalos de confiança inferior (ICI) e superior (ICS) somados à média não se sobrepõem a um nível de confiança em 95%.

Tabela 1. Tempo de processamento de pacotes (ms).

Switch P4	ICI	Média	ICS
Com o mecanismo	0.256	0.279	0.302
Apenas encaminhamento	0.205	0.216	0.227

5. Conclusão

Diante dos desafios encontrados no contexto de redes de pontos de troca de tráfego, a identificação dos fluxos elefantes pode fortemente contribuir na melhora dos serviços prestados aos participantes do PTT. Nessa perspectiva, explorando recursos encontrados em *switches* com suporte a programabilidade, essa trabalho visa realizar a identificação de fluxos elefantes diretamente no plano de dados programável de uma rede de PTT.

O protótipo desenvolvido em P4 se mostrou significativamente mais eficiente em relação ao mecanismo desenvolvido com o protocolo *OpenFlow*, no qual o processo de análise é realizado com o envolvimento do plano de controle. De modo diferente, no mecanismo desenvolvido em P4 todo o processo de análise e identificação é realizado diretamente nos *switches*. Isso permite analisar cada pacote que chega no *switch*, além de realizar a identificação de forma imediata com o pacote que excede os limiares de volume e duração que caracterizam o fluxo como elefante. Os resultados demonstram que é possível identificar e reagir a estes fluxos de forma rápida e eficiente, inserindo um volume de dados de monitoração na rede significativamente menor em relação ao mecanismo que utiliza amostragem de fluxos, implementado com *OpenFlow*. Ainda que a utilização dos recursos de memória nos *switches* seja um desafio, a proposta em P4 permite ao administrador da rede estabelecer o espaço para utilização no mecanismo de acordo com a acurácia desejada. Por fim, no cenário de redes de pontos de troca de tráfego, a identificação e reação aos fluxos elefantes deve ser realizada de forma rápida e precisa. Por isso, acredita-se que esta proposta pode fortemente contribuir no gerenciamento de tráfego para estas infraestruturas.

Como trabalhos futuros, pretende-se realizar um estudo para estabelecer os limiares de forma dinâmica e autônoma. Além disso, deseja-se implementar o mecanismo em *switch hardware* com suporte a programabilidade P4, para avaliá-lo em um contexto real.

Agradecimentos

Os autores agradecem pelo apoio recebido para o desenvolvimento deste trabalho por meio do processo nº 15/24494-8, Fundação de Amparo à Pesquisa do Estado de São Paulo (FAPESP), ao CNPq e à CAPES.

Referências

Araujo, G., Marotta, M., Wickboldt, J., Both, C., Gaspary, L., Rochol, J., and Granville, L. (2017). Caracterizando estratégias de domínio espacial para gerenciamento de regras em redes definidas por software. *35o. Simpósio Brasileiro de Redes de Computadores e Sistemas Distribuídos - SBRC 2017*.

Augustin, B., Krishnamurthy, B., and Willinger, W. (2009). Ixps: mapped? In *Proceedings of the 9th ACM SIGCOMM conference on Internet measurement conference*, pages 336–349. ACM.

Basat, R., Einziger, G., Friedman, R., Luizelli, M., and Waisbard, E. (2017). Constant time updates in hierarchical heavy hitter. *In Proceedings of SIGCOMM '17, LA, CA, USA, August 2017*.

Bosshart, P., Daly, D., Gibb, G., Izzard, M., McKeown, N., Rexford, J., Schlesinger, C., Talayco, D., Vahdat, A., Varghese, G., et al. (2014). P4: Programming protocol-independent packet processors. *ACM SIGCOMM Computer Communication Review*.

Cardona Restrepo, J. C. and Stanojevic, R. (2012). Ixp traffic: a macroscopic view. In *Proceedings of the 7th Latin American Networking Conference*, pages 1–8. ACM.

Curtis, A. R., Mogul, J. C., Tourrilhes, J., Yalagandula, P., Sharma, P., and Banerjee, S. (2011). Devoflow: Scaling flow management for high-performance networks. *ACM SIGCOMM Computer Communication Review*, 41(4):254–265.

Gregori, E., Improta, A., Lenzini, L., and Orsini, C. (2011). The impact of ixps on the as-level topology structure of the internet. *Computer Communications*, 34(1):68–82.

Guo, L. and Matta, I. (2001). The war between mice and elephants. In *Network Protocols, 2001. Ninth International Conference on*, pages 180–188. IEEE.

Gupta, A., Vanbever, L., Shahbaz, M., Donovan, S. P., Schlinker, B., Feamster, N., Rexford, J., Shenker, S., Clark, R., and Katz-Bassett, E. (2015). Sdx: A software defined internet exchange. *ACM SIGCOMM Computer Communication Review*.

Knob, L. A. D., Esteves, R. P., Granville, L. Z., and Tarouco, L. M. R. (2016). Sdefix—identifying elephant flows in sdn-based ixp networks. In *Network Operations and Management Symposium (NOMS), 2016 IEEE/IFIP*, pages 19–26. IEEE.

Knob, L. A. D., Esteves, R. P., Granville, L. Z., and Tarouco, L. M. R. (2017). Mitigating elephant flows in sdn-based ixp networks. In *Computers and Communications (ISCC), 2017 IEEE Symposium on*, pages 1352–1359. IEEE.

Sivaraman, V., Narayana, S., Rottenstreich, O., Muthukrishnan, S., and Rexford, J. (2017). Heavy-hitter detection entirely in the data plane. In *Proceedings of the Symposium on SDN Research*, pages 164–176. ACM.

Suh, J., Kwon, T. T., Dixon, C., Felter, W., and Carter, J. (2014). Opensample: A low-latency, sampling-based measurement platform for commodity sdn. In *Distributed Computing Systems (ICDCS), 2014 IEEE 34th International Conference on*.

Tong, D. and Prasanna, V. (2015). High throughput hierarchical heavy hitter detection in data streams. In *High Performance Computing (HiPC), 2015 IEEE 22nd International Conference on*, pages 224–233. IEEE.

Wickboldt, J. A., De Jesus, W. P., Isolani, P. H., Both, C. B., Rochol, J., and Granville, L. Z. (2015). Software-defined networking: management requirements and challenges. *IEEE Communications Magazine*, 53(1):278–285.

ANEXO B ARTIGO PUBLICADO – GLOBECOM 2018

Este artigo apresenta IDEAFIX, um mecanismo para identificar fluxos elefantes em redes de pontos de troca de tráfego com suporte a programabilidade P4. A abordagem consiste em analisar as características dos fluxos para cada pacote que ingressa na rede, imediatamente nos *switches* de borda. Essas informações são então armazenados em registradores P4, indexados por chaves hash e confrontadas com limiares predefinidos para classificação dos fluxos. Além disso, nesse artigo é descrito um mecanismo para atualização dos registradores para contornar o problema de falsos negativos, bem como o *pipeline* de processamento de pacotes pelo dispositivo de rede que implementa o mecanismo proposto. A avaliação experimental mostra que IDEAFIX é significativamente mais eficiente que as abordagens do estado-da-arte implementadas com redes SDN (*Software Defined Networking*) tradicionais que utilizam o protocolo *OpenFlow* para controle de fluxos e *sFLow* para amostragem de tráfego.

- **Título:**
  IDEAFIX: Identifying Elephant Flows in P4-Based IXP Networks
- **Conferência:**
  IEEE Global Communications Conference (GLOBECOM)
- **Tipo:**
  Trilha principal (*full-paper*)
- **Qualis:**
  A1
- **URL:**
  <https://xpressdrivein.org/glo18/pdf/1570450165.PDF>
- **Data:**
  9 a 13 de Dezembro de 2018
- **Realizado em:**
  Abu Dhabi, United Arab Emirates

IDEAFIX: Identifying Elephant Flows in P4-Based IXP Networks

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

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

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

I. Introduction

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

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

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

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

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

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

978-1-5386-4727-1/18/$31.00 ©2018 IEEE

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

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

II. Related Work

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

A. SDN in IXP

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

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

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

B. Flows Analysis in Programmable Data Planes

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

III. IDEAFIX

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

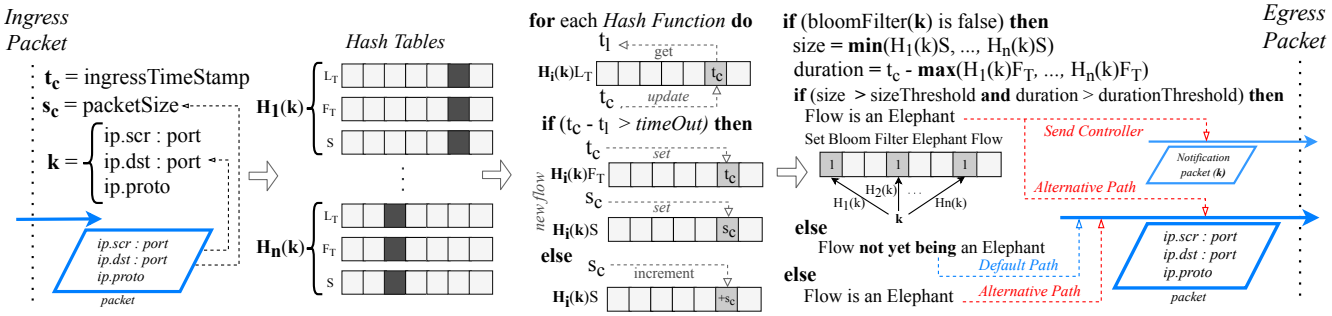
Figure 1 presents the four main steps of IDEAFIX detection process: *extraction*, *consolidation*, *classification*, and *notification*. First, when a packet arrives at the switch, we map the extracted 5-tuple of the packet through a hash function. Next, the consolidation step identifies to which network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

A. Extraction and consolidation

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

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

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



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

Fig. 1. Packet process.

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

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

B. Classification and notification

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

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

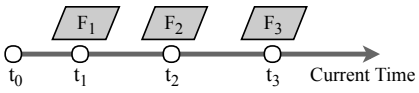


Fig. 2. Flows start time.

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

IV. Evaluation

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

A. Comparison Approach

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

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

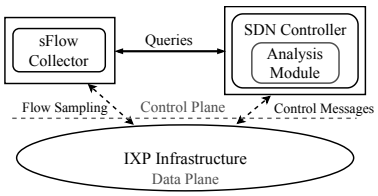


Fig. 3. State-of-the-art mechanism.

B. Scenario

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

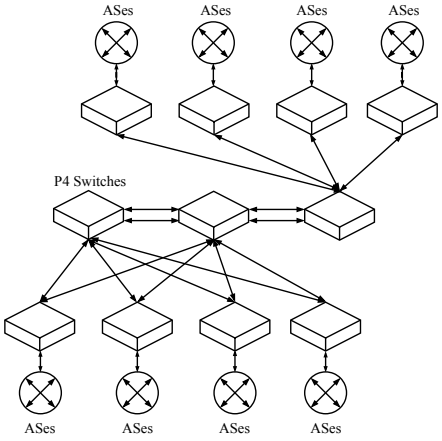


Fig. 4. IXP network topology [1].

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

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

C. Metrics

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

D. Results

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

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

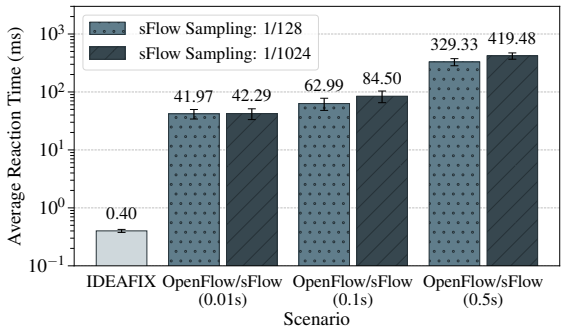


Fig. 5. Threshold violation/surpassing reaction time.

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

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

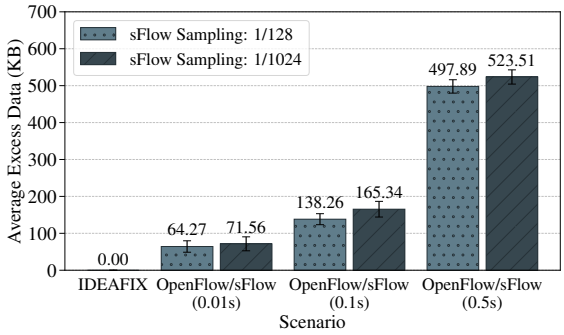


Fig. 6. Excess data.

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

TABLE I
USE OF RESOURCES BY THE IDENTIFICATION MODULE.

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

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

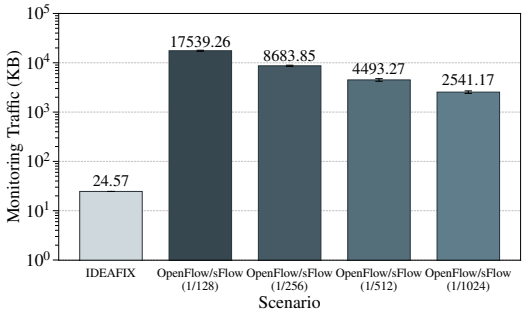


Fig. 7. Monitoring data.

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

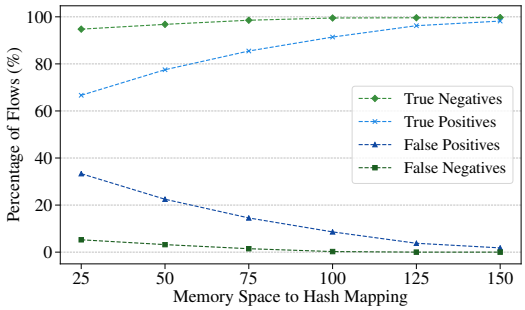


Fig. 8. IDEAFIX accuracy.

The IDEAFIX accuracy (Figure 8) was evaluated by varying the memory space to hash mapping on the switches in relation to the number of flows inserted in the network (*i.e.*, 2048). With 25% of available space, there is a percentage of about 5% of false negatives and 35% of false positives. This is a result of the strategy that overestimates the flows, wherein false positives are preferred over false negatives when hash

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

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

TABLE II
PACKET PROCESSING TIME (ms).

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

E. Summary

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

V. CONCLUSION

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

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

ACKNOWLEDGEMENT

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

REFERENCES

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

ANEXO C ARTIGO ACEITO PARA PUBLICAÇÃO – AINA 2019

Esse artigo apresenta um mecanismo para predizer o comportamento de fluxos usando observações históricas e, observando padrões temporais, identificar os fluxos elefantes mesmo antes que estes venham a exceder os limiares de classificação. A abordagem consiste em prever o tamanho e a duração de novos fluxos a partir de um modelo de regressão localmente ponderada (LWR - *Locally Weighted Regression*), utilizando como amostras o comportamento de fluxos anteriores e sua correlação temporal com o novo fluxo. Os resultados experimentais mostram que o mecanismo é capaz de prever o volume e a duração de novos fluxos e reagir rapidamente aos fluxos elefantes, aproximadamente 50.29ms com até 32 amostras no modelo de previsão. Esses números são muito menores que o tempo necessário para o fluxo exceder os limiares que o classificaria como um elefante.

- **Título:**
  Predicting Elephant Flows in Internet Exchange Point Programmable Networks
- **Conferência:**
  XXXIII International Conference on Advanced Information Networking and Applications (AINA)
- **Tipo:**
  Trilha principal (*full-paper*)
- **Qualis:**
  A2
- **Data:**
  27 a 29 de Março de 2019
- **Realizado em:**
  Kunibiki Messe, Matsue, Japan

Predicting Elephant Flows in Internet Exchange Point Programmable Networks

Marcus Vinicius Brito da Silva, Arthur Selle Jacobs,
Ricardo José Pfitscher, and Lisandro Zambenedetti Granville

Institute of Informatics – Federal University of Rio Grande do Sul
Av. Bento Gonçalves, 9500 – Porto Alegre, Brazil
{mvbsilva, asjacobs, rjpfitscher, granville}@inf.ufrgs.br

Abstract. Internet Exchange Points (IXPs) are high-performance networks that allow multiple autonomous systems to exchange traffic, with benefits ranging from cost reductions to performance improvements. As in any network, IXP operators face daily management challenges to promote better usage of the services provided by the network. An essential problem in IXP management concerns the identification of elephant flows, which are characterized by having traffic size and duration significantly higher than other flows. The current approaches to the identification of elephant flow in IXP networks depend that the analyzed flows exceed predefined thresholds to classify them as elephants. However, although it is not perceptible initially, elephant flows are elephant ones since their first packet. Hence, in this paper, we present a mechanism to predict flows behavior using historical observations and, by recognizing temporal patterns, identify elephant flows even before they exceed such thresholds. Our approach consists in predicting new flows size and duration through a Locally Weighted Regression (LWR) model, using the previous flows behavior and its temporal correlation with the new flow. The experimental results show that our mechanism is able to predict the volume and duration of new flows, and react to elephant flows rapidly, approximately 50.3 ms with up to 32 historical samples in the prediction model. These numbers are much smaller than the time each flow would take to exceed the thresholds to classify it as an elephant. In addition, the mechanism accurately predicts up to 80% of elephant flows in our evaluation scenarios and approximately 5% of false positives.

Keywords: Software-Defined Networking, Network Management, P4 Language

1 Introduction

Internet Exchange Points (IXPs) are high-performance networks and perform an essential role in the Internet ecosystem [1], accounting for at least 20% of all traffic exchanged between autonomous systems (ASes) [2]. As in any network, IXP operators face daily management challenges to promote a better usage of the services provided by the network. An important problem in IXP management concerns the identification of elephant flows, who are characterized by having traffic size and duration significantly higher than other flows [3]. As a consequence, elephant flows can significantly impact

on the traffic of the smaller flows that share the same path on the IXP infrastructure, thus compromising the overall perceived network Quality of Service (QoS) [4].

Because of the impacts that elephant flows have on network performance, network operators must quickly detect them to promptly perform mitigation actions [5]. As elephant flow classification depends on the flows' size and duration, it takes an amount of time t for these values (size and duration) to reach a pre-established threshold; that delays the reaction in at least t units of time. However, elephant flows are obviously elephant ones since their first packet, although detection occurs later (t). Today, there is a lack of solutions that take this situation into account, thus delaying decisions/actions, and consequently creating problems for smaller flows.

Proposed solutions such as DevoFlow [6], OpenSample [7], and SDEFIX [8] present mechanisms for identifying elephant flows in IXP networks using sFlow [9] and managing paths (as reaction to elephant flows) using SDN/OpenFlow [10]. As a first attempt to more quickly identify elephant flows in programmable IXP networks, we had proposed IDEAFIX [11], where flow duration and size were analyzed for each ingress packet immediately at the edge P4 [12] switch. Although IDEAFIX reduces the detection delay when compared to controller-based approaches, it still requires flows size and duration to reach the thresholds to be identified as elephant flow.

In this paper, to shorten the time interval until classification thresholds are surpassed, we present a mechanism to identify elephant flows in IXP programmable networks using historical observations. Our approach consists of predicting flow size and duration through a Locally Weighted Regression (LWR) model [13] [14]. The sample weights, in the LWR model, are attributed from a Gaussian distribution adjusted by the network operator according to the desired sample time window. In addition, the network operator can define a tolerance range to validate estimations according to the calculated prediction interval for each of them. If the prediction interval is less than the tolerance, then the inferred values are considered valid and immediately confronted with thresholds to characterize the flow as an elephant one or not, right in the beginning. When the prediction interval is larger than the tolerance, the inference is invalidated and the analysis proceeds to the second approach, based on the IDEAFIX identification process.

Experimental results show that our mechanism is able to predict the volume and duration of new flows and identify elephant flows rapidly, approximately 50.3ms with up to 32 samples (*i.e.*, previous flows behavior). Even when sample numbers are significantly large, *i.e.*, 2048 or 4096, our mechanism can predict and react to elephant flows at an interval of up to 126.3ms and 174.6ms, respectively. These numbers are much smaller than the time at which the flow would take to exceed the thresholds to classify it as an elephant one. In addition, the packet processing time of software-emulated P4 switches also influences the reaction time. On a hardware switch, packet processing time would be in nanoseconds. Finally, the mechanism's accuracy in well-behaved scenarios obtained at least 80% of prediction, even with conservative tolerance, and approximately 5% of false positives.

The remainder of this paper is organized as follows. In Section 2, related work is discussed. In Section 3, we describe the concept of elephant flows and the management challenges they require in an IXP network, as well as the statistical method used in our proposal. In Section 4, we describe our novel mechanism to predict and identify

elephant flows in IXP networks. In Section 5, we present the evaluation of our proposal, as well as the achieved results. Finally, in Section 6 we present the main conclusions of the study and future work perspectives.

2 Related Work

In DevoFlow [6], the OpenFlow protocol is used to keep track of elephant flows with different monitoring mechanisms, such as packet sampling and port triggering. DevoFlow changes the OpenFlow switches by adding new features and counters to the hardware in order to facilitate the identification of flows in the network. In OpenSample [7], sFlow is used to perform flow sampling on a solution to manage SDN networks. Then, flow rates are calculated by subtracting TCP sequence numbers from two samples of the same flow and dividing the value by the elapsed time between them. However, the statistics of the switches are not considered and the flows need to be sampled and sent to the control plane to be processed. This delays elephants flow identification and add a significant amount of sampling data in the network. Therefore, only when a threshold is exceeded, in terms of byte count, the flow is classified as an elephant one and it is rerouted to the least congested path between the endpoints.

In SDEFIX [8], an identification module is used to classify flows, analyzing the collected data by sFlow according to predefined rules. When size and duration thresholds are exceeded, the flow is classified as elephant and it is mitigated according to policies written by the network operator. For example, elephant flows can be routed through alternative paths in the IXP infrastructure, so as not to affect the small flows that are sharing the same paths in the network. As in the previous approaches, SDEFIX performs elephant flow analysis and identification integrally in the control plane and a reaction occurs only when thresholds are exceeded.

In our previous work [11], we proposed IDEAFIX, aiming to identify elephant flows in IXPs faster by relying on programmable data planes. To do that, IDEAFIX takes advantage of P4 features to store information about the size and the duration of network flows. This information is extracted from each packet that enters the network and compared to predefined thresholds for flow classification. If flow size and duration do not exceed such thresholds, then it is marked as not yet elephant, and the packet is routed to the default path. If thresholds are exceeded, however, then the flow is classified as an elephant one and a notification is sent to the control plane to report the detected elephant flow. Then, the network operator can define actions to mitigate the effects of that flow, in accordance with, for example, QoS policies.

Although the previous approaches allow the identification of elephant flows in IXP networks, there is a dependency on that the analyzed flows exceed the thresholds to be classified as an elephant flow. However, although it is not perceptible initially, elephant flows are elephant ones since their first packet. In this paper, to dispense with waiting dependence for the threshold to be exceeded, we present a mechanism to predict flows behavior using historical observations and, from the temporal patterns recognition, identify elephant flows even before they exceed thresholds.

3 Elephant Flows in IXP Networks and LWR

In the Internet, most of the flows have a small size and/or lifetime (*i.e.*, mice flows or small flows) [15] [16], althought there is a smaller number of flows that accounts for the majority of the traffic, also having a longer lifetime; these are the elephant flows [3] [17]. The presence of elephant flows in networks is common, and because they are flows with a long lifetime and size they may cause performance issues that demand proper management actions from network operators. For example, elephant flows can impact on smaller flows that occasionally share the same network data path. Also, elephant flows can consume memory resources of network devices, and this also can lead to undesirable delays, queuing, and packet losses [18] [3] [8]. In the case of IXP networks, the problem turns to be even more critical, because of the amount of traffic that IXP networks must deal with. Thus, the faster the elephant flows are identified, the faster their effects can be mitigated [8].

Figure 1 presents the traffic behavior in Australia IXPs [19] over the course of a week. IX-Australia offers peering in Western Australia (WA-IX), New South Wales (NSW-IX), and Victoria (VIC-IX). Although this is an aggregated traffic behavior, it is possible to recognize a periodic pattern, with discrete variations. This IXP's behavior is also seen in the Amsterdam Internet Exchange (AMS-IX), as well as on more than 30 IXPs networks in Brazil [20]. Other IXPs experience the same traffic behavior as well. Even though it is not possible to explicitly spot the elephant flows inside Figure 1, they are there. In some cases, it is observed that elephant flows can contribute with more than 59% of total traffic volume [16]. In addition, elephant flows traffic has a substantial, but not perfect, correlation with traffic in total flows [21].

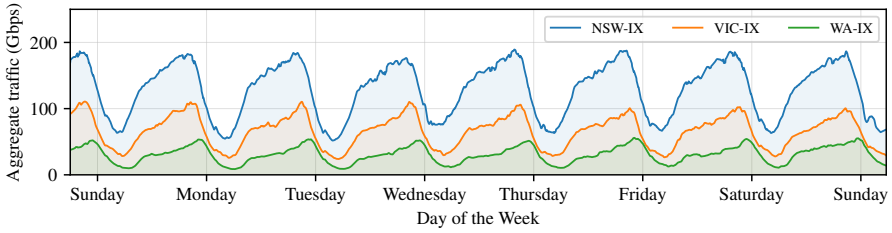


Fig. 1. Australia IXPs aggregate traffic.

Considering that there is a periodicity in the traffic of an IXP network (as shown in Figure 1), it is possible to predict the size and duration of new flow instances by observing the previous flows' temporal behavior. In other words, as the traffic pattern exhibits a periodicity, the events that previously resulted in elephant flows will later probably happen again. When taking the history of flows to predict volume and duration of new ones, we also consider that, in such a history, younger instances of flows must weight more than older instances in the prediction computation. By doing so, we can monitor flows behavior in the IXP network at run-time and predict the behavior of new flows according to their periodicity.

3.1 Locally Weighted Regression prediction method

To predict network flows behavior, we leverage a statistical method, *i.e.*, locally weighted regression (LWR) [13] [22]. LWR allows to predict the value of dependent variables from a set of independent variables, through localized weighting. LWR is based on the assumption that the neighboring values of the desired point, in a sample range, are the best indicators for the prediction [13]. In our context, the size and duration of previous flow instances (samples in the model) are dependent variables, its start timeStamp are the independent variables, and the predictions are the new flow size and duration in its start timeStamp (desired point). The neighbors are the flow instances with timeStamp closest of the new flow's timeStamp.

This method fits a surface to "local" points using distance-weighted regression [14]. LWR is derived from standard linear regression [23], as shown by Equation 1. LRW proposal consists of defining the linear model β parameters, minimizing the squared errors for each sample $(y_i - f(x_i))$, weighted locally by w_i weight.

$$F(\beta_0, \beta_1, ..., \beta_m) = \sum_{i=1}^{n} w_i (y_i - f(x_i))^2 \tag{1}$$

After the derivation [13] [22] of Equation 1, the linear model β parameters, used to predict values, are obtained by normal equations:

$$\beta = \left(X^T W^T W X\right)^{-1} X^T W^T W y \tag{2}$$

where X is an $n \times (m+1)$ matrix consisting of n data point, each represented by its m input dimensions and a "1" in the last column, y is a vector of corresponding outputs for each data point, W is a diagonal matrix of order n with the w_i weights of each data point, and β is the $m+1$ vector of unknown regression parameters. Thus, prediction of the outcome of a query point x_q becomes:

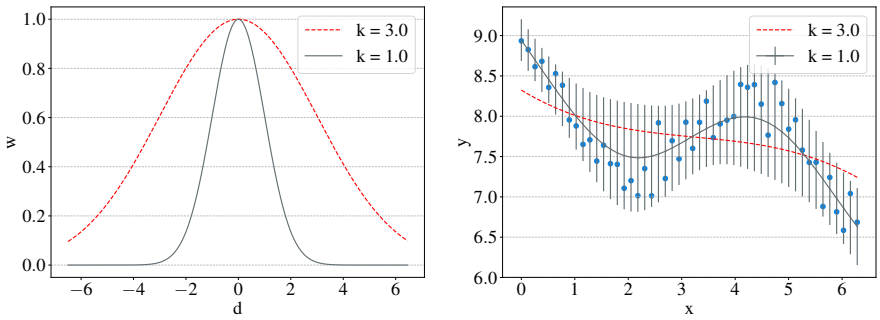
$$y_q = x_q^T \beta \tag{3}$$

The weights allow establishing the influence of the samples according to their distance (*i.e.*, age) to the desired point. Many weighting functions are proposed in literature [13] [24]. The widely used weighting function is the Gaussian kernel weighting function [25], which can be defined as follows:

$$w_i = exp\left(\frac{-d_i^2}{2k^2}\right) \tag{4}$$

From Equation 4, each sample (x_i, y_i) in the model will receive a weight w_i in the interval $[0, 1]$ (as shown in Figure 2(a)), according to its distance $d_i = \sqrt{(x_q - x_i)^2}$ to the desired point x_q. That is, how much less the distance of the (x_i, y_i) point to the (x_q, y_q) desired point (*i.e.*, $d_i \approx 0$), the greater the influence of (x_i, y_i) on the model because weight will be closer to 100% (*i.e.*, $w_i \approx 1$). The parameter k scales the size of the kernel to determine how local the regression will be. Figure 2(a) shows the Gaussian kernel magnitude with k = 1.0 and k = 3.0. That is, the greater the value of k, the larger the significant samples range in the model. However, if k is small, then the model will be influenced only by very close samples.

Figure 2(b) shows the model behavior influenced by the adjustment of the magnitude k of the Gaussian kernel. The samples were obtained by: $f(x) = -(\sin(x) + (0.4x) + 0.26\eta) + 9$; where η is a samples random noise, and $0 \leqslant x \leqslant 2\pi$. Due to small sample variation $(6.5 \leqslant y \leqslant 9.0)$ in the model, when $k = 1.0$, LWR result is well behaved and smoothly. When $k = 3.0$, the result is similar to traditional linear regression, which does not allow to follow the sample changes. The fine adjustment of parameter k is essential to use LWR properly. We propose k to be defined by the sample standard deviation of the x_i independent variables. This allows to dynamically adapt the Gaussian kernel according to changes in the behavior of the network.



(a) Gaussian kernel magnitude.

(b) LWR influenced by k parameter.

Fig. 2. Gaussian kernel and LWR prediction interval.

We use prediction intervals [22] to assess the quality of fits in predictions. Prediction intervals I_q are expected bounds of the prediction error at a query point x_q, which can be defined as follows:

$$I_q = x_q^T \beta \pm t_{\alpha/2,n'-p'} s \sqrt{1 + x_q^T (X^T W^T W X)^{-1} x_q} \tag{5}$$

where $t_{\alpha/2,n'-p'}$ is Student's t-value of $n' - p'$ (Equation 6) degrees of freedom for a $100(1-\alpha)\%$ prediction bound.

$$n' = \sum_{i=1}^{n} w_i^2 \quad \text{and} \quad p' = \frac{n'}{n} m \tag{6}$$

Figure 2(b) shows the prediction interval behavior for (x_q, y_q) inferences with $k = 1.0$. The prediction interval increases when there is a large sample variation $(e.g., x \approx 2)$. When the model is well behaved $(e.g., x \approx 0)$, the forecast interval is smaller. This allows obtaining the accuracy of the inference method even with large sample variations.

4 Approach

In this section, we describe the proposed mechanism to predict and identify elephant flows using historical observations and temporal correlation. Figure 3 shows the architecture of our proposal, composed of an IXP network infrastructure abstraction with a

programmable data plane, a historical database, and a prediction module, in the control plane, attached to the network controller. When an edge switch receives the first packet of a flow, the control plane receives a notification to compute the path by which the flow will be routed in the IXP network. Then, running at the control plane, the mechanism uses the LWR model to predict the flow size and duration. When predicted values are considered valid, from the prediction interval tolerated, the mechanism characterizes the flow as an elephant or not, according to thresholds. Such a process of flows' parameters prediction and related characterization is discussed following, in four phases: sample selection, prediction, validation of predictions and flow classification.

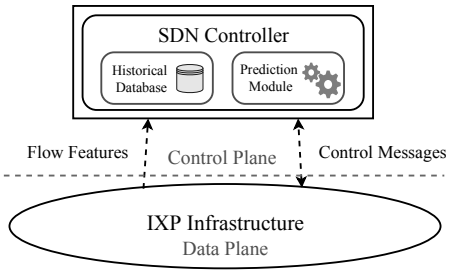


Fig. 3. Architecture.

The prediction mechanism relies on a historical database to store information about the behavior of previous flow instances. This information is extracted directly from the data plane, where an agent uses P4 to account, in each edge switch, the volume and duration of each flow that ingresses in the IXP network. To identify each flow individually, a 5-tuple (*i.e.*, source and destination IP addresses, source and destination ports, and transport protocol type) is mapped by hash functions to generate index keys. When a flow is finalized (*e.g.*, TCP FIN Flag is valid or timeout exceeded), a notification (on top of UDP) is sent to the control plane to report the flow size and duration along with its 5-tuple and start flow time (*i.e.*, *timeStamp*).

The sample selection is based on the tuple that defines the new flow. Initially, all records with the same source and destination IP address, transport protocol, and at least one of the application ports are selected. Samples are selected within a time window defined by the network operator to delimit the size of the historical database. For example, it is possible to set a sampling interval in days, weeks, months, and so on. In addition, selection can occur from hierarchical aggregation of IP addresses, based on HHH [26]. This enables more samples in the inference process, *e.g.*, to sub-nets behavior analysis.

The prediction step estimates the flow parameters according to historical information. To do that, we use the previously described LWR model (see Section 3.1). We use flow sizes and duration as dependent variables y_q in Equation 3, allowing the mechanism to predict these values individually. In both cases, the independent variables x_i are the previous flow *timeStamp* (samples), and x_q is current *timeStamp* (new flow). To establish temporal correlation and periodicity, we defined the sample temporal distance d_i concerning to the new flow time of the day (*i.e.*, within 24 hours), in according to Equation 7. That is, the largest temporal distance of a sample to the current flow will

be $24/2$ hours. Thus, it is possible to correlate periodic occurrences of elephant flows daily. For exemplification purposes, the *timeStamp* is used in hours.

$$d_i = \left|x_q - x_i\right| mod\ 24 = \begin{cases} d_i & \textbf{if } d_i \leq 24/2 \\ 24 - d_i & \textbf{if } d_i > 24/2 \end{cases} \tag{7}$$

The samples weighting in the LWR model is defined according to Equation 4. We make a modification in the Gaussian function, as shown in Equation 8. From this, the network operator can establish a minimum weight for a given range of samples. That is, distances in the range $[0,k]$ may receive a minimum weight equal to l (*e.g.*, 30% or 40%), when d_i (Equation 7) is equal to k. This allows operators to determine how conservative the inference process will be. For example, in scenarios where the network is saturated, the network operator can be more rigorous in the samples weighing to define which temporal distance will have the most significant influence on the predictions.

$$w_i = exp\left(\frac{d_i^2}{k^2} \cdot \ln(l)\right) \tag{8}$$

To adjust the magnitude of the Gaussian kernel, we propose k to be defined by the sample standard deviation of the x_i independent variables. This allows to dynamically adapt the Gaussian kernel according to changes in the behavior of the network. Finally, when sample weights are defined, it is possible to determine the β parameters of the linear model (Equation 2) and then predict the new flow volume and duration (Equation 3) according to its start *timeStamp* (*i.e.*, x_q).

After predicting flows parameters, the mechanism performs the validation of LWR inferred values. For this validation, the network operator must define a tolerance for the calculated prediction interval over each inference. The prediction interval determines the fluctuation margin for the inferred value, as shown in Figure 2(b). Thus, the network operator can determine the tolerance according to, for example, the demand of the network. That is, when the network is underutilized, the operator can set a softer tolerance; otherwise, in saturation cases, the tolerance can be more conservative.

Lastly, when the inferences are not validated (*i.e.*, $pred\_interv > tol$), it will not be possible to predict the flow behavior, and consequently, it will not be possible to classify it as an elephant one. In this case, analysis and possible identification occurs based on our previous IDEAFIX identification reference. In IDEAFIX, the packets of each flow are analyzed directly in the programmable data plane, and elephant flow identification happens whenever the volume and duration of a flow exceed the predefined thresholds.

5 Evaluation

To assess the feasibility of our proposal, we focus on evaluating three main aspects: (*i*) reaction time, *i.e.*, the interval between the ingress time of the flow first packet in the IXP network, the LWR predicting time, and the identified elephant flow management time; (*ii*) excess data, *i.e.*, the number of bytes that transited in default path until the flow has been identified as an elephant one and a reaction occurs; and (*iii*) mechanism accuracy, *i.e.*, the percentage of valid elephant flows predictions in function of the prediction interval tolerance. For all metrics, lower values are better.

Figure 4 depicts the topology used in the evaluation experiments. We used a topology based on the AMS-IX [27] infrastructure, as also used in the related work [11] [8], with 8 ASes connected by edge switches to a programmable data plane IXP. Each edge switch is directly linked to a programmable switch, which, in turn, has at least two connection paths to the core of the IXP network. Switches were implemented in the language $P4_{16}$ and by using the software switch BMv2. The infrastructure was emulated using *Mininet* 2.3, with a bandwidth of 1Gbps per link and no propagation delay.

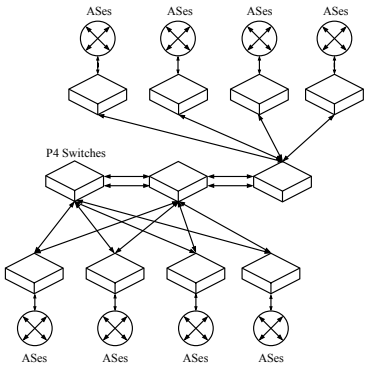
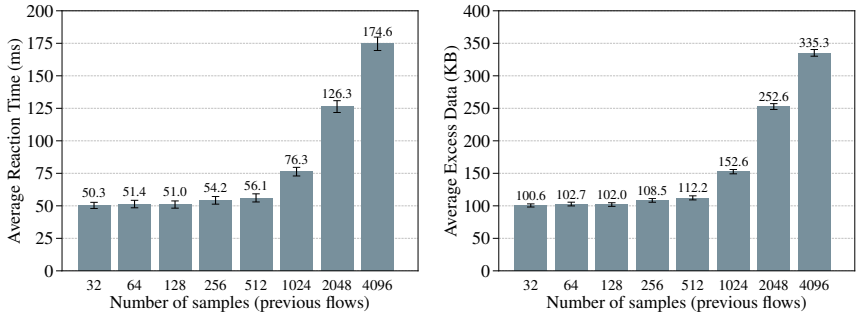


Fig. 4. IXP network topology [8].

We generated a workload with distinct sizes of TCP flows between the connected ASes, using *iPerf*. The flow bandwidth was established at 10 Mbps, and the duration was determined through a normal distribution, with a mean of 150 seconds, and a standard deviation of 20 seconds, for elephant flows. For small flows, we used a mean of 10 seconds and a standard deviation of 4 seconds [8] [11]. The IXP network traffic was distributed periodically over nine weeks (*i.e.*, $\approx$ 1,600 hours). The thresholds were defined in 10 MB and 10 seconds [8] [4]. Each experiment lasted 10 minutes, repeated 32 times, and 2,048 flows were generated, of which 12% were elephant flows [11]. We used a computer with Intel Core i7-4790 processor, 16GB of RAM, and *Ubuntu 16 LTS*.

Figure 5(a) shows the results of the reaction time to elephant flows. The reaction time was analyzed exclusively as a function of the number of samples in the prediction model. This allows us to observe the time it takes for a reaction to occur with the prediction model being generated at run-time. In the y-axis, the average reaction time (in milliseconds) has no significant difference at a 95% confidence level for a set of up to 512 samples. The most significant difference is evident when there are sets with at least 1,024, 2,048, and 4,096 samples. In these cases, the average reaction time varies between 76.3ms, 126.3ms, and 174.6ms, respectively.

These numbers are much smaller than the time at which the flow would take to exceed the thresholds to classify it as an elephant. That is, for the thresholds we used, based on the related work [8] [11], an elephant flow would take at least 10s to be identified, at best case. Thus, it is possible to notice that predicting the flows behavior early on allows one to identify the elephant flows as soon as possible. Moreover, the packet processing time of software-emulated P4 switch also influences the reaction time in our approach. On a hardware switch, packet processing time would be in nanoseconds.



(a) Reaction time. (b) Excess data.

Fig. 5. Reaction time and excess data analysis.

Figure 5(b) shows the elephant flows data volume that traveled along the standard path until a reaction occurred. In the y-axis, the average excess data (in Kilobytes) has no significant difference at a 95% confidence level for up to 512 samples. As in the previous analysis, the most significant difference one is evident when there are sets with at least 1,024, 2,048, and 4,096 previous flow samples in the prediction model. In these cases, we can consider that the elephant flows still transmit about 152.6KB, 252.6KB, and 335.3KB on the standard path in the network, respectively.

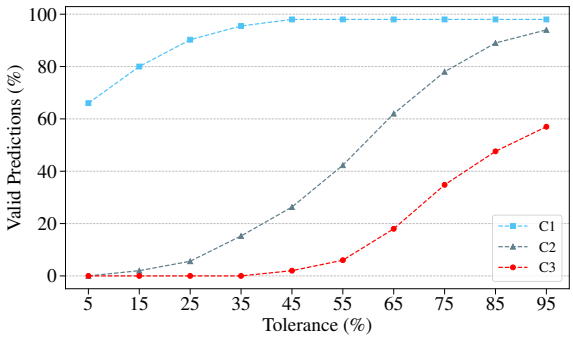


Fig. 6. Predict mechanism accuracy.

The mechanism accuracy (in Figure 6) has evaluated from the percentage of valid predictions, *i.e.*, correctly predicted and identified elephant flows. The first scenario (C1 curve) with low traffic behavior variation and regular periodicity, *i.e.*, flows follow a well-defined frequency. Scenario two (C2 curve) with more significant variation in the flows behavior and periodicity. In scenario three (C3 curve), there was a high variation in the flows behavior and low periodicity, *i.e.*, flows do not have a well-defined frequency pattern. Thus, we evaluate the percentage of valid predictions according to the

tolerance defined by the network operator for the prediction interval. Prediction interval was calculated for each predicted value at a confidence level of 90%.

Figure 6 shows that in a well-behaved scenario (C1 curve), 80% of the predictions are validated even with a conservative tolerance (*e.g.*, 15%). In the worst case (C3 curve), when the traffic behavior and periodicity are not regular, approximately 20% of the predictions could be validated, with a tolerance of at least 65%. In the mean case (C2 curve), more than 60% of the predictions were validated at a tolerance of 65%. These results show that the method can predict and validate values even in non-regular scenarios. However, it requires more flexibility in the predictions tolerance. These results represent, at least, 90% of success in elephant flows identification, out of the total number of elephant flows inserted in the network at each test (*i.e.*, $\approx$ 245 flows), with approximately 5% of false positives. Lastly, the elephant flows whose predictions were not valid, was identified directly in the data plane, based on our previous IDEAFIX [11] approach, immediately after exceeding the thresholds.

6 Conclusions

In this paper, we presented a mechanism to predict and react to elephant flows in programmable IXP networks before thresholds are exceeded. Our approach consists of predicting flow size and duration through a Locally Weighted Regression (LWR) model, following the temporal correlation between the behavior of previous flows. Our experimental results show that our mechanism is able to predict the volume and duration of new flows, and react to elephant flows rapidly, *i.e.*, approximately 50.3ms with up to 32 samples in the model. These numbers are much smaller than the time at which the flows would take to exceed the thresholds to classify them as elephant ones. In addition, the mechanism's accuracy was 80% of validated predictions even with a conservative tolerance of 15%, and approximately 5% of false positives. As future work, we will consider other methods, based on machine learning, to predict flows behavior and mechanisms to define the thresholds dynamically. We also plan to deploy our solution in actual IXP networks if the operations conditions allow.

References

1. B. Augustin, B. Krishnamurthy, and W. Willinger, "IXPs: Mapped?" in *ACM SIGCOMM Conference on Internet Measurement*, ser. IMC '09. NY, USA: ACM, 2009, pp. 336–349.
2. J. C. Cardona Restrepo and R. Stanojevic, "IXP traffic: a macroscopic view," in *Proceedings of the 7th Latin American Networking Conference*. ACM, 2012, pp. 1–8.
3. L. Guo and I. Matta, "The war between mice and elephants," in *Network Protocols, 2001. Ninth International Conference on*. IEEE, 2001, pp. 180–188.
4. L. A. D. Knob, R. P. Esteves, L. Z. Granville, and L. M. R. Tarouco, "Mitigating elephant flows in SDN-based IXP networks," in *Computers and Communications (ISCC), 2017 IEEE Symposium on*. IEEE, 2017, pp. 1352–1359.
5. E. Gregori, A. Improta, L. Lenzini, and C. Orsini, "The impact of IXPs on the AS-level topology structure of the Internet," in *Computer Communications*. Elsevier, 2011, pp. 68–82.
6. A. R. Curtis, J. C. Mogul, J. Tourrilhes, P. Yalagandula, P. Sharma, and S. Banerjee, "DevoFlow: Scaling flow management for high-performance networks," in *ACM SIGCOMM Conference on Internet Measurement*, vol. 41. NY, USA: ACM, 2011, pp. 254–265.

7. J. Suh, T. T. Kwon, C. Dixon, W. Felter, and J. Carter, "Opensample: A low-latency, sampling-based measurement platform for commodity sdn," in *34th IEEE International Conference on Distributed Computing Systems (ICDCS)*. IEEE, 2014, pp. 228–237.
8. L. A. D. Knob, R. P. Esteves, L. Z. Granville, and L. M. R. Tarouco, "SDEFIX—Identifying elephant flows in SDN-based IXP networks," in *IEEE/IFIP Network Operations and Management Symposium (NOMS)*. IEEE, 2016, pp. 19–26.
9. sFlow, "sFlow.org," http://www.sflow.org, 2018.
10. N. McKeown, T. Anderson, H. Balakrishnan, G. Parulkar, L. Peterson, J. Rexford, S. Shenker, and J. Turner, "OpenFlow: enabling innovation in campus networks," in *ACM SIGCOMM Conference on Internet Measurement*. NY, USA: ACM, 2008, pp. 69–74.
11. M. V. B. da Silva, A. S. Jacobs, R. J. Pfitscher, and L. Z. Granville, "IDEAFIX: Identifying Elephant Flows in P4-Based IXP Networks," in *Proceedings of the IEEE Global Telecommunications Conference (GLOBECOM)*. IEEE, 2018.
12. P. Bosshart, D. Daly, G. Gibb, M. Izzard, N. McKeown, J. Rexford, C. Schlesinger, D. Talayco, A. Vahdat, G. Varghese *et al.*, "P4: Programming protocol-independent packet processors," in *ACM SIGCOMM Conference on Internet Measurement*. ACM, 2014, pp. 87–95.
13. W. S. Cleveland and S. J. Devlin, "Locally weighted regression: an approach to regression analysis by local fitting," in *Journal of the American statistical association*, vol. 83, no. 403. Taylor & Francis, 1988, pp. 596–610.
14. E. E. Elattar, J. Goulermas, and Q. H. Wu, "Electric load forecasting based on locally weighted support vector regression," in *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, vol. 40, no. 4. IEEE, 2010, pp. 438–447.
15. Y. Zhang, L. Breslau, V. Paxson, and S. Shenker, "On the Characteristics and Origins of Internet Flow Rates," in *ACM SIGCOMM Conference on Internet Measurement*, vol. 32, no. 4. New York, NY, USA: ACM, Aug. 2002, pp. 309–322.
16. T. Mori, R. Kawahara, S. Naito, and S. Goto, "On the characteristics of Internet Traffic variability: Spikes and Elephants," in *IEICE TRANSACTIONS on Information and Systems*, vol. 87, no. 12, 2004, pp. 2644–2653.
17. W. Fang and L. Peterson, "Inter-AS traffic patterns and their implications," in *IEEE Global Telecommunications Conference (GLOBECOM)*, vol. 3. IEEE, 1999, pp. 1859–1868.
18. T. Mori, M. Uchida, R. Kawahara, J. Pan, and S. Goto, "Identifying elephant flows through periodically sampled packets," in *ACM SIGCOMM Conference on Internet Measurement*, ser. IMC '04. ACM, 2004, pp. 115–120.
19. IX Australia, "Australia Internet Exchange Point," https://www.ix.asn.au/, 2018.
20. Internet Steering Committee in Brazil, "Brazil Internet Exchange Points," http://ix.br/trafego/agregado/rs, 2018.
21. Y. Li, H. Liu, W. Yang, D. Hu, X. Wang, and W. Xu, "Predicting inter-data-center network traffic using elephant flow and sublink information," in *IEEE Transactions on Network and Service Management*, vol. 13, no. 4. IEEE, 2016, pp. 782–792.
22. S. Schaal and C. G. Atkeson, "Robot juggling: implementation of memory-based learning," *IEEE Control Systems*, vol. 14, no. 1, pp. 57–71, 1994.
23. R. Jain, *The art of computer systems performance analysis: techniques for experimental design, measurement, simulation, and modeling*. John Wiley & Sons, 1990.
24. J. S. Simonoff, *Smoothing methods in statistics*. Springer Science & Business Media, 2012.
25. M. P. Wand and W. R. Schucany, "Gaussian-based kernels," in *Canadian Journal of Statistics*, vol. 18, no. 3. Wiley Online Library, 1990, pp. 197–204.
26. R. Basat, G. Einziger, R. Friedman, M. Luizelli, and E. Waisbard, "Constant Time Updates in Hierarchical Heavy Hitter," in *ACM SIGCOMM Conference on Internet Measurement*, ser. SIGCOMM '17. NY, USA: ACM, 2017, pp. 127–140.
27. AMS-IX, "Amsterdam Internet Exchange Infrastructure," https://ams-ix.net/technical/ams-ix-infrastructure, 2018.