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.

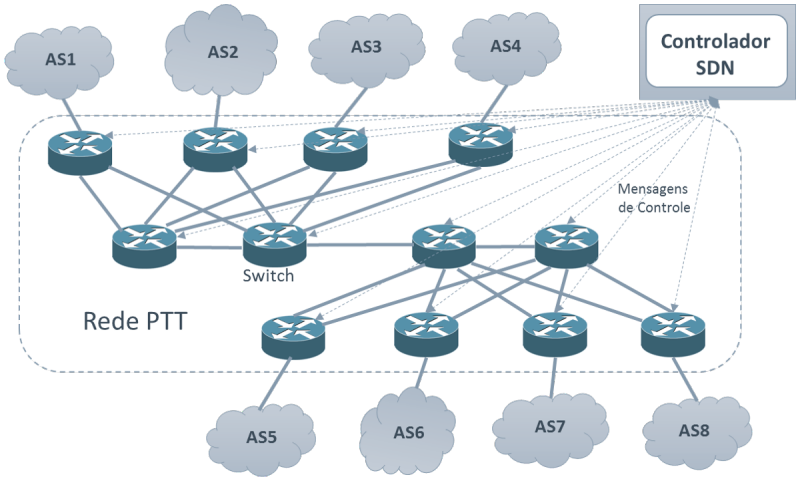


Figura 1. Cenário de uma rede de PTT com *switches* programáveis.

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.

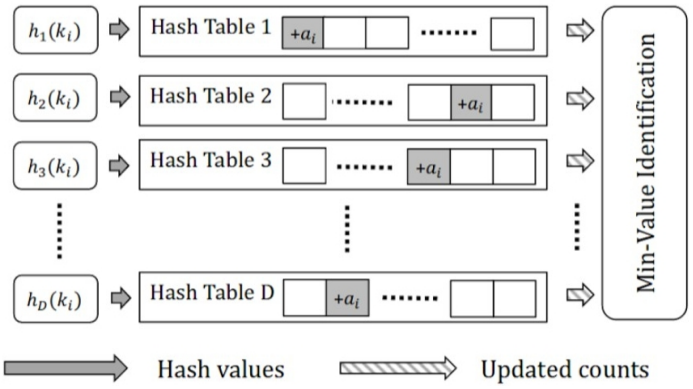


Figura 2. Mapeamento *hash* de múltiplos níveis [Tong and Prasanna 2015].

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.

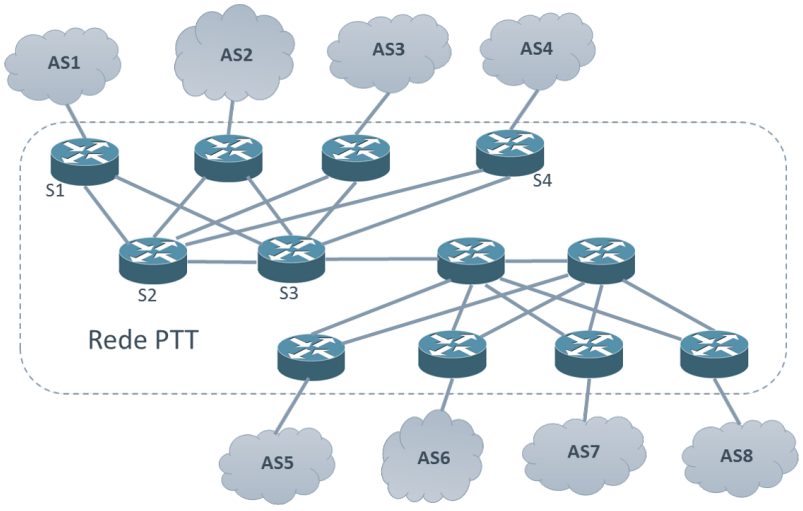


Figura 3. Cenário de Avaliação.

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*.

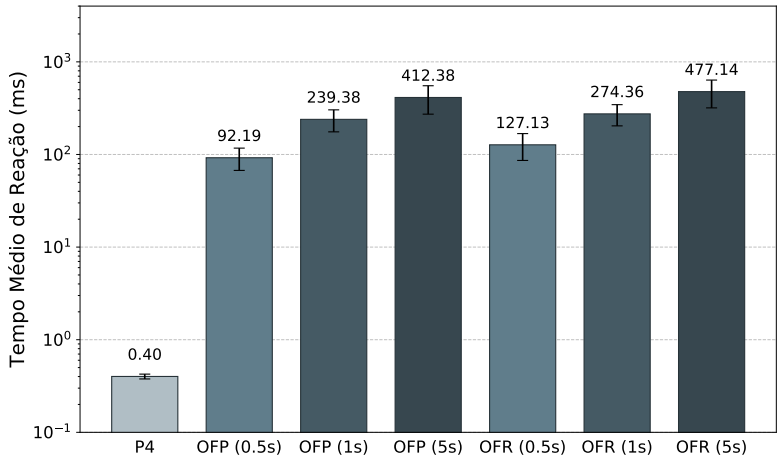


Figura 4. Tempo de reação.

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%.

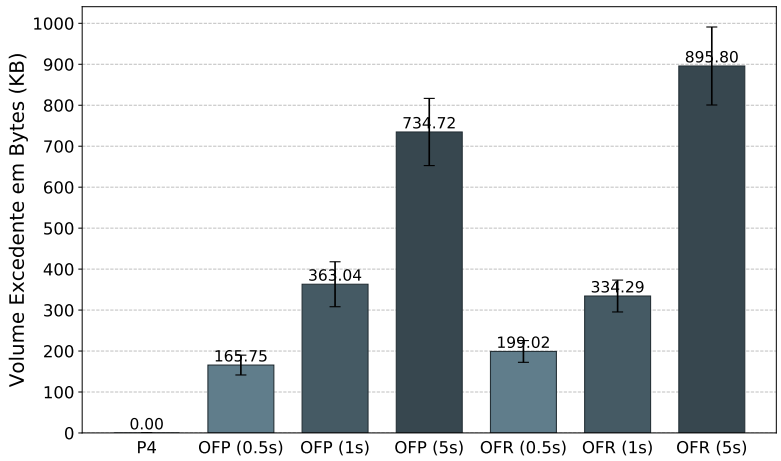
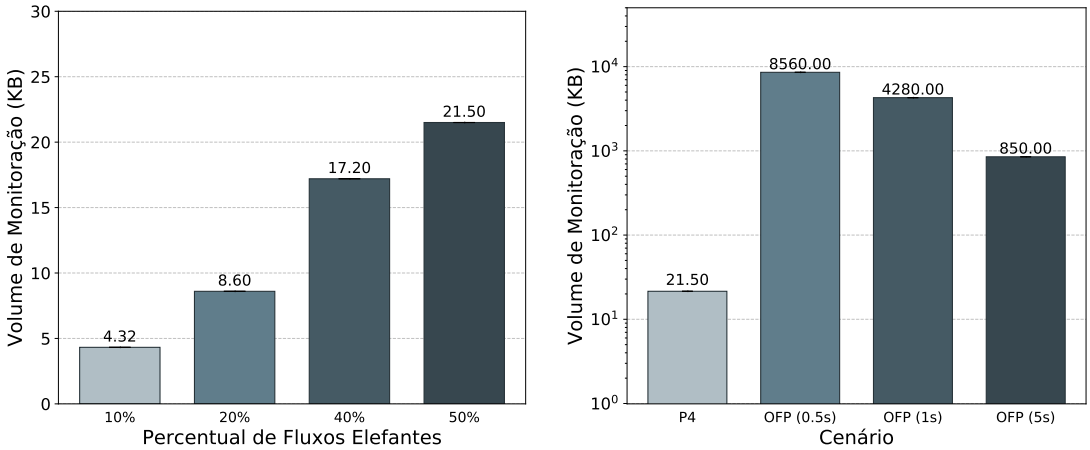


Figura 5. Volume de *bytes* excedentes.

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.

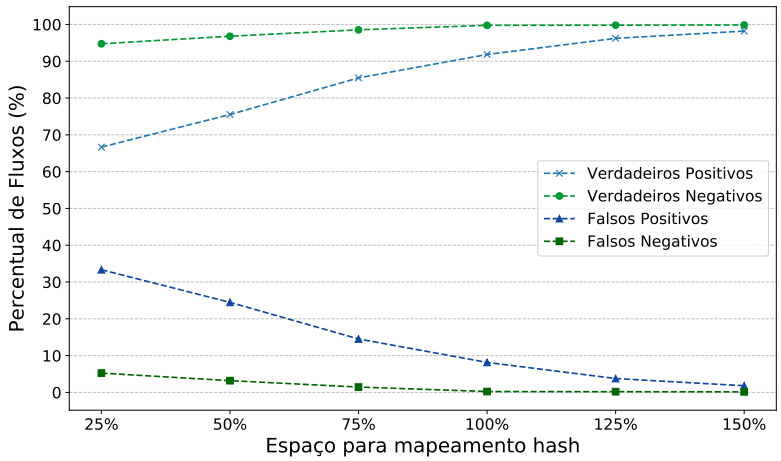


Figura 7. Acurácia do mecanismo em função do espaço de mapeamento Hash.

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.