MTS-PolKA: Divisão de Tráfego Multicaminhos em Proporção de Peso com Roteamento na Fonte

Giancarlo O. dos Santos[1], Cristina K. Dominicini[1], Gilmar L. Vassoler[1], Rafael S. Guimarães[1], Isis Oliveira[1], Domingos Jose P. Paraiso[1], Rodolfo S. Villaca[2]

[1]Programa de Pós-Graduação em Computação Aplicada (PPComp)
Instituto Federal do Espírito Santo (Ifes)
Serra, ES - Brasil

[2]Programa de Pós-Graduação em Informática (PPGI)
Universidade Federal do Espírito Santo (Ufes)
Vitória, ES - Brazil

{giancarlo, cristina.dominicini, gilmarvassoler, rafaelg}@ifes.edu.br

{isisolip, domingos.paraiso}@gmail.com, rodolfo.villaca@ufes.br

Abstract. *The article presents the innovative proposal called MTS-PolKA to optimize traffic in datacenter networks. Introduces a dynamic method of dividing traffic with labels (routeIDs and weightIDs) in packet headers, using static tables in switches to allow flexible adjustments in real time, eliminating complex reconfigurations. The approach employs source routing with the modified M-Polka Protocol, using a residual number system for stateless source routing and no changes to the final hosts. MTS-PolKA stands out for its agility in (re)configuring paths and weights, with the control plane calculating route identifiers (routeIDs), weight (weightIDs) and node (nodeIDs). Experiments demonstrate the effectiveness of the solution, enabling agile reconfigurations of traffic division profiles at the source, with the potential for improving performance and efficiency in data center networks.*

Resumo. *O artigo apresenta a proposta inovadora chamada MTS-PolKA para otimizar o tráfego em redes de datacenters. Introduz um método dinâmico de divisão de tráfego com rótulos (routeIDs e weightIDs) no cabeçalho dos pacotes, utilizando tabelas estáticas nos switches para permitir ajustes flexíveis em tempo real, eliminando reconfigurações complexas. A abordagem emprega o roteamento na origem com o Protocolo M-Polka modificado, utilizando um sistema numérico de resíduos para roteamento de fonte sem armazenamento de estado e sem alterações nos hosts finais. O MTS-PolKA destaca-se pela agilidade na (re)configuração de caminhos e pesos, com o plano de controle calculando identificadores de rota (routeIDs), peso (weightIDs) e nó (nodeIDs). Experimentos demonstram a eficácia da solução, possibilitando reconfigurações ágeis de perfis de divisão de tráfego na origem, com potencial de melhorar o desempenho e eficiência em redes de datacenters.*

1. Introdução

Nos últimos anos, os *datacenters* se tornaram os pilares da infraestrutura de computação da Internet. Inúmeros aplicativos de processamento distribuído (como colaboração social e

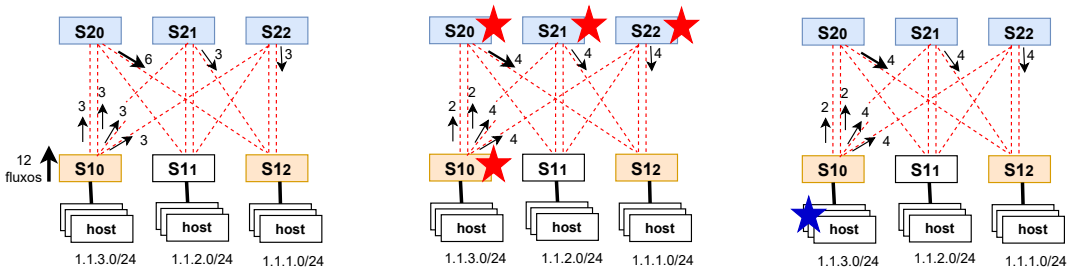


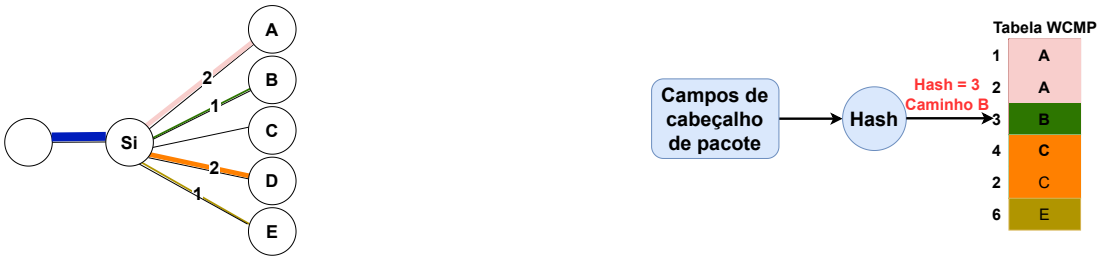
Figura 1. Divisão assimétrica de tráfego multicaminho em topologias de árvore multi-raiz do tipo Clos Tier 2. (a) ECMP: caminhos estáticos de mesmo peso. (b) WCMP: alterações nos pesos exigem alteração nas tabelas em cada *switch* no caminho. (c) *MTS-PolKA*: alterações nos pesos não requerem modificações nas tabelas de cada *switch* ao longo do caminho; elas são definidas exclusivamente na origem e inseridas no cabeçalho do pacote.

computação de alto desempenho) são executados diariamente em *datacenters* de grande escala que contêm mais de 100.000 servidores [Dixit et al. 2013]. Os *datacenters* modernos são comumente organizados em topologias de árvore multi-raiz, como *Fat-Tree* e *Clos*, para fornecer vários caminhos com o mesmo custo entre esses servidores [Huang et al. 2021].

No entanto, utilizar toda a largura de banda disponível requer uma maneira eficaz de equilibrar o tráfego, especialmente quando a carga nesses caminhos pode se alterar rapidamente devido às flutuações de tráfego. Portanto, conseguir balancear a carga em uma pequena escala de tempo, com base nas condições de tráfego dos caminhos, torna-se crucial para obter um bom desempenho neste gerenciamento [Hsu et al. 2020]. Assim, para que a arquitetura de rede de *datacenter* alcance um alto desempenho, a chave é a engenharia de tráfego, em que o controlador de rede ou os agentes distribuídos devem ser capazes de selecionar caminhos e balancear a carga entre eles, de forma ágil, para adaptar-se a padrões de tráfego com comportamentos dinâmicos [Jyothi et al. 2015].

Os algoritmos tradicionais baseados em *ECMP (Equal Cost Multiple Path)* consideram apenas topologias simétricas e regulares, devido à sua estratégia de roteamento estática baseada apenas em caminhos com a mesma capacidade de banda e mesma quantidade de saltos [Hsu et al. 2020]. Esse requisito não é realista para a maioria dos cenários atuais de redes de *datacenter*, pois pode ser violada pelas seguintes condições [Zhou et al. 2014]: i) mudanças rápidas nas condições de tráfego; ii) ocorrência de falhas nos nós ou enlaces da rede; iii) existência de componentes de rede heterogêneos; e iv) topologias assimétricas. A Figura 1 ilustra um exemplo de uma topologia de árvore multi-raiz do tipo *Clos Tier 2*, que pode oferecer multicaminhos de mesmo comprimento para um mesmo destino, mas, com diferentes larguras de banda. Isso pode acontecer devido a diferentes capacidades dos enlaces ou condições dinâmicas de concorrência de tráfego, e vai exigir uma distribuição assimétrica de tráfego na rede. Na Figura 1(a), é possível perceber que a estratégia de *hashing* do *ECMP* não é capaz de resolver de forma eficiente esse problema.

Outras abordagens existentes para resolver este problema são o *WCMP* (*Weighted-Cost MultiPath*) [Cui et al. 2021], baseado na configuração de tabelas de divisão de tráfego, e o *DASH* (*Adaptive Weighted Traffic Splitting in Programmable Data Planes*) [Hsu et al. 2020], baseada em uma estrutura de dados otimizada que explora registra-



((a)) Distribuição de pesos 2:1:2:1.

((b)) *WCMP*: Tabelas com pesos.

Figura 2. Problema: necessidade de realizar uma reconfiguração nos *switches* para permitir a distribuição assimétrica de tráfego nos multicaminhos.

dores em um pipeline com múltiplos estágios para atualizar a distribuição em tempo real.

Conforme mostrado na Figura 1(b), essas soluções permitem explorar múltiplos caminhos, alocar ativamente o fluxo para o caminho apropriado conforme a razão de peso desejada, e utilizar eficientemente a largura de banda dos enlaces disponíveis. Entretanto, um problema permanece: a necessidade de modificar essas tabelas ou estruturas de dados de cada *switch* quando for necessário alterar os pesos dos caminhos. Como exemplo, ainda na Figura 1(b), isso significaria alterar todas as tabelas dos dispositivos de rede indicados com estrelas vermelhas, para cada fluxo sendo reorganizado. Quanto maior a quantidade de *switches* no caminho, e de fluxos sendo gerenciados, maior será o tempo necessário para convergência das tabelas dos dispositivos envolvidos nesta operação, impactando diretamente na agilidade desta operação. Em síntese, uma engenharia de tráfego eficiente em redes de *datacenter* requer: i) distribuir os fluxos de forma ágil, explorando os multicaminhos, proporcionalmente à largura de banda disponível em cada caminho, e ii) ajustar dinamicamente, também de forma ágil, os fluxos por esses caminhos.

Para atender a esta demanda, o objetivo deste artigo é propor uma solução que permita configurar, na velocidade requerida pelas atuais redes de *datacenter*, como cada *switch i* pertencente a um caminho de rede *r* deve dividir o tráfego nas suas portas de saída. Além disso, essa divisão deve obedecer proporções de pesos pré-configuradas para cada subconjunto *p* de suas portas. A Figura 2(a) mostra um exemplo ilustrativo, de parte da proposta: na figura, um switch Si, com 5 portas, deve distribuir o tráfego de um fluxo *f* qualquer, configurado no ingresso da rede, conforme a proporção de pesos 2:1:2:1 para as portas o p= {A, B, D e E}. Neste ponto, é importante ressaltar que a proposição de mecanismos para o monitoramento de fluxos não faz parte do escopo deste trabalho.

A solução a ser apresentada neste trabalho se baseia na arquitetura *M-PolKA* (*Multipath Polynomial Key-based Architecture*) [Guimarães et al. 2022], e propõe uma abordagem de divisão de tráfego multicaminho denominada *MTS-PolKA* (*Multipath Traffic Split Polynomial Key-based Architecture*). O *MTS-PolKA* permite selecionar, na origem, um perfil de divisão de tráfego em proporção de pesos para cada fluxo de rede, por meio de um rótulo no cabeçalho do pacote. Cada *switch* no caminho mantém uma tabela estática com os perfis de divisão de tráfego disponíveis. A inovação está na capacidade de alterar a distribuição do tráfego, com alta granularidade, simultaneamente em todos os *switches* do caminho, por meio de uma simples modificação de rótulo no cabeçalho dos pacotes de cada fluxo ingressante na rede. Esse método elimina a necessidade de se reconfigurar tabelas ou estruturas de dados em cada *switch*, e permite uma engenharia de tráfego mais

ágil, eficiente e dinâmica. em redes de *datacenters*. Considerando o exemplo exposto na Figura 1(c), isso significaria alterar apenas o cabeçalho dos pacotes em um único local onde o tráfego é gerado (estrela azul) para que essa alteração seja refletida em toda a topologia. As principais contribuições deste trabalho *MTS-PolKA* incluem:

- **Extensão do roteamento de origem multicaminho M-PolKA, para permitir a configuração de perfis de divisão de tráfego personalizadas:** A arquitetura M-PolKA permite que a origem selecione os múltiplos caminhos de um fluxo conforme uma distribuição que usa uma árvore *multicast*, sem reconfigurações complexas nas tabelas de comutação, apenas inserindo rótulos no ingresso dos pacotes da rede. Entretanto, o M-PolKA apenas clona os pacotes para todas as portas que pertencem ao caminho selecionado, sem permitir que se configure uma forma de distribuição do tráfego personalizada. Esta lacuna foi preenchida pelo *MTS-PolKA*.
- **Divisão dinâmica de tráfego:** Ao contrário das abordagens tradicionais de divisão de tráfego, que exigem modificações nas tabelas de comutação ou nas estruturas de dados, o *MTS-PolKA* permite que a fonte selecione dinamicamente os perfis de distribuição de tráfego para cada um desses fluxos usando um rótulo de rota e outro rótulo de pesos. Isso permite alterações simultâneas na distribuição de tráfego em todos os *switches* no caminho.
- **Implementação eficiente no plano de dados de *switches* programáveis:** O *MTS-PolKA* utiliza o Sistema Numérico de Resíduos (do inglês, *RNS, Residue Number System*) [Guimarães et al. 2022] e tabelas estáticas, previamente configuradas, para definir os perfis de tráfego disponíveis na rede, eliminando a necessidade de se alterar informações de estado em cada *switch* no caminho selecionado, simplificando as operações de engenharia de tráfego e possibilitando uma implementação eficiente em *switches* programáveis.

Como prova de conceito, implementou-se o *MTS-PolKA* na linguagem P4 e avaliou-se a solução proposta usando o emulador *Mininet*. Diversos experimentos foram conduzidos para demonstrar o funcionamento do *MTS-PolKA*. Os resultados evidenciam a expressividade da nossa proposta para explorar multicaminhos de rede, mantendo a estabilidade do fluxo e possibilitando reconfigurações ágeis de perfis de divisão de tráfego na origem, com potencial de melhorar o desempenho e a eficiência em redes de *datacenters*.

O restante deste trabalho está organizado da seguinte forma. A Seção 2 apresenta alguns trabalhos relacionados, evidenciando as contribuições deste trabalho. A Seção 3 descreve o projeto e a implementação do *MTS-PolKA*. A Seção 4 apresenta e discute os resultados dos experimentos. Por fim, a Seção 5 traz as conclusões e trabalhos futuros.

2. Trabalhos Relacionados

O método mais popular para balancear os fluxos em diferentes caminhos é o *ECMP*, que usa *hashing* para distribuir fluxos igualmente entre caminhos de mesmo tamanho, apesar de ter pouca aplicação em topologias assimétricas de caminhos [Valentim et al. 2019]. O *WCMP*, por sua vez, generaliza o *ECMP* para permitir distribuições de carga não uniformes, conforme exemplificado na Figura 2(b). O *WCMP* utiliza uma estrutura de dados baseada em *hashing* e, atualizá-la requer a modificação dos IDs de caminho em múltiplas tabelas. Atualmente, essas alterações se mostram inviáveis devido ao número

Tabela 1. Trabalhos Relacionados de Divisão de Tráfego em Redes de Datacenter

Método	Núcleo sem estado	Granularidade			Método de Roteamento	Divisão dos Fluxos		Desvantagens
		Fluxo	Subfluxo	Pacote		Cam. Rede	Cam. Transp.	
ECMP	✗				*Tabela Hash*			*Colisões na tabela hash.*
WCMP	✗	✓			*Tabela Hash*	✓		*Modificações em tabelas.*
HULA	✗		✓		*Tabela Hash*	✓		*Único melhor caminho.*
DASH	✗		✓		*Hash c/ registradores*	✓		*No. de caminhos suportados*
MP-TCP + SR	✓		✓		*SR*		✓	*Alterações no host final.*
RPS	✗			✓	*Random*	✓		*Pacotes fora de ordem.*
QDAPS	✗			✓	*Tabela Hash*	✓		*Assimetria de topologia.*
MTS-Polka	✓		✓		*SR*	✓		*Tamanho do cabeçalho (bits).*

limitado de acessos à memória, por pacote, em cada estágio do *pipeline* de um dispositivo programável [Rottenstreich et al. 2018].

Conforme definido em [Hsu et al. 2020], trabalhos recentes como *CONGA*, *HULA* e *Contra* são abordagens de balanceamento de carga que realizam o balanceamento de carga diretamente no plano de dados, sem intervenção do plano de controle, atualizando dinamicamente o estado do plano de dados. Entretanto, essas soluções também apresentam desvantagens, pois consideram apenas um "melhor"caminho de cada vez, podendo subutilizar os outros "melhores"caminhos. O *DASH*, por sua vez, permite o espalhamento dos fluxos por vários caminhos, proporcionalmente à carga atual, ajustando-se dinamicamente às mudanças de carga [Hsu et al. 2020]. Entretanto, o *DASH* apresenta uma restrição do número de caminhos suportados [Robin and Khan 2022]. Além disso, também depende de modificações de estado nos dispositivos da rede.

Outra solução, disponível na literatura, é a integração do *MP-TCP (MultiPath TCP)* com o roteamento de origem *(SR, Source Routing)*. No entanto, a solução *MP-TCP + SR* requer configurações específicas nos *hosts* finais [Pang et al. 2017], e sua aplicabilidade pode ser limitada em ambientes que não permitem mudanças significativas na pilha de protocolos dos *hosts*, tais como em plataformas de nuvem pública. Outras abordagens tentam solucionar o problema de divisão de tráfego multicaminhos, fazendo a divisão dos pacotes de um fluxo baseadas em estratégias de pulverização. Por exemplo, o *Random Packet Spraying* (*RPS*) [Dixit et al. 2013] Entretanto, essa técnica possui um problema associado à reordenação de pacotes que pertencem a um fluxo, conhecido por interagir negativamente com o controle de congestionamento TCP. Já o *Queuing Delay Aware Packet Spraying (QDAPS)* [Huang et al. 2021] busca mitigar esse problema, mas ainda é bastante sensível à assimetria de topologia.

A Tabela 1 sumariza a comparação das diferentes estratégias de divisão de tráfego multicaminhos em redes de *datacenter*, considerando como parâmetros: granularidade, presença de um núcleo sem estado, método de roteamento, localização da distribuição do fluxo e desvantagens associadas a cada proposta. A presença de um núcleo sem estado indica alta agilidade de (re)configuração de caminhos e pesos, na velocidade demandada pelas aplicações suportadas pelos planos de dados atuais, sendo essa a principal característica distintiva do *MTS-PolKA*.

3. Projeto e Implementação do *MTS-PolKA*

O *MTS-PolKA* deve permitir que a origem selecione a divisão de tráfego desejada, nos multicaminhos, usando rótulos no cabeçalho dos pacotes dos fluxos, e cada *switch* nos

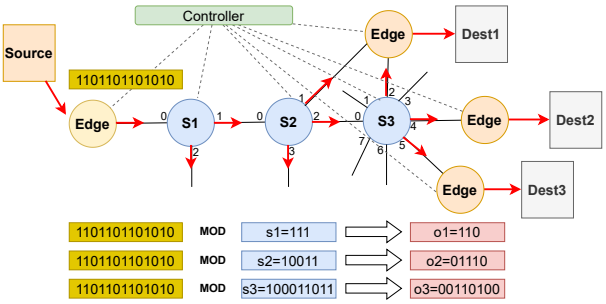


Figura 3. Exemplo do roteamento M-PolKA original.

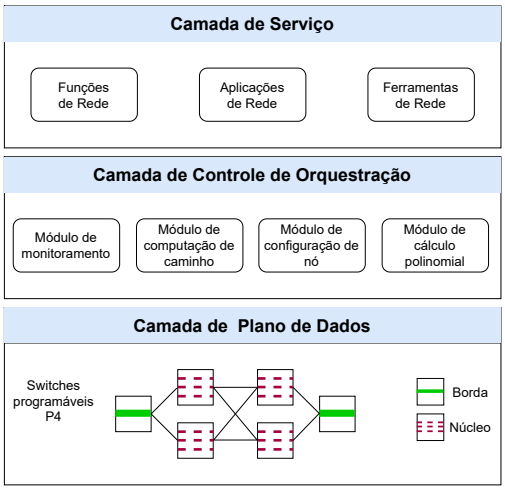


Figura 4. Arquitetura *MTS-PolKA*.

caminhos selecionados deve manter uma tabela estática com um conjunto de perfis de divisão de tráfego disponíveis para seleção.

3.1. Roteamento M-Polka

O *MTS-PolKA* utiliza o protocolo *M-PolKA* para conseguir um roteamento de fonte sem estados baseado em *RNS*, onde as decisões de encaminhamento no núcleo dependem de operações de módulo polinomial sobre um rótulo de rota. O *M-PolKA* foi escolhido por ser o único método de roteamento de fonte encontrado na literatura capaz de codificar um rótulo de rota para um ciclo de multicaminho ou árvore de maneira agnóstica à topologia.

Como mostrado na Figura 3, a arquitetura do *MTS-PolKA* envolve *switches* de núcleo (`Si`), de borda (`Edge`), e um controlador centralizado (`Controller`), permitindo o roteamento multicaminhos por meio de 3 (três) identificadores polinomiais com representação binária: *routeID*, *nodeID* e *portID*. O estado de transmissão das portas de saída (*portID*) é fornecido pelo resto da divisão polinomial binária (isto é, uma operação de módulo `MOD`) entre o identificador de rota (*routeID*) e o identificador do nó (*nodeID*). No exemplo da Fig. 3, o controlador instala regras de fluxo na borda para incorporar o identificador de rota, *routeID* = 10101100101100 nos pacotes de um fluxo. O resultado das operações de módulo nos *switches* de núcleo está representado pelas setas em vermelho. Por exemplo, o resto de $R(t) = 10101100101100$ quando dividido por $s_2(t) = 10011$ é $o_2(t) = 0110$. Assim, em s_2, as portas 1 e 2 encaminham esse fluxo para o próximo salto, enquanto as portas 0 e 3 não o encaminham.

Entretanto, o *M-PolKA* original clona os pacotes para todas as portas ativas sem fazer nenhuma divisão de tráfego. De forma diferente, este trabalho propõe a aplicação do *M-PolKA* para dividir o tráfego com proporção de peso entre as portas de saída. Isso permite explorar a diversidade de caminhos para fornecer balanceamento de carga.

3.2. Arquitetura do *MTS-PolKA*

O controlador *MTS-PolKA* calcula três identificadores polinomiais: i) ***nodeID***, um identificador atribuído aos *switches* do núcleo na configuração da rede; ii) ***routeID***, um identificador de rota multicaminhos; e iii) ***weightID***, um identificador de peso que define o perfil de

divisão de tráfego em cada *switch* no caminho do tráfego.Tanto o *routeID* quanto o *weightID* são calculados por meio do Teorema Chinês dos Restos (CRT, do inglês *Chinese Remainder Theorem*) [Dominicini et al. 2020] e incorporados aos pacotes pelos *switches* de borda, que possuem regras de fluxo que podem ser instaladas de forma proativa pelo controlador, sem a necessidade de se consultar o plano de controle para cada pacote que ingressa na rede.

No plano de dados, a implementação do *mod* polinomial explora a operação de CRC (do inglês, Cyclic Redundancy Check) suportada pelo equipamento. Como detalhado em [Guimarães et al. 2022], o tamanho do *routeID* é dado pela multiplicação do número máximo de saltos da topologia por 2 bytes (para uso do algoritmo CRC16 em topologias com até 4.080 nós). A mesma lógica pode ser adotada para o cálculo do *weightID*. A implementação deste artigo adotou um número máximo de 10 saltos com um cabeçalho de 20 bytes para cada campo de *routeID* e *weightID*. Assim, existe um custo em termos de bits no cabeçalho para explorar o sistema RNS no roteamento, mas que pode ser reduzido aplicando algumas técnicas de codificação conhecidas [Dominicini et al. 2020].

Além disso, dois identificadores polinomiais são calculados, no plano de dados, para cada pacote: i) ***portID***, um identificador que representa as portas de saída ativas de cada *switch* de núcleo, ou seja, quais portas transmitirão uma proporção do tráfego a ser encaminhado; e ii) ***profileID***, um identificador obtido pela operação de módulo entre o *weightID* do pacote e o *nodeID*, que representa o perfil de proporção de divisão de tráfego nas portas de saída ativas do *switch*. Em resumo, a arquitetura *MTS-PolKA* é composta por nós no núcleo da rede (em azul), nós de borda (em amarelo) e um controlador logicamente centralizado, conforme Figura 4. As camadas da arquitetura são descritas a seguir:

- **Camada do Plano de Dados:** Assume uma topologia física composta por *switches* programáveis habilitados para P4, atuando como nós de borda ou núcleo para a divisão de tráfego multicaminho em proporção de peso e roteamento na fonte.
- **Camada de Controle e Orquestração:** Composta por quatro módulos: i) `Monitoramento`: descobre a topologia e coleta dados sobre o estado da rede, tais como falha de enlace e congestionamento, a fim de fornecer um ciclo de *feedback* para decisões de orquestração; ii) `Computação de caminho`: calcula múltiplos caminhos para um fluxo, considerando opções de nós e enlaces, disjuntos ou não disjuntos; iii) `Configuração de nó`: configura *nodeID*s e tabelas de perfis de tráfego nos nós de núcleo, além de configurar as entradas de fluxo em nós de borda; e iv) `Cálculo polinomial`: calcula *weightID*s, *routeID*s e *nodeID*s para os caminhos e distribuições de tráfego definidos.
- **Camada de Serviço:** Responsável pela implementação de funções de rede, aplicações do usuário e demais ferramentas utilizadas nas etapas de implantação e validação da arquitetura.

3.3. Plano de Controle

Como ilustrado na Figura 5, o controlador *MTS-PolKA* tem duas atribuições principais: i) definir a rota, baseando-se na definição de uma árvore *multicast* que identifica, para cada um dos *switches* no caminho, quais são as portas ativas que devem encaminhar os pacotes, conforme representação desta árvore *multicast*; e ii) definir os pesos para divisão de tráfego em cada porta ativa de cada um dos nós da árvore *multicast*.

No núcleo da rede, o controlador é responsável por configurar os *switches* com

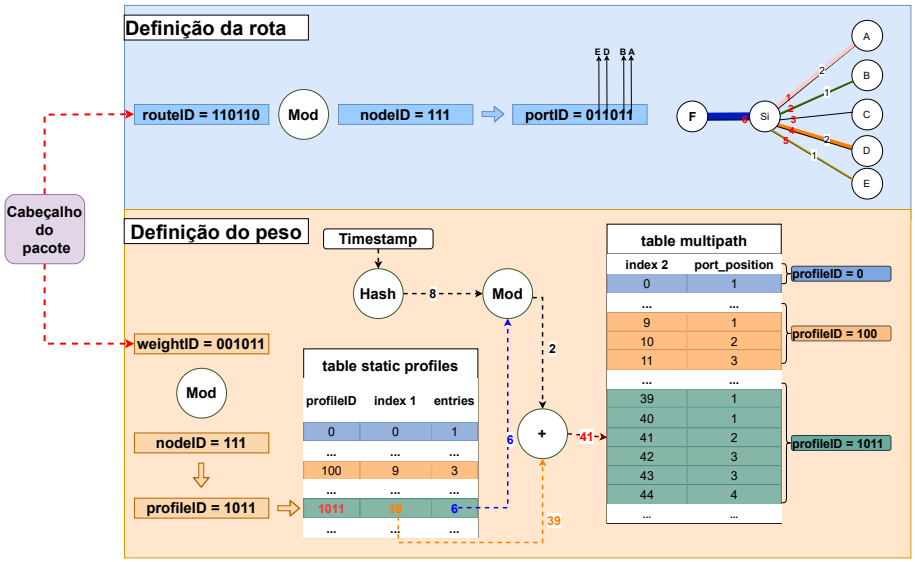


Figura 5. Exemplo de funcionamento do *MTS-PolKA* com perfil de divisão de tráfego selecionado pela origem com um rótulo no cabeçalho do pacote. Ambas as tabelas são configuradas previamente pelo controlador.

os *nodeID*s, e as tabelas estáticas de cada *switch* com os perfis de divisão de tráfego disponíveis. Na borda, por sua vez, o controlador disponibiliza um conjunto de multicaminhos e perfis de divisão de tráfego por meio dos *routeID*s e *weightID*s, que podem ser escolhidos pelos fluxos ingressantes, conforme o estado da rede, direcionando e realizando o baleanceamento do tráfego. Essa solução explora a diversidade de caminhos disponíveis na rede, resultando em uma utilização eficiente da largura de banda disponível na rede.

3.4. Plano de Dados

Cada *switch* de borda possui uma tabela de fluxos, previamente preenchida pelo controlador, que insere os rótulos *routeID* e *weightID* nos pacotes de cada fluxo. Essa tabela mapeia informações sobre o fluxo (e.g., endereço IP de destino, portas) em decisões de roteamento e balanceamento de carga. Posteriormente, esses rótulos são usados em cada *switch* de núcleo para determinar o estado das portas de saída e selecionar os perfis de divisão de tráfego correspondentes, conforme Figura 5.

A Figura 5 exemplifica o funcionamento do plano de dados no *MTS-PolKA* para um *switch* de núcleo *Si* que possui 6 portas e distribuição de tráfego com proporção de pesos 2:1:2:1 para as portas que encaminham para os *switches* A, B, D e E, respectivamente. No ingresso do pacote, deve ser realizada a operação de `Definição de rota` usando a operação de módulo (`MOD`) entre o *routeID* = 110110 e o *nodeID* = 111. O resultado desta operação serve para definir as portas de saída ativas (*portID* = 011011) que serão usadas no encaminhamento do pacote. Dessa forma, o tráfego é encaminhado pelas portas correspondentes aos *bits* de valor 1 em *portID*, com a análise realizada da direita para a esquerda. Assim, os pacotes são transmitidos para os *switches* A, B, D e E.

Em seguida, é necessário descobrir qual o perfil de divisão de tráfego será usado no encaminhamento deste pacote para cada porta de saída ativa, conforme o *weightID* definido no pacote. Previamente, o controlador configurou, em cada *switch* de núcleo, o seu *nodeID* e duas tabelas estáticas (`table static profiles` e `table multipath`),

que definem, respectivamente, os perfis de tráfego suportados e como o tráfego deve ser distribuído pelas portas ativas conforme o perfil selecionado. Dessa forma, existem diversos perfis de tráfego disponíveis em cada *switch* de núcleo, que podem ser selecionados pela borda para os pacotes de cada fluxo, sem nenhuma configuração adicional nos *switches* de núcleo. A quantidade e a variedade de perfis suportados é uma decisão do plano de controle.

Seguindo no exemplo da Figura 5, a tabela `table static profiles` determina "como"a próxima tabela (`table multipath`) deverá ser acessada conforme o perfil de divisão de tráfego selecionado. Ao empregar a operação de módulo entre *weightID* = 001011 e *nodeID* = 111, o *switch Si* obtém-se o *profileID* = 1011. Para esse perfil, na tabela `table multipath` existem 6 entradas (*entries*) que se iniciam a partir da posição 39 (*index1*). Na tabela `table multipath`, o perfil *profileID* = 1011 está representado na cor verde e a quantidade de linhas define os pesos para cada porta de saída ativa: 2 entradas para a primeira porta ativa ($port\_position = 1$), 1 entrada para a segunda porta ativa ($port\_position = 2$), 2 entradas para a terceira porta ativa ($port\_position = 3$) e 1 entrada para a quarta porta ativa ($port\_position = 4$), com um total de 6 entradas. Deste modo, o perfil *profileID* = 1011 deve dividir o tráfego na seguinte proporção: a primeira porta ativa deve receber 2/6 do tráfego, a segunda porta ativa deve receber 1/6 do tráfego, a terceira porta ativa deve receber 2/6 do tráfego e a quarta porta ativa deve receber 1/6 do tráfego. O número de entradas na `table multipath` representa justamente a proporção esperada e será explorada por uma função de *hashing*.

Para espalhar os pacotes de um fluxo nas portas de saída ativas conforme a proporção definida no perfil, o *switch Si* executa uma função de *hashing* com o *timestamp* de ingresso do pacote. Neste exemplo, considere que o resultado do *hashing* é 8, e que este valor é submetido a uma operação de módulo inteiro pelo número de entradas em verde ($entries = 6$ na `table multipath`), cujo resultado = 2. Este resultado deve ser adicionado ao índice ($index1 = 39$), obtido na tabela `table static profiles`, resultando no índice $index2 = 41$ da `table multipath`. Por fim, essa linha do $index2 = 41$ indica que a segunda porta ativa ($port\_position = 2$) deve ser escolhida como porta de saída deste pacote específico. Neste exemplo, como o *portID* = 011011 foi previamente calculado na etapa de definição da rota, a segunda porta ativa significa que o pacote deve ser encaminhado pela porta de índice 2 (índices das portas começam em 1, da direita para a esquerda no *portID*). Ainda na figura, a porta 2 representa o caminho de saída do pacote para o nó B. Aplicando o algoritmo de seleção de portas descrito até aqui, e considerando a variação aleatória do *hashing* do *timestamp* de ingresso dos pacotes, o tráfego associado ao perfil *profileID* = 1011 será espalhado entre as portas ativas com uma proporção de pesos igual a 2:1:2:1.

A Figura 6 apresenta a mesma lógica do exemplo da Figura 5, mas representando a implementação usando a linguagem P4 do *pipeline* do plano de dados dos *switches* de núcleo no *MTS-PolKA*. Na etapa de *parsing*, se o campo *etherType* for `TYPE SR`, indicando a presença do cabeçalho *MTS-PolKA*, a próxima etapa é a leitura dos rótulos *routeID* e o *weightID*, via *lookahead*. Com base nos valores lidos, o bloco de *ingress* realiza operações de módulo polinomial entre o *routeID* e o *nodeID* e entre o *weightID* e o *nodeID*, para calcular os estados das portas de transmissão (*portID*) e o perfil de divisão de tráfego (*profileID*), respectivamente. Depois, o *profileID* é utilizado para busca na tabela `table static profiles` para recuperar os valores das variáveis *entries* e *index1*.

Esses valores são usados em uma operação de módulo inteiro com o *hashing* do *timestamp* do pacote para obter a variável $index2 = index1 + hash(tstamp) \mod entries$. Então, este último índice é usado para acessar a tabela `table multipath` e identificar qual porta ativa deve transmitir aquele pacote específico (*port_position*). Por fim, é feita uma busca por qual é a porta ativa no vetor de estados que corresponde ao valor de *port_position*, e, uma vez encontrada, o pacote é transmitido por essa porta.

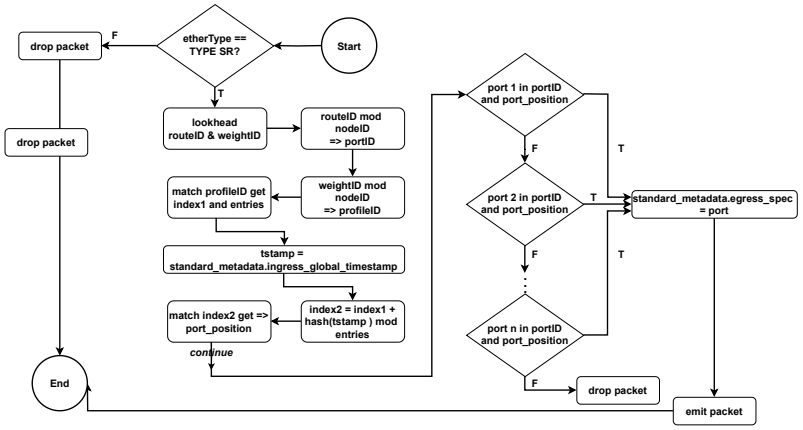


Figura 6. *Pipeline* na linguagem P4 dos *switches* de núcleo do *MTS-PolKA*.

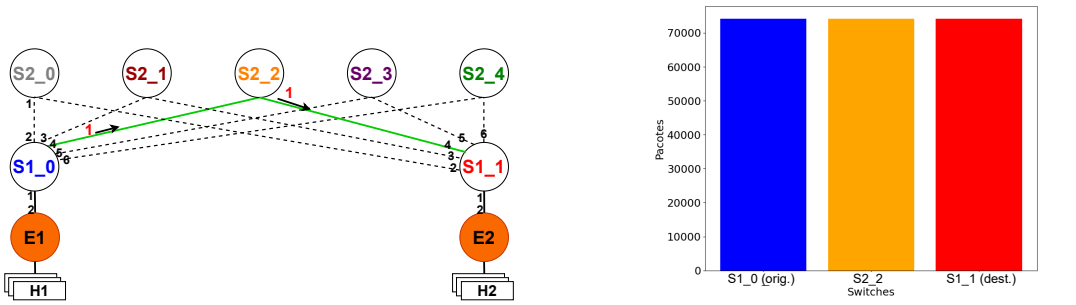
4. Avaliação Experimental do *MTS-PolKA*

O ambiente de desenvolvimento e testes usou um servidor *PowerEdge* R650, com 32 GB de *RAM* e processador Intel *Xeon Silver* 4314 de 2.4 GHz. O emulador *Mininet* foi utilizado para conduzir os experimentos, permitindo a compilação de programas P4-16 e configuração de *switches bmv2*. Para realizar os cálculos polinomiais, como o cálculo *RNS* de *nodeID*s e *routeID*s, utilizamos uma biblioteca *Python*, com o pacote *galoistools* da biblioteca *sympy*. Além disso, foi implementado um aplicativo de plano de controle que interage com os *switches* através da CLI do *bmv2 simple_switch* e se conecta via porta *Thrift RPC*. Esse aplicativo implementa a camada de Controle e Orquestração, conforme seção 3.2, com exceção do módulo de monitoramento que está fora do escopo deste artigo.

Foram realizados três experimentos: i) um teste funcional, para demonstrar a aplicação de diferentes perfis de divisão de tráfego; ii) um teste de reconfiguração ágil de perfis de divisão de tráfego na origem; e iii) um teste com múltiplos fluxos concorrentes. O teste envolveu medições usando a ferramenta *bwm-ng* e foi gerado via *iperf3*, tendo um *host* como servidor e outro como cliente. Nos testes 1 e 2, foram enviados pacotes UDP com vazão de 10 Mbps. Já no teste 3, foram enviados fluxos TCP e UDP com vazão de 4 Mbps.

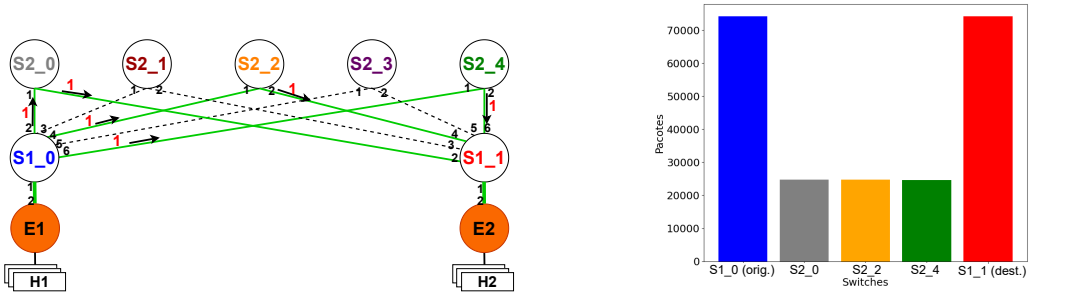
4.1. Teste 1: Validação Funcional

Considere o seguinte cenário: três diferentes perfis de tráfego foram selecionados na origem para um *switch* de núcleo específico (`S1_0`), que representa, por exemplo, um *switch* de acesso em uma rede de *datacenter* com topologia hierárquica. Os três perfis são representados nas Figuras 7(a), 8(a) e 9(a) que serão melhor detalhados posteriormente nesta seção. Os demais *switches* mantiveram uma distribuição de tráfego constante usando o Perfil 0, que representa, neste exemplo, direcionar todo o tráfego pela primeira porta ativa. Conforme já explicado, o perfil de divisão de tráfego foi determinado pela origem, e



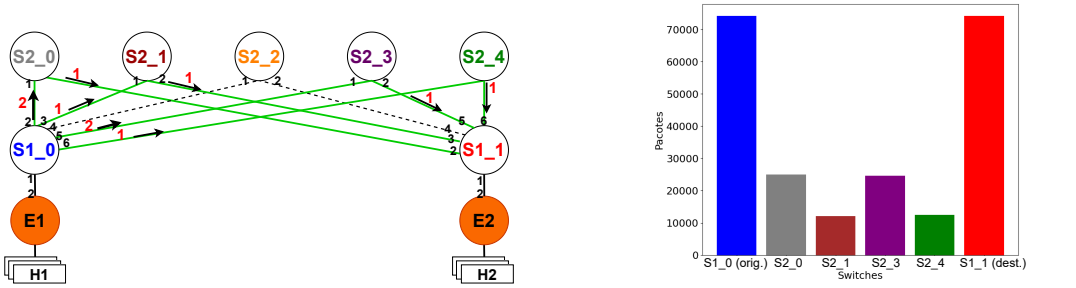
((a)) Cenário perfil 0 ((b)) Número de pacotes por switch

Figura 7. Teste 1: perfil 0



((a)) Cenário perfil 100 ((b)) Número de pacotes por switch

Figura 8. Teste 1: perfil 100



((a)) Cenário perfil 1011 ((b)) Número de pacotes por switch

Figura 9. Teste 1: perfil 1011

inserido no fluxo pelo *switch* de borda, por meio da utilização de dois rótulos no cabeçalho do pacote: i) o *routeID*, para selecionar as portas ativas em cada *switch* no caminho e montar a árvore de transmissão; e ii) *weightID*, para selecionar a proporção de pesos.

Teste com o Perfil 0 (Figura 7): Usando a árvore de transmissão indicada (Figuras 7(a)) e uma proporção de peso de 1 na primeira porta ativa, todo o tráfego de saída foi roteado para a porta `S1_0-eth4` no *switch* `S1_0`. Os resultados (Figura 7(b)) mostram a divisão equilibrada do tráfego UDP conforme o perfil 0 nos *switches* `S1_0`, `S2_2` e `S1_1`.

Teste com Perfil 100 (Figura 8): Usando a árvore de transmissão indicada (Figura 8(a)) e uma proporção de peso de 1:1:1 , o *switch* `S1_0` dividiu o tráfego de saída igualmente em suas três portas ativas (`S1_0-eth2`, `S1_0-eth4` e `S1_0-eth6`). A Figura 8(b) apresenta os resultados, ilustrando a distribuição de pacotes UDP seguindo o perfil 4 (1:1:1) nos *switches* `S1_0`, `S2_0`, `S2_2`, `S2_4` e `S1_1`.

Teste com o Perfil 1011 (Figura 9): Usando a árvore de transmissão indicada (Figura 9(a)) e uma proporção de peso de 2:1:2:1, o *switch* `S1_0` distribuiu o tráfego de forma variada em suas quatro portas ativas. Os resultados, representados na Figura 9(b), demonstraram a distribuição de pacotes UDP alinhada com o perfil 11 (2:1:2:1) nos *switches* `S1_0`, `S2_0`, `S2_2`, `S2_4` e `S1_1`.

4.2. Teste 2: Reconfiguração Ágil

O experimento consistiu na reconfiguração ágil entre os perfis de divisão de tráfego 0, 100 e 1011, já apresentados nas Figuras 7, 8 e 9. A divisão de tráfego foi determinada pela origem e, novamente, o *switch* $E1$ é responsável por inserir os rótulos *routeID* e *weightID* no cabeçalho dos pacotes. Foram capturados os tráfegos nas portas ativas dos switches $S2\_0$, $S2\_2$ e $S1\_1$. Importante ressaltar que, durante o experimento, os valores de *routeID* e *weightID* variaram conforme o perfil selecionado. Entretanto, por limitações de espaço e visando simplificar a descrição dos experimentos, os valores foram omitidos. O teste realizado consiste em três fases: uso do Perfil 0 por 10 s, em seguida, migração para o Perfil 100 por mais 10 s, e nova migração para o Perfil 1011 por mais 10 s.

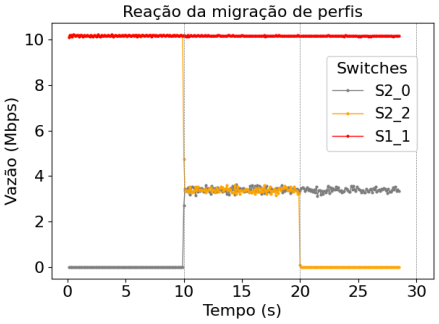


Figura 10. Teste 2: Resultado

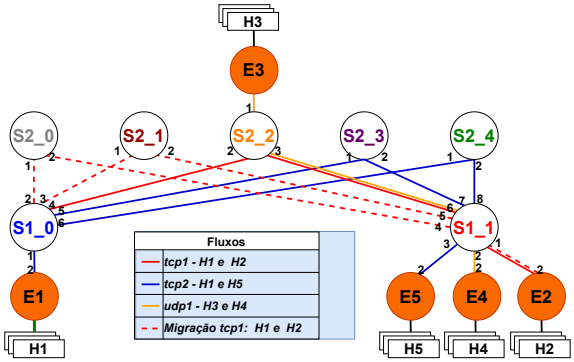


Figura 11. Teste 3: Cenário

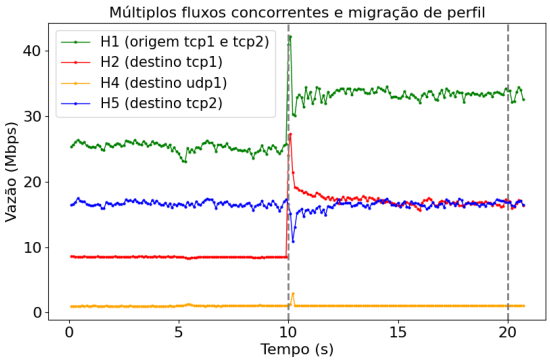


Figura 12. Teste 3: Resultado.

A vazão obtida nas portas ativas dos *switches* durante o experimento estão representadas na Figura 10, evidenciando um fluxo estável e constante durante as transições entre os perfis, sem flutuações ou interrupções. Dessa forma, a partir dos resultados da Figura 10, pode-se afirmar que, com o *MTS-PolKA*, a divisão de tráfego pode ser reconfigurada rapidamente apenas definindo novos rótulos na borda, sem necessidade de atualizar as tabelas dos nós de núcleo. Em abordagens baseadas em tabelas, há uma latência grande para configuração e convergência, já que o controlador precisa aplicar as alterações a todos os nós afetados ao longo do caminho [Jyothi et al. 2015].

4.3. Teste 3: Múltiplos fluxos e concorrência

Neste experimento, mostramos o funcionamento de múltiplos fluxos e a migração ágil de perfil de tráfego para otimizar o uso de recursos com concorrência. Foram enviados: um fluxo TCP entre *H1* e *H2* (*tcp1*, em vermelho), um fluxo TCP entre *H1* e *H5* (*tcp2*, em azul) e um fluxo UDP com vazão de 4Mbps entre *H3* e *H4* (*udp1*, em amarelo), conforme Figura 11. A divisão de tráfego foi determinada pelos *switches* de borda $E1$ e $E3$, que são responsáveis por inserir os rótulos *routeID* e *weightID* no cabeçalho dos pacotes. São usados dois perfis de divisão de tráfego: perfil 0 (envia todo tráfego para a primeira porta ativa) e perfil 1 (pesos 1:1, envia metade do tráfego para a primeira porta ativa e metade do tráfego para a segunda porta ativa).

No período de **0 s a 10 s**, o fluxo *tcp1* seguiu um único caminho ($S1\_0 - S2\_2 - S1\_1$) com perfil de tráfego 0, o fluxo *tcp2* usou o perfil de tráfego 1, dividindo o tráfego igualmente entre dois caminhos ($S1\_0 - S2\_3 - S1\_1$ e $S1\_0 - S2\_4 - S1\_1$), e o fluxo *udp1* seguiu um único caminho ($S2\_2 - S1\_1$) com perfil de tráfego 0. Como o fluxo *tcp1* inicia o experimento concorrendo com o fluxo *udp1* no enlace $S2\_2 - S1\_1$ e existem caminhos com enlaces ociosos, o experimento executou uma migração de caminho e perfil de tráfego para o fluxo *tcp1*. No período de **10 s a 20 s**, o fluxo *tcp1* foi migrado, apenas alterando a regra de fluxo que insere os rótulos dos pacotes referentes ao fluxo *tcp1* em $E1$, e passou a usar o perfil de tráfego 1 com a nova árvore de transmissão, dividindo o tráfego igualmente entre os dois caminhos que estavam ociosos ($S1\_0 - S2\_0 - S1\_1$ e $S1\_0 - S2\_1 - S1\_1$).

As vazões obtidas durante o experimento estão representadas na Figura 12 e mostram o aumento da vazão atingida pelo fluxo *tcp1* (linha vermelha) após a migração eliminar a concorrência deste fluxo com o fluxo *udp1* e passar a explorar dois caminhos ao invés de um único caminho. Além disso, é possível perceber o aumento da vazão agregada no host H1 (linha verde), que gera o tráfego, devido ao aumento da banda total disponível com a exploração de multicaminhos. Dessa forma, pode-se afirmar que, com o *MTS-PolKA*, a divisão de tráfego pode ser facilmente reconfigurada definindo novos rótulos na borda, com o funcionamento de múltiplos fluxos. Ainda, é possível explorar essa funcionalidade de migração ágil de perfil de tráfego para agregação de múltiplos fluxos TCP com o objetivo de otimizar o uso dos múltiplos caminhos com concorrência.

5. Conclusões e Trabalhos Futuros

Neste artigo foi introduzida uma abordagem inovadora denominada *MTS-PolKA*, que permite a seleção, na origem, de um perfil de divisão de tráfego baseado em proporção de pesos em uma árvore de distribuição de tráfego. Diferentemente das abordagens tradicionais, como *ECMP* e *WCMP*, que enfrentam limitações consideráveis na adaptação a flutuações de tráfego, o *MTS-PolKA* consegue alterar a distribuição de tráfego em todos os *switches* do caminho, de forma ágil, por meio de uma pequena modificação no cabeçalho do pacote, ocorrida na borda, sem alterações no núcleo da rede. Como resultado, foi observada uma rápida reconfiguração do tráfego para otimizar o uso dos recursos de rede em cenários experimentais com o emulador *Mininet* e *switches* programáveis em P4. Trabalhos futuros incluem extensão da avaliação de desempenho e escalabilidade da proposta em comparação com os trabalhos relacionados em diferentes cenários. Em especial, pretendemos analisar a sobrecarga de memória e banda do método, além da implementação em *switches* programáveis Tofino ®.

Agradecimentos

Este trabalho possui financiamento parcial de: Ifes, Fapes (#941/ 2022, #2023/ RWXSZ, #2022/ ZQX6), Fapesp/ MCTI/ CGI.br (#2020/ 05182-3) e CAPES (processo 2021-2S6CD, nº FAPES 132/2021) por meio do PDPG (Programa de Desenvolvi-mento da P os-Graduação, Parcerias Estratégicas nos Estados).

Referências

Cui, Z., Hu, Y., and Hou, S. (2021). Adaptive weighted cost multipath routing on pisa. In *2021 IEEE International Conference on Artificial Intelligence and Industrial Design (AIID)*, pages 541–544.

Dixit, A., Prakash, et al. (2013). On the impact of packet spraying in data center networks. In *2013 Proceedings IEEE INFOCOM*, pages 2130–2138.

Dominicini, C. et al. (2020). Polka: Polynomial key-based architecture for source routing in network fabrics. In *2020 6th IEEE Conference on Network Softwarization (NetSoft)*, pages 326–334.

Guimarães, R. S. et al. (2022). M-polka: Multipath polynomial key-based source routing for reliable communications. *IEEE Transactions on Network and Service Management*, 19(3):2639–2651.

Hsu, K.-F., Tammana, et al. (2020). Adaptive weighted traffic splitting in programmable data planes. In *Proceedings of the Symposium on SDN Research*, SOSR '20, page 103–109, New York, NY, USA. Association for Computing Machinery.

Huang, J., Lyu, et al. (2021). Mitigating packet reordering for random packet spraying in data center networks. *IEEE/ACM Transactions on Networking*, 29(3):1183–1196.

Jyothi, S. A., Dong, M., and Godfrey, P. B. (2015). Towards a flexible data center fabric with source routing. In *Proceedings of the 1st ACM SIGCOMM Symposium on Software Defined Networking Research*, pages 1–8.

Pang, J., Xu, G., and Fu, X. (2017). Sdn-based data center networking with collaboration of multipath tcp and segment routing. *IEEE Access*, 5:9764–9773.

Robin, D. D. and Khan, J. I. (2022). Clb: Coarse-grained precision traffic-aware weighted cost multipath load balancing on pisa. *IEEE Transactions on Network and Service Management*, 19(2):784–803.

Rottenstreich, O., Kanizo, et al. (2018). Accurate traffic splitting on commodity switches. In *Proceedings of the 30th on Symposium on Parallelism in Algorithms and Architectures*, SPAA '18, page 311–320, New York, NY, USA. Association for Computing Machinery.

Valentim, R. et al. (2019). Rdna balance: Balanceamento de carga por isolamento de fluxos elefante em data centers com roteamento na origem. In *Anais do XXXVII Simpósio Brasileiro de Redes de Computadores e Sistemas Distribuídos*, pages 1000–1013, Porto Alegre, RS, Brasil. SBC.

Zhou, J., Tewari, et al. (2014). Wcmp: Weighted cost multipathing for improved fairness in data centers. In *Proceedings of the Ninth European Conference on Computer Systems*, EuroSys '14, New York, NY, USA. Association for Computing Machinery.