UNIVERSIDADE FEDERAL DO RIO GRANDE DO SUL
INSTITUTO DE INFORMÁTICA
CURSO DE CIÊNCIA DA COMPUTAÇÃO

LUCAS NODARI

Combinando controle central e mecanismos locais para melhorar balanceamento de carga em redes WAN

Monografia apresentada como requisito parcial para a obtenção do grau de Bacharel em Ciência da Computação

Orientador: Prof. M.e Rodrigo Ruas Oliveira
Co-orientador: Prof. Dr. Marinho Pilla Barcellos

Porto Alegre
2017

RESUMO

Técnicas tradicionais de balanceamento de carga são configuradas para um comportamento esperado, baseado em informações disponíveis. Se a demanda é previsível, rotas ótimas podem ser calculadas para minimizar a utilização da rede. Entretanto, variações abruptas no tráfego podem fazê-lo sair do comportamento esperado e causar congestionamento e perdas de pacotes. Recentemente, algumas abordagens buscam reagir dinamicamente a variações no tráfego utilizando controle logicamente centralizado ou logicamente distribuído. Porém as propostas atuais dependem de baixa latência ou características específicas. Por isso, sua aplicabilidade torna-se limitada em redes com grande amplitude geográfica, como as WANs. Neste trabalho, propomos uma estratégia híbrida, parte proativa, parte reativa. A estratégia alia pré-configuração via controlador e reação local, com o objetivo de otimizar o comportamento global e minimizar a latência de reação. Comparamos a proposta com o método tradicional ECMP e encaminhamento sem balanceamento. As avaliações demonstram resultados promissores, com uma redução de até 40% nos tempos médios de término dos fluxos em relação ao ECMP.

Palavras-chave: Balanceamento de carga. WAN.

Combining centralized control and local mechanisms to improve load balancing in WAN

ABSTRACT

Traditional load balancing is configured for expected demands with the aid of available data. When the demand is predictable, one can obtain optimal routes that minimize network utilization. However, unpredictable traffic shifts may lead to unexpected behavior, which, in turn, can cause congestion and packet loss. Recent work attempts to react dynamically to traffic shifts by using either logically centralized or logically distributed control. Unfortunately, current proposals depend on low latency or specific conditions. Therefore, they do not scale in large, geographically dispersed networks, such as WANs. In this work, we propose a hybrid approach which is part proactive and part reactive. It combines a central controller-based preconfiguration and device-based local reaction to optimize network-wide behavior and minimize reaction time. We compare our proposal to the traditional ECMP and routing without load balancing. Evaluations show promising results, with a reduction of up to 40% in average flow completion time when compared to ECMP.

Keywords: Load Balancing, WAN.

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

DAG	Directed Acyclic Graph
ECMP	Equal Cost Multi Path
FCT	Flow Completion Time
LB	Load Balancing
No LB	No Load Balancing (referindo-se ao método sem balanceamento)
RTT	Round-Trip Time
TC	Traffic Control
WAN	Wide Area Network

SUMÁRIO

1 INTRODUÇÃO

Tradicionalmente, técnicas de balanceamento de carga em redes são preparadas para um comportamento esperado, definido a partir de previsões baseadas em matrizes de tráfego (FELDMANN et al., 2001; AWDUCHE et al., 2002; ROUGHAN; THORUP; ZHANG, 2003; ZHANG et al., 2005). Teoricamente, se a demanda é previsível, rotas ótimas podem ser calculadas para organizar fluxos de forma a minimizar a utilização da rede. Entretanto, apesar da demanda ser estável na maior parte do tempo, existem períodos em que ela varia rapidamente, de forma imprevisível, por durações que impedem algoritmos de balanceamento reajustar (UHLIG; BONAVENTURE, 2002; KODIALAM; LAKSHMAN; SENGUPTA, 2004; RUI; MCKEOWN, ; XU; ZHANG; BHATTACHARYYA, 2005). Análises realizadas por Chen et al. (2016) em coletas de matrizes de tráfego da rede Abilene (ZHANG, 2004) demonstram que a demanda pode contrastar com o comportamento esperado em até uma ordem de magnitude.

Quando picos de tráfego geram congestionamento, roteadores e enlaces sobrecarregados acarretam perdas de pacotes e prejudicam a eficiência e confiabilidade da rede (KANDULA et al., 2005). Uma forma de mitigar esse problema é configurar o roteamento para o pior caso tratável (APPLEGATE; BRESLAU; COHEN, 2004; APPLEGATE; COHEN, 2006; KODIALAM; LAKSHMAN; SENGUPTA, 2004; AZAR et al., 2003; LI; HARMS; HOLTE, 2005), ou um caso intermediário configurável entre o esperado e o pior (WANG et al., 2006), para evitar congestionamento em situações diversas. Em contrapartida, configurações baseadas em casos extremos podem levar ao uso ineficiente de recursos quando o tráfego tiver comportamento estável (CHEN et al., 2016).

Ao invés de procurar configurações pré-definidas ideais, trabalhos recentes buscam reagir dinamicamente a variações no tráfego. A reconfiguração do roteamento pode ser feita de forma logicamente centralizada (por meio de um controlador) (AL-FARES et al., 2010; KOPONEN et al., 2010; CURTIS et al., 2011; BENSON et al., 2011; HONG et al., 2013; RASLEY et al., 2014) ou logicamente distribuída (ALIZADEH et al., 2014; KATTA et al., 2016; WANG et al., 2016). Devido à latência presente no contexto de WANs, métodos logicamente centralizados podem somente ser utilizados quando é possível controlar as fontes de tráfego (YEGANEH; GANJALI, 2012; SCHMID; SUOMELA, 2013; YEGANEH; TOOTOONCHIAN; GANJALI, 2013; CHEN et al., 2016; KATTA et al., 2016). Métodos logicamente distribuídos são limitados em relação à escalabilidade pois necessitam manter estado sobre todos os caminhos possíveis para cada destino.

Este trabalho propõe investigar o uso de uma estratégia híbrida para reagir dinamicamente a variações no tráfego, aliando pré-configuração e reação. Nossa estratégia é baseada em uma pré-configuração usando controladores e uma reação usando somente o estado local dos dispositivos. Integrando ambos, busca-se oferecer o melhor comportamento global ao mesmo tempo em que minimiza-se a latência de reação. Enquanto o controle global é utilizado para pré-configurar a rede uma única vez, dispositivos mantém métricas locais de utilização das portas e selecionam a melhor opção para cada novo fluxo. Com isso evita-se tráfego de controle e, por sua vez, elimina-se a latência de reação para fluxos conhecidos e pré-configurados.

Implementamos um protótipo para comparar a proposta com o método tradicional de balanceamento de carga ECMP (Equal-Cost Multi-Path) e encaminhamento sem nenhuma forma de balanceamento. Para avaliação do esquema de balanceamento proposto quanto aos seus benefícios e limitações, estudamos o desempenho em diferentes cenários com variação de carga.

Avaliações preliminares indicam resultados promissores. Foi possível observar um ganho de até 40% na redução dos tempos médios de término dos fluxos em relação ao ECMP para casos onde a demanda difere do caso esperado. Identificamos também que a proposta se comportou no mínimo igual ao ECMP, indicando que sua sobrecarga de processamento de pacotes é desprezível. Para casos específicos, onde a demanda é alta mas não há necessidade de balanceamento, identificamos uma sobrecarga de processamento de pacotes em relação ao método sem balanceamento. Porém, tal comportamento condiz com análises anteriores (LIU et al., 2014).

O restante deste trabalho está organizado conforme segue. O Capítulo 2 discute o estado da arte em balanceamento de carga em redes WAN. O Capítulo 3 apresenta os fundamentos utilizados para realização da proposta. O Capítulo 4 descreve detalhadamente a proposta, apresenta as premissas e os aspectos principais. O Capitulo 5 apresenta o protótipo que foi desenvolvido, a metodologia de avaliação e os resultados. O Capítulo 6 encerra o documento com as principais conclusões e possibilidades de trabalhos futuros.

2 ESTADO DA ARTE

Existe uma ampla quantidade de propostas na área de balanceamento de carga em redes. Neste trabalho selecionamos uma amostra representativa e as classificamos em três categorias principais: métodos proativos (Seção 2.1), reativos "puros" (Seção 2.2) e reativos híbridos (Seção 2.3). Ressaltamos que nossa proposta se enquadra na terceira categoria.

Nossa categorização diferencia os trabalhos com base na sua dependência em relação a pré-configurações e a capacidade de se adaptar a determinados estados da rede. Métodos proativos pré-configuram o roteamento para um único comportamento e mantém esta configuração para quaisquer condições de tráfego. Métodos reativos alteram o roteamento para tratar variações no tráfego por meio de uma visão global do estado da rede. Um método híbrido pré-configura o roteamento para tratar um conjunto de comportamentos e alterna reativamente entre estas configurações baseando-se no estado local.

2.1 Métodos Proativos

Métodos proativos realizam uma configuração pré-definida baseada em informações sobre a rede – alguns casos usam exclusivamente a topologia, enquanto outros possuem matrizes de tráfego. O principal objetivo é definir uma configuração que sustente casos extremos ou comportamentos médios. As configurações frequentemente permanecem por longos períodos de tempo, pois alterar o roteamento pode interromper serviços devido ao tempo de convergência para um estado estável (APPLEGATE; COHEN, 2006).

Métodos sem conhecimento da demanda. A classe de técnicas denominada *Oblivious routing* (APPLEGATE; BRESLAU; COHEN, 2004; APPLEGATE; COHEN, 2006; KODIALAM; LAKSHMAN; SENGUPTA, 2004; AZAR et al., 2003; LI; HARMS; HOLTE, 2005) busca criar um roteamento baseando-se somente na topologia, sem considerar a demanda. A topologia é utilizada para criar uma configuração de roteamento que otimiza o desempenho no pior caso estimado. A principal motivação para o uso desse tipo de abordagem é a possibilidade de configurar a rede quando estimativas do comportamento do tráfego não estão disponíveis. Porém, quando o tráfego é estável, otimizar para o pior caso pode ter um alto custo (WANG et al., 2006).

Métodos com conhecimento da demanda. Tais métodos otimizam o roteamento para uma demanda representativa do histórico de tráfego (AGARWAL; NUCCI; BHAT-

TACHARYYA, 2005; ZHANG et al., 2005). A demanda esperada é dada por um comportamento médio estimado a partir de matrizes de tráfego obtidas em uma janela de tempo. As demandas são utilizadas como entrada para um problema de otimização, cuja solução é um roteamento ótimo de acordo com alguma métrica específica para cada técnica (FORTZ; THORUP, 2000; SRIDHARAN; GUÉRIN; DIOT, 2005; XU; CHIANG; REXFORD, 2011). COPE (WANG et al., 2006) combina os métodos de configuração de rotas aplicados em *oblivious routing* com a otimização baseada na demanda. Seu objetivo é utilizar previsões de tráfego para limitar a perda no pior caso possível até uma porcentagem aceitável. Os métodos nesta categoria só podem ser usados se a demanda for estável, pois variações de tráfego podem ter intensidade alta e acarretar congestionamento em pontos não mapeados, impactando o desempenho da rede (CHEN et al., 2016).

2.2 Métodos Reativos Puros

A maior limitação de métodos proativos é a falta de capacidade para manipulação adaptativa do tráfego de acordo com variações dinâmicas. Métodos reativos puros tentam resolver esse problema reagindo dinamicamente a variações, baseando-se em uma visão global do estado da rede.

Existem dois principais tipos de abordagens reativas puras: centralizada, onde o controlador é o único responsável por definir o balanceamento da carga; e distribuída, onde os dispositivos de encaminhamento interagem e definem o balanceamento sem participação do controlador. A seguir discute-se cada um dos tipos.

Reação centralizada. Nas técnicas de balanceamento reativo centralizado geralmente um controlador atua em intervalos periódicos coletando o estado da rede para distribuir o tráfego adequadamente (AL-FARES et al., 2010; CURTIS et al., 2011; JAIN et al., 2013; HONG et al., 2013; BENSON et al., 2011). O controlador realiza um ciclo que obtém informações de utilização e decide os caminhos que os fluxos irão utilizar. Ele pode deslocar fluxos de caminhos sobrecarregados para caminhos sub-utilizados ou somente alocar fluxos novos para os caminhos sub-utilizados.

Esses métodos possuem um controle mais preciso sobre o tráfego e podem melhorar a utilização de recursos, evitando sobre-provisionamento. Entretanto, eles possuem algumas restrições fortes. Para que as ações do controlador estejam coerentes com o estado da rede, uma entre duas premissas precisam estar presentes: ou a latência é baixa ou existe controle sobre as fontes de tráfego. A primeira premissa é realista em tráfego

intra-datacenter e é empregada pela maior parte dos trabalhos da literatura (AL-FARES et al., 2010; CURTIS et al., 2011; BENSON et al., 2011). A segunda premissa é realista em tráfego de WANs inter-datacenter e é adotada para escalonar fluxos de baixa prioridade, não orientados a interações com usuários (JAIN et al., 2013; HONG et al., 2013; KANDULA et al., 2014; KUMAR et al., 2015).

Reação distribuída. Os problemas do controle centralizado inspiraram a criação de métodos que atuam diretamente no plano de dados (ELWALID et al., 2001; KANDULA et al., 2005; MICHAEL; TANG, 2015; ALIZADEH et al., 2014; KATTA et al., 2016; WANG et al., 2016). Algumas abordagens utilizam medições fim-a-fim nas bordas da rede, enquanto outras atualizam as medições em cada dispositivo intermediário, propagando informações de utilização de forma similar ao roteamento por vetor de distância.

Esses métodos necessitam manter em cada dispositivo informações sobre todos os caminhos possíveis até um destino. Por causa disso, eles exploram um compromisso entre quantidade de informações mantidas e qualidade da reação. Para serem escaláveis, os métodos que exploram uma maior quantidade de caminhos são projetados para topologias de datacenter (ALIZADEH et al., 2014; KATTA et al., 2016). Essa escolha permite reduzir a quantidade de estado por caminhos possíveis explorando características topológicas específicas, porém ela limita a aplicabilidade das propostas.

2.3 Métodos Reativos Híbridos

A classe anterior é ineficiente quando a latência é alta e possui uma sobrecarga de controle que limita sua escalabilidade. Métodos reativos híbridos tem como objetivo definir uma configuração de roteamento que não necessita ser alterada e reagir dinamicamente a variações no comportamento do tráfego. Eles geralmente utilizam uma visão global para criar um conjunto de configurações e estado local para decidir entre estas configurações.

Há dois trabalhos principais que se enquadram nessa definição. (MATNI; TANG; DOYLE, 2015) compara arquiteturas de controle completamente centralizadas, completamente distribuídas e uma forma híbrida que combina ambas. Eles avaliam como diferentes arquiteturas se comportam ao decidir quais fluxos devem ser aceitos e quais devem ser rejeitados em cada ponto da rede, de forma a manter a utilização em um nível estável, pré-definido. Os resultados obtidos motivam o estudo de arquiteturas híbridas e distribuí-

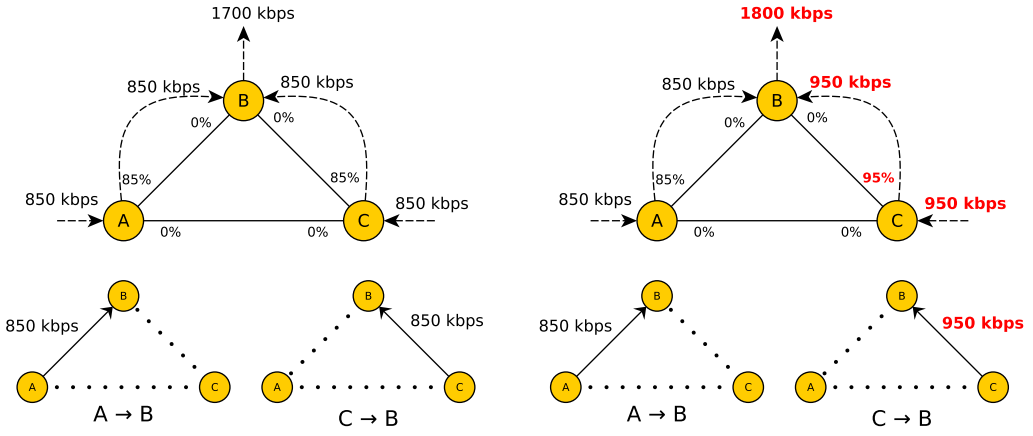
das em redes de alta latência, porém são limitados porque o modelo de fluxos considerado é irrealista e não há uma proposta de método de balanceamento.

DUCE (CHEN et al., 2016) é a alternativa mais próxima deste trabalho. Ele utiliza um controlador para criar duas configurações de roteamento, uma para comportamento esperado e outra para o pior caso possível de tratar. As configurações de roteamento são instaladas nos dispositivos, os quais decidem qual utilizar baseando-se, para isso, em estado local.

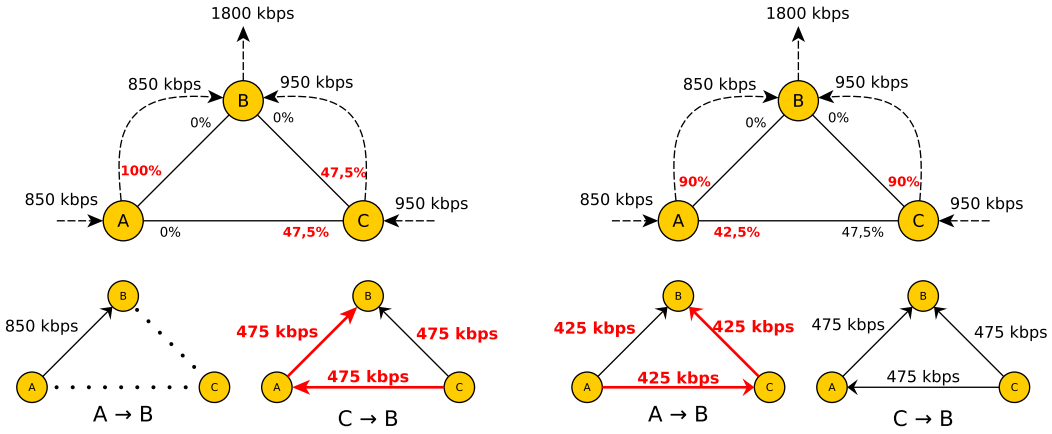
A estratégia proposta se diferencia de DUCE por três razões principais. Primeiro, enquanto DUCE utiliza duas configurações pré-definidas de roteamento, a estratégia proposta pode utilizar quaisquer caminhos disponíveis e que não estejam saturados. Segundo, DUCE altera entre uma configuração de caso médio para uma de pior caso quando detecta fluxos com uma taxa acima de um limiar. Tal escolha pode resultar em uso ineficiente de recursos e mudanças bruscas no roteamento. Por sua vez, a estratégia proposta aloca fluxos excedentes em caminhos alternativos sob demanda, acarretando mudanças graduais nas rotas. Terceiro, a forma de dividir a carga para balanceamento é diferente nas duas. DUCE divide ao nível de pacotes, enquanto que a proposta irá dividir ao nível de fluxos. Balanceamento por pacotes possibilita uma granularidade maior, porém prejudica o desempenho do TCP por causa de reordenamento. Balanceamento por fluxos evita reordenamento colocando pacotes de um mesmo fluxo no mesmo caminho.

A Figura 2.1 exemplifica a estratégia de alternar entre um caso médio e o pior caso. É demonstrada uma rede simples com três roteadores A, B e C, conectados através de enlaces com capacidade de 1 Mbps, as fontes de tráfego são abstraídas. O caso esperado é uma demanda de até 1 Mbps entre cada dispositivo, o pior caso tratável é uma demanda de 2 Mbps entre algum roteador até outro. As demandas de A e C para B inciam em 850 kbps (Figura 2.1(a)), quando a demanda de C aumenta para 950 kbps (Figura 2.1(b)), C alterna para a configuração do pior caso, colocando metade da demanda em cada caminho até B (Figura 2.1(c)). A fica congestionado desnecessariamente e troca o modo de encaminhamento (Figura 2.1(d)). A quantidade de tráfego em cada caminho até B se estabiliza em 900 kbps. Isto não sobrecarrega nenhum enlace, porém não é ideal pois metade de todo o tráfego atravessa um caminho com uma latência potencialmente maior e a utilização total da rede aumenta. Ocorre um aumento na utilização do enlace de A para C muito superior ao necessário sem diminuir a utilização dos enlaces de A para B e C para B.

Figura 2.1: Exemplo de funcionamento da técnica DUCE, alternando o roteamento de forma prejudicial, em um cenário com três roteadores (A, B e C) conectados com enlaces de 1 Mbps de capacidade.



(a) Demandas iniciais de 850kbps (b) Demanda de C para B aumenta para 950kbps



(c) Roteamento de C alterna para o pior caso tratável (d) Roteamento de A alterna para o pior caso tratável

Fonte: Os Autores

3 FUNDAMENTOS

Este capítulo descreve os principais conceitos que amparam o trabalho. Começamos na Seção 3.1 com a descrição de um esquema de definição de caminhos que permite a definição de múltiplas rotas para cada fluxo. Após, discutimos na Seção 3.2 o conceito de *flowlets* e sua importância na melhoria da granularidade de divisão de fluxos para o balanceamento. Por fim, apresentamos na Seção 3.3 aspectos do *pipeline* de processamento de pacotes do OpenFlow, os quais serão importantes para o método de reação proposto.

3.1 Roteamento via grafos acíclicos dirigidos

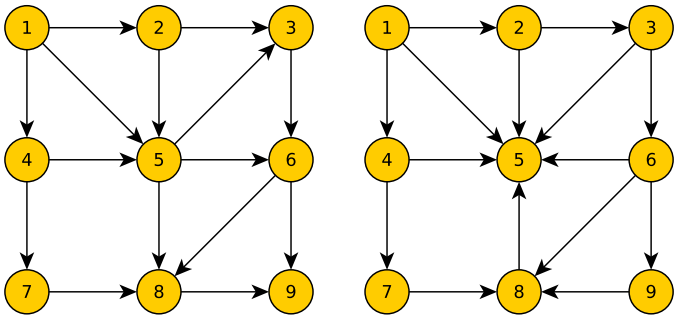
Roteamento por grafo acíclico dirigido (Directed Acyclic Graph – DAG) propõe que os algoritmos de roteamento usem um grafo acíclico como resultado de sua computação ao invés de uma árvore de caminhos (LIU et al., 2013; LIU, 2011; GAFNI; BERTSEKAS, 1981). Cada rota até determinado destino é expressa como um DAG com um único dreno. Nesse esquema, todos os enlaces podem fazer parte do roteamento pois cada um deles direciona o tráfego para o destino.

A característica fundamental do roteamento via DAGs é garantir, por definição, a não formação de ciclos. Além disto, assumindo que não ocorrem falhas, não irão ocorrer buracos negros, pois o dreno situa-se somente no destino. Essas características são importantes para a proposta porque permitem que qualquer escolha local de encaminhamento, dentre as opções possíveis, levem os pacotes ao destino de maneira correta.

A Figura 3.1 demonstra duas configurações de DAG para um grafo com 9 nós. Para cada possível destino na rede cria-se uma DAG diferente. As Figuras 3.1(a) e 3.1(b) exemplificam possíveis DAGs para quando os nós destino são 5 e 9, respectivamente. Nas ilustrações, as fontes de tráfego foram abstraídas. Em uma rede real, os caminhos são calculados entre os roteadores de borda da rede e, após, cada possível subrede em cada nó é mapeada para uma DAG. Por exemplo, suponha que um fluxo entra pela borda do roteador 4 e possui como destino uma máquina situada em uma subrede ligada ao roteador de borda 9. Nesse caso, o fluxo seria roteado usando a DAG ilustrada na Figura 3.1(a).

Uma das formas mais simples de gerar um DAG é utilizando um algoritmo de reversão de enlaces (GAFNI; BERTSEKAS, 1981). Em cada iteração, um nó que não é o destino e não possui nenhum enlace saindo, reverte a direção de seus enlaces. Esse algoritmo converge, garantidamente, em até n^2 reversões (BUSCH; SURAPANENI; TIRTHA-

Figura 3.1: Exemplos de DAG para roteamento. A origem é o nó 1.



(a) Destino é o nó 9. (b) Destino é o nó 5.

Fonte: Os Autores

PURA, 2003). Existem variações deste algoritmo que conseguem desempenho maior dependendo do cenário (CHARRON-BOST et al., 2015).

3.2 Balanceamento via *flowlets*

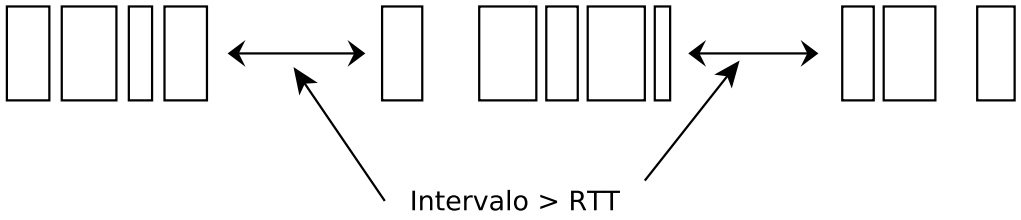
Divisão de carga por *flowlets* (KANDULA et al., 2007) propõe identificar fluxos como rajadas de pacotes com um limite de tempo. O objetivo dessa técnica é aumentar a granularidade de alocação de fluxos sem penalizar o desempenho com reordenamento de pacotes. Um *flowlet* é identificado por endereço IP fonte, endereço IP destino, protocolo de transporte, porta fonte e porta destino. Esses fluxos iniciam quando um dispositivo recebe o primeiro pacote e terminam se não aparecer mais nenhum pacote, após um tempo parametrizável. Geralmente, o tempo escolhido é o maior RTT da rede.

A Figura 3.2 exibe um exemplo de *flowlet*. Na figura, cada retângulo representa uma rajada de pacotes e espaços entre retângulos representa um intervalo ocioso entre rajadas. Suponha que um mesmo fluxo, utilizando uma mesma conexão TCP, envia rajadas de pacotes e que o tempo de duração para identificação de um *flowlet* é definido pelo maior RTT da rede (maior latência entre bordas). Nesse caso, o mesmo fluxo poderia ser alocado para três caminhos distintos sem que houvesse impacto no reordenamento de pacotes.

3.3 Aspectos do processamento de pacotes do OpenFlow

A reação local proposta neste trabalho é baseada em funcionalidades do processamento de pacotes do OpenFlow. Há três aspectos essenciais a serem discutidos: encadea-

Figura 3.2: Exemplo de *flowlet*. Retângulos são rajadas de pacotes enquanto espaços em branco são intervalos entre rajadas. Intervalos maiores do que RTT delimitam o *flowlet*.



Fonte: Adaptado de (KANDULA et al., 2007)

mento de tabelas, metadados e medidores.

Encadeamento de tabelas. O pipeline de um dispositivo OpenFlow (Figura 3.3) contém uma ou mais tabelas, cada uma contendo uma ou mais regras de fluxo (Open Networking Foundation, 2014). O processamento de um pacote sempre inicia na primeira tabela, independente do seu tipo.

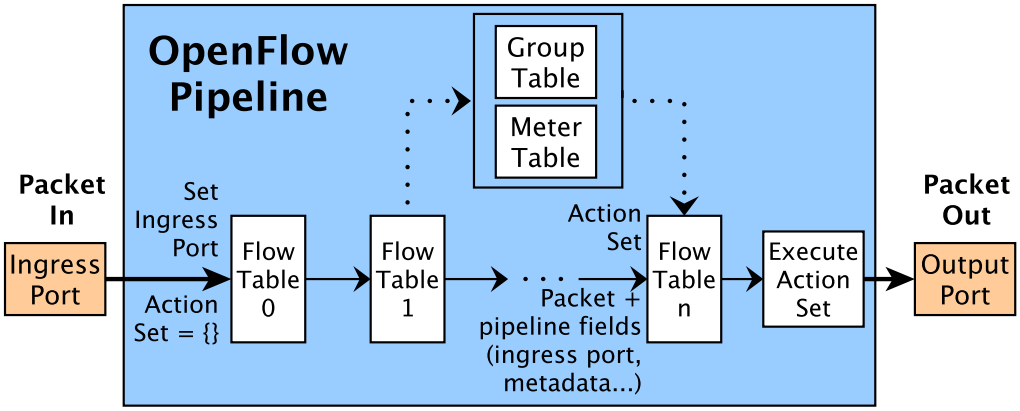
Quando um cabeçalho de pacote é processado por uma tabela ele é comparado com uma série de regras. Se uma regra é encontrada, o dispositivo executa um conjunto de instruções associados a mesma. Essas instruções podem encaminhar o pacote para uma porta de saída ou outra tabela, modificar atributos do pacote, ou aplicar um medidor. Se a instrução encaminha o pacote para uma outra tabela, o processo é repetido. É importante destacar que, para evitar ciclos no pipeline, uma tabela só pode encaminhar um pacote para uma outra com um número de identificação lexicograficamente superior.

Uma instrução pode direcionar um pacote para a tabela de medidores. Dependendo da configuração e estado do medidor, o pacote pode ser descartado, ou retornado ao processamento normal.

Metadados no processamento de pacotes. Os metadados são um conjunto de informações adicionais sobre um pacote codificados em um campo de 32bits. Eles podem ser utilizados para passar informações entre tabelas através do pipeline com o objetivo de influenciar escolhas subsequentes. Quando um pacote é apresentado para uma tabela, os campos para comparação consistem do pacote, a porta de entrada e os metadados. Caso os metadados não tenham sido inicializados, eles são ignorados. Caso contrário eles podem conter um valor arbitrário (ou um conjunto de valores, codificados no campo de 32bits). Eles possuem contexto diverso definido pela aplicação de controle.

A capacidade para modificação do campo de metadados é limitada. As instruções de uma regra aplicada a um pacote podem somente escrever um valor neste campo. No

Figura 3.3: Processamento no pipeline de um dispositivo OpenFlow.



Fonte: Adaptado de (Open Networking Foundation, 2014)

OpenFlow, modificar o campo de metadados condicionalmente só é possível utilizando regras que comparam com um valor pré-definido, para colocar outro valor pré-definido. Não é possível, atribuir valores variáveis aos metadados (por exemplo, incrementar ou copiar um valor do cabeçalho).

Medidores em regras. A tabela de medidores contém um conjunto de contadores, cada um com um conjunto de faixas de medição (Open Networking Foundation, 2014). Um medidor guarda a taxa de bytes ou pacotes processados nele e permite controlar a taxa destes bytes e pacotes.

Medidores são associados diretamente a regras. Qualquer regra pode especificar um medidor em sua lista de ações, cada medidor, por sua vez, conta a taxa agregada de todas regras para qual está associado. Múltiplos medidores podem ser aplicados a um pacote, usando em tabelas sucessivas ou em múltiplas ações de aplicar medidor.

Medidores contém faixas que são aplicadas quando a taxa medida atinge um valor mínimo. Pacotes são processados por uma única faixa, dependente da taxa medida no momento. Cada faixa possui um tipo que indica como os pacotes são processados. Os tipos possíveis são: *drop* e *dscp remark*. Se um pacote é colocado em uma faixa do primeiro tipo, o pacote é descartado. Caso o tipo seja o segundo, o atributo *dscp* do pacote é modificado para algum valor específico e o pacote retorna para o pipeline de tabelas.

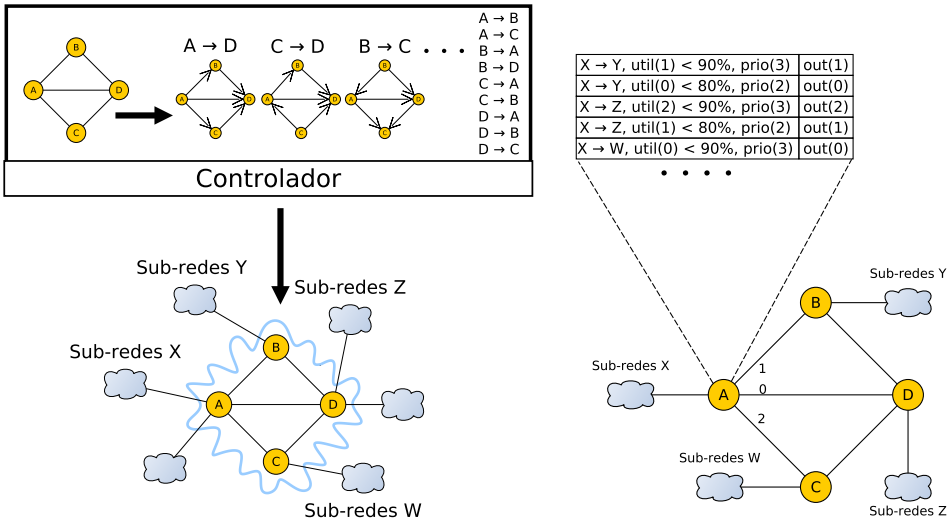
4 PROPOSTA

Este capítulo discute a arquitetura híbrida proposta. Na Seção 4.1 apresentamos uma visão conceitual da arquitetura. A seguir, detalhamos aspectos relacionados ao plano de controle (Seção 4.2) e à reação local (Seção 4.3).

4.1 Visão Geral

A arquitetura proposta combina controle global e reação local com o objetivo de prover caminhos globalmente ótimos e minimizar o tempo de reação a variações no volume de tráfego. A Figura 4.1, explicada a seguir, exemplifica o funcionamento da arquitetura.

Figura 4.1: Visão geral.



(a) Controlador utiliza uma representação abstraída da rede para encontrar os melhores caminhos entre cada prefixo.

(b) Dispositivos escolhem entre os caminhos utilizando somente informações de estado local.

Fonte: Os Autores

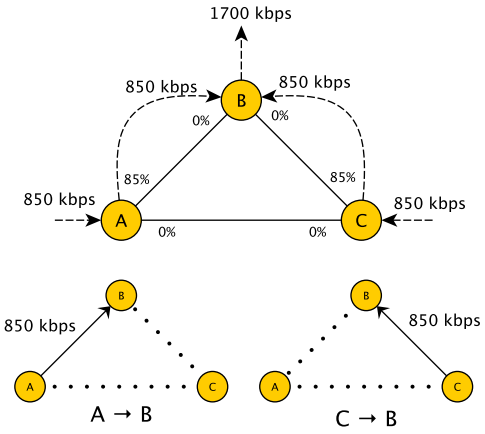
O controlador possui dois papéis principais: calcular as possíveis rotas de cada fluxo e transcreve-las em regras nas tabelas dos dispositivos. Inicialmente (Figura 4.1(a)), em uma etapa *offline*, o controlador calcula as rotas com base na topologia. Para viabilizar o balanceamento, quando possível, ele calcula múltiplos caminhos alternativos para cada fluxo. A seguir ele instala configurações nos dispositivos que indicam quais são as portas de saída para cada fluxo e em que ordem de prioridade elas devem estar.

No presente trabalho, o controlador não interage com os dispositivos após a configuração inicial. Isso ocorre porque assumimos que não ocorrem mudanças na estrutura

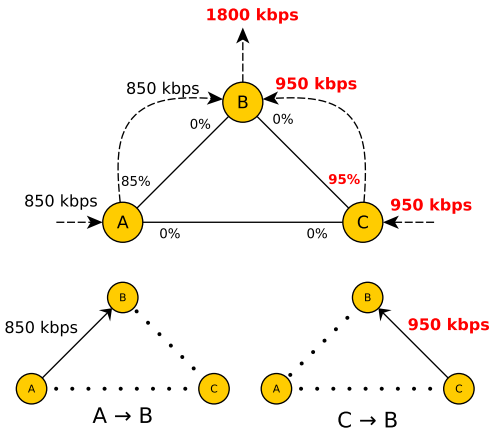
física da rede ou no conjunto de pares origem-destino de cada demanda (conforme descrito ao final desta seção, nas premissas).

Os dispositivos selecionam a porta para colocar cada fluxo através de um algoritmo de seleção local baseado somente na utilização de cada porta e em uma dada ordem de prioridade (Figura 4.1(b)). Após a escolha, os dispositivos criam uma regra mais específica para manter o fluxo na mesma porta. As regras novas recebem um limite de tempo, permanecendo ativas somente na duração de um *flowlet*. Como observado no exemplo da Figura 4.1(b), cada escolha possui uma taxa atribuída a qual deve estar abaixo de um limiar para que a regra possa ser escolhida. Caso todas estejam acima do limiar, o fluxo é alocado para uma porta padrão, pré-definida pelo controlador.

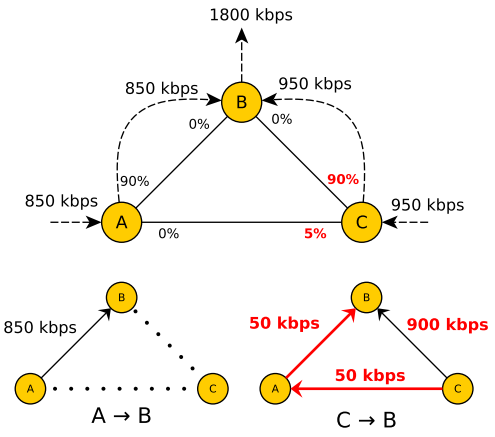
Figura 4.2: Exemplo do comportamento esperado da estratégia proposta, em um cenário com três roteadores (A, B e C) conectados com enlaces de 1 Mbps de capacidade.



(a) Demandas iniciais de 850kbps



(b) Demanda de C para B aumenta para 950kbps. *Sem a estratégia*, o tráfego segue somente pelo menor caminho.



(c) Demanda de C para B aumenta para 950kbps. *Com a estratégia*, o tráfego que excede o limiar é desviado para o caminho alternativo.

Fonte: Os Autores

A Figura 4.2 exemplifica o comportamento esperado do método proposto. Cada uma das 3 figuras parciais exibe 3 versões do mesmo grafo. No grafo maior, as setas tracejadas representam demandas e porcentagens representam a utilização das portas de saída. Os grafos menores representam os caminhos utilizados para cada par enviar o tráfego entre cada par origem-destino de roteadores de borda. Os valores indicam a quantidade de tráfego sendo direcionada para cada enlace. Nas Figuras 4.2(a) e 4.2(b), as opções primárias (linhas cheias) são os enlaces que conectam diretamente com o destino e as secundárias (linhas pontilhadas) a única outra opção. As trocas entre portas de saída (e, por sua vez, entre caminho) são definidas para começarem a ocorrer quando a utilização atinge 90%.

Na Figura 4.2(a), existem duas demandas inciais de 850 kbps. Como nenhuma supera o limiar, ambas são encaminhadas somente por meio do caminho primário. A seguir, na Figura 4.2(b), a demanda de C para B aumenta para 950 kbps. Isto faz a utilização superar o limiar de 90%. Por causa disto, C começa a colocar novos fluxos no caminho secundário. Na Figura 4.2(c), a divisão do tráfego se estabiliza, quando 900 kbps seguem pelo caminho primário e 50 kbps pelo caminho secundário. Esta mudança do caminho de parte do tráfego ocorre na granularidade de *flowlets*, ou seja, é necessário que alguns fluxos acabem ou que haja um intervalo entre rajadas maior do que RTT.

O trabalho considera as seguintes três premissas:

P1 Não ocorrem alterações físicas nos dispositivos de encaminhamento ou enlaces. Logo, falhas e outras modificações topológicas estão fora do escopo da proposta. O principal objetivo é lidar com congestionamento causado por variações na demanda. Para lidar com modificações na topologia seria possível receber outra configuração do controlador (o que pode ser insatisfatório, em virtude da demanda) ou utilizar regras sobressalentes para reação rápida. Alternativas serão estudadas em trabalhos futuros, conforme discutido na conclusão.

P2 O controlador possui informação prévia sobre todos os custos e capacidades dos enlaces. Essa escolha busca simplificar a configuração dos caminhos e é uma prática já comumente adotada para realizar o roteamento.

P3 Todos os prefixos existentes na rede são conhecidos pelo controlador. Dessa forma, é possível mapear os caminhos calculados para prefixos destino e pré-configurar a rede. Na prática, os prefixos poderiam ser conhecidos implementando-se BGP diretamente no controlador como em (ROTHENBERG et al., 2012; SCHLINKER et al., 2017).

4.2 Controle Global

Esta seção descreve os métodos utilizados para calcular caminhos e transcrevê-los em regras. Para cada um deles são apresentadas decisões de projeto e implementação.

Definição dos caminhos. Para maximizar o número de caminhos e garantir que as escolhas locais não acarretem laços, o controlador define um DAG para cada possível destino. Existem muitos métodos possíveis para selecionar caminhos na rede, cada qual possui vantagens e desvantagens de acordo com as métricas e requisitos desejados (FORTZ; THORUP, 2000; VALLET; BRUN, 2014). Uma escolha típica é minimizar a quantidade de recursos utilizados, a qual pode ser obtida priorizando os menores caminhos (LIU et al., 2014).

A configuração do roteamento busca criar um DAG que prioriza os caminhos mínimos como heurística para reduzir a utilização de recursos. Isto é similar ao problema de encontrar os K-Menores-Caminhos em um grafo, quando K é um valor suficiente para que todas arestas sejam visitadas uma vez. Por causa disto, aplica-se uma adaptação do algoritmo de Dijkstra que encerra somente quando todas as arestas tiverem sido incluídas na lista de caminhos. O algoritmo pode receber como entrada um grafo com os custos de cada enlace, priorizando escolhas de acordo com a topologia.

Regras de encaminhamento. O controlador coloca nos dispositivos um conjunto de regras que possibilitam escolher o melhor caminho no momento em que pacotes de novos fluxos são processados. Em alto nível, a lógica pode ser abstraída conforme o Algoritmo 1. O algoritmo recebe como entrada o pacote, o limiar para escolha de porta e a porta padrão. Ele inicia quando chega um pacote no dispositivo. Enquanto o cabeçalho do pacote casa com uma ou mais regras não verificadas, seleciona-se a mais prioritária. Se a utilização da porta associada à regra estiver abaixo do limiar, é instalada uma regra para o fluxo do pacote na porta escolhida. Essa regra nova mantém o fluxo na mesma porta pela duração de um *flowlet*, evitando reordenamento de pacotes. Caso nenhuma porta tiver utilização abaixo do limiar, é escolhida uma porta padrão.

Para implementar essa lógica em OpenFlow, é necessário desenrolar o laço em várias tabelas encadeadas e usar indicadores para sinalizar quando a utilização de determinada porta supera o limiar. A implementação feita considera os metadados como um único valor que indica qual das n possíveis opções de porta será utilizada para encaminhar o pacote. O valor do campo metadados varia entre 0 e $n-1$, onde 0 significa a primeira opção de porta e $n-1$ a última.

Algoritmo 1: Algoritmo de escolha de portas

```
1: in: P     // Pacote
2: in: L     // Limiar para escolha de porta
3: in: d     // Porta padrão
4:
5: for R in match(P.hdr) do     // Para cada regra (em ordem de prioridade)
6:    p ← port(R);
7:    if usage(p) < L then
8:       install_rule(P.hdr, p);
9:       return;
10:   end if
11: end for
12: install_rule(P.hdr, p);
```

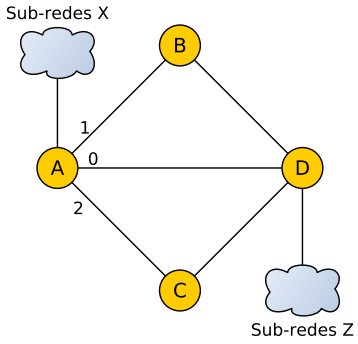
As regras são encadeadas conforme a seguinte lógica. Uma primeira regra inicializa os metadados relacionados ao pacote em 0 e encaminha para a próxima tabela. A tabela seguinte "testa" a primeira opção de porta. Testar a porta significa verificar a sua taxa de utilização e tomar uma entre duas ações: (1) caso a taxa seja abaixo de um limiar, os metadados conservam o valor atual; (2) caso contrário, incrementa-se o valor nos metadados. Após o teste, o processamento do pacote é encaminhado para a tabela seguinte, a qual possui duas opções de acordo com as ações tomadas na ação Testar. Caso os metadados tenham conservado o valor atual, encaminha-se na porta testada. Caso contrário, testa-se a próxima porta. Após a escolha da porta, uma regra especializada é criada para o fluxo. A seguir explica-se o funcionamento dessas etapas por meio de um exemplo. A implementação das ações de testar e especializar regras serão discutidas na próxima seção.

A Figura 4.3 e a Tabela 4.1 exemplificam o processo de escolha de portas. Supondo que um fluxo chega no dispositivo A e possui como destino uma sub-rede conectada a D. O pacote que inicia o fluxo começa sendo processado na tabela 1. Inicialmente, os metadados do pacote são inicializados em 0 e o cabeçalho é encaminhado para a tabela 2. Na tabela 2, o pacote é testado na porta 0; caso a utilização esteja acima do limiar, os metadados do pacote são incrementados e o processamento segue para a tabela 3, independentemente do resultado. Na tabela 3, caso os metadados não tenham sido incrementados, o fluxo é colocado na porta 0; caso contrário o fluxo é testado na porta 1 e o processamento segue para a próxima tabela. Na tabela 4, o processo é similar ao da tabela anterior, porém a porta de saída será a 1 e a próxima porta testada será a 2, dependendo do resultado na tabela anterior. Na última tabela, o fluxo é colocado na porta 2 se o valor dos metadados permaneceu igual no último teste, caso contrário, é colocado em uma porta padrão.

Conforme é possível perceber, há um custo adicional no tamanho das tabelas.

Enquanto que um esquema de encaminhamento sem nenhuma forma de balanceamento utilizaria somente uma regra para cada destino, o método proposto necessita de duas regras para cada porta, isto para cada par fonte e destino. Como o presente trabalho foca-se em switches em software, no qual a quantidade de regras não é um problema, esse custo não será analisado. Alternativas para reduzir a quantidade de regras necessárias serão analisadas em trabalhos futuros, conforme discutido na conclusão.

Figura 4.3: Exemplo de regras para o dispositivo A, para um fluxo que inicia em A e termina em D (em sub-redes conectadas a esses dispositivos). Existem três opções de saída: o enlace entre A e D (primeira opção), o enlace entre A e B (segunda) e o enlace entre A e C (terceira).



Fonte: Os Autores

Tabela 4.1: Regras de encaminhamento do dispositivo A.

Tabela de fluxo	*Regra*	*Ação*
1	*	Inicializa metadados em 0 Vai para tabela 2
2	Fonte=X Destino=Z	Testa porta 0 Vai para tabela 3
3	Fonte=X Destino=Z Metadados=0	Encaminha na porta 0
	Fonte=X Destino=Z Metadados=1	Testa porta 1 Vai para tabela 4
4	Fonte=X Destino=Z Metadados=1	Encaminha na porta 1
	Fonte=X Destino=Z Metadados=2	Testa porta 2 Vai para tabela 5
5	Fonte=X Destino=Z Metadados=2	Encaminha na porta 2
	Fonte=X Destino=Z Metadados=3	Encaminha na porta padrão

Fonte: Os Autores

4.3 Reação Local

Conforme descrito, cada dispositivo de encaminhamento toma decisões locais quanto ao balanceamento seguindo as instruções definidas pelo controlador. Para isso, ele precisa realizar duas ações: teste da porta de saída e especialização de regras. Esta seção explica ambas.

Antes de discutir tais ações, vale ressaltar que elas são modificações simples no dispositivo de encaminhamento. É possível realizar o método proposto sem as modificações, porém seria necessário uma quantidade elevada de regras ou comunicação com o controlador, acarretando problemas de escala. Observe, entretanto, que essas modificações ocorrem apenas no firmware, ou seja, não é necessário hardware novo.

Ação de teste da porta de saída. O teste de porta é composto por uma ação de aplicar um medidor do OpenFlow, que incrementa condicionalmente os metadados dependendo da utilização da porta. Quando esta ação é executada, o medidor calcula a utilização. Se ela estiver acima do limiar, o valor nos metadados é incrementado.

A taxa de utilização de uma porta é estimada através da média móvel da taxa de envio, sendo atualizada a cada novo pacote processado. A média móvel considera a taxa medida no instante e a taxa anterior com um peso, obtido empiricamente, de 40% para a atual.

Todo tráfego entrante no mecanismo de escolha de caminhos é contado nos medidores. Porém, a especialização de regras faz com que somente os pacotes que iniciam fluxos passem pela ação de teste.

Ação de especialização de regras. Especialização de regras é uma ação que cria uma regra mais específica a partir de uma regra genérica, similarmente a Curtis et al. (2011). Nesse caso, o primeiro pacote de um fluxo que casa com uma regra genérica leva à criação de uma regra nova, mais específica, para esse fluxo. Neste trabalho, as regras genéricas contém apenas o prefixo destino, enquanto as regras específicas são formadas por IP fonte e destino, protocolo de transporte, porta fonte e destino.

As regras específicas são adicionadas por tempo limitado (tempo de um *flowlet*), para permitir maior granularidade nas escolhas de alocação de fluxos para caminhos. Isso é feito colocado-se um valor de *timeout* ocioso nas regras específicas.

A regra é colocada na primeira tabela, com o valor máximo de prioridade. Isso permite que o fluxo seja encaminhado através do mesmo caminho que foi escolhido durante o seu tempo de vida. A regra permanece enquanto chegarem pacotes do fluxo. Caso

não cheguem mais pacotes por um tempo superior ao *timeout*, ela será removida automaticamente. Após remoção, se chegar um novo pacote do fluxo, ele será processado novamente na lógica de escolha. Dependendo do resultado, o fluxo pode ser colocado na mesma porta ou em outra.

5 AVALIAÇÃO

Este capítulo demonstra a metodologia de avaliação. A Seção 5.1 apresenta detalhes do ambiente de experimentação. A Seção 5.2 descreve os métodos que foram comparados com a proposta. A Seção 5.3 discute os experimentos realizados e resultados obtidos com os mesmos.

5.1 Ambiente de experimentação

Implementamos um protótipo utilizando as seguintes ferramentas para emulação, controle e teste de redes: Mininet, Ofsoftswitch13, Ryu, Iperf, TC e Trickle. Por limitações de escala, o método desenvolvido foi testado em uma topologia simples, com 4 dispositivos e um hospedeiro conectado em cada.

Controlador. Utilizamos o Ryu (Ryu SDN Framework Community, 2015) para efetuar a pré-configuração do encaminhamento. O algoritmo para geração de DAG é implementado como uma aplicação no Ryu, que recebe como entrada um grafo com os custos dos enlaces. O controlador possui uma conexão *out-of-band* com todos os dispositivos. Após os dispositivos conectarem, a aplicação calcula os caminhos e configura cada um deles.

Dispositivo de encaminhamento. Usamos o switch em software Ofsoftswitch13 (FERNANDES, 2012). Esse switch foi o escolhido por ser a melhor alternativa atual com suporte maduro a medidores (*meter tables*). O suporte a medidores no OpenVSwitch ainda está em desenvolvimento e foi previsto para a versão 2.8[1], porém sua versão atual é a 2.7[2].

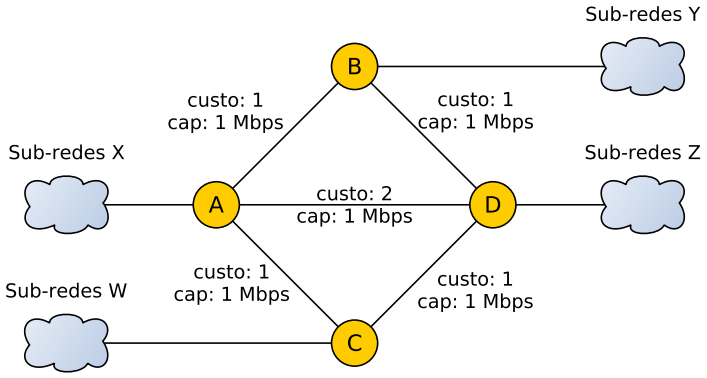
Topologia. Utilizamos o Mininet (LANTZ; HELLER; MCKEOWN, 2010) para criar o ambiente de testes. Todos os testes foram realizados em uma topologia simples (Figura 5.1) com 4 dispositivos de encaminhamento. Os dispositivos representam roteadores de borda, cada um conectado a uma subrede. As subredes são representadas por um único hospedeiro conectado diretamente com um dos dispositivos. O hospedeiro atua como fonte de toda demanda de tráfego do prefixo ou para o prefixo.

Foram atribuídos custos para os enlaces de tal forma que existam 3 caminhos de mesmo custo entre A e D (Figura 5.1). O primeiro caminho é saindo de A direto para D,

[1] <http://docs.openvswitch.org/en/latest/faq/qos/>
[2] <https://github.com/openvswitch/ovs/releases>

o segundo passa por B e o terceiro passa por C.

Figura 5.1: Topologia de testes. Os valores em cada aresta indicam o custo e a capacidade de cada enlace.



Fonte: Os Autores

Geração de tráfego. Utilizamos Iperf (TIRUMALA et al., 2005) e Trickle (ERIKSEN, 2005) para controlar a carga de trabalho. O primeiro foi utilizado para gerar diversas micro demandas de fluxo de um hospedeiro em uma subrede para outro hospedeiro em outra. O segundo serviu para controlar a taxa de envio de cada demanda. Ambos foram combinados para emular a presença de diversos hospedeiros em cada subrede.

A emulação dos hospedeiros ocorreu da seguinte forma. Um único hospedeiro representa diversos clientes enviando tráfego a partir do prefixo de origem até um servidor que está no prefixo de destino. O tráfego gerado é composto por 120 fluxos com tamanho igual, iniciados com um intervalo (Δ) de 250 milésimos entre cada e limitados a uma taxa máxima (Θ) de 80 kbps. O tempo de execução de um fluxo é definido por um tamanho parametrizável atribuído a cada fluxo. Com isso, uma demanda especifica é obtida por meio da quantidade de fluxos em execução ao mesmo tempo. Essa quantidade e a demanda estão em função do tamanho de cada fluxo. A seguir formalizamos o cálculo da quantidade e da demanda em função do tamanho. A Tabela 5.1 descreve o significado de cada variável e constante usados na formalização.

Quantidade de fluxos (Equação 5.1): a demanda (D) é dada pela taxa fixa (Θ) multiplicada pelo número de fluxos ativos (n). Para descobrir a quantidade exata de fluxos, dividimos a taxa fixa (Θ) pela demanda (D).

$$D = \Theta * n \Leftrightarrow n = D/\Theta \tag{5.1}$$

Como a quantidade de fluxos é um número inteiro, pegamos o valor natural mais próximo abaixo do valor obtido. Assim, o número de fluxos ativos (n) é obtido com uma

pequena variação da equação anterior: $n = \lfloor D/\Theta \rfloor$.

Dada a quantidade necessária de fluxos ativos para gerar a demanda, é preciso descobrir qual a duração necessária para cada fluxo de forma que gere-se, na média, n fluxos ativos.

Duração dos fluxos (Equação 5.2): A duração de cada fluxo (t) é dada pelo seu tamanho (T) dividido pela taxa de envio Θ. Porém o tamanho (T) é uma incógnita. Outra forma de calcular a duração é em função do intervalo de inicio dos fluxos (Δ), que é constante para os experimentos. Se um fluxo é gerado a cada (Δ) segundos e deve haver no máximo n fluxos, o tempo de duração dos fluxos é dado por $t = \Delta * n$. (Exemplo, se um fluxo é criado a cada segundo, para que hajam 20 fluxos concorrentes é necessário que cada um dure 20s.)

$$t = \Delta * n \tag{5.2}$$

Sabendo a duração necessária de cada fluxo, ainda é preciso calcular qual o tamanho que irá acarretar essa duração.

Tamanho dos fluxos (Equação 5.3): De forma simples, o tamanho (T) é a duração (t) multiplicado pela taxa de envio (Θ).

$$T = \Theta * t \tag{5.3}$$

Os três cálculos anteriores são suficientes para obter o tamanho de cada fluxo de forma a gerar a demanda necessária. Porém, eles são equivalentes a uma simplificação dada pela substituição das Equações 5.1 e 5.2 na Equação 5.3, conforme segue.

Tamanho em função da demanda (Equação 5.4): Substituindo a variável duração (t) pela Equação 5.2 na Equação 5.3 temos que tamanho (T) é a multiplicação das constantes Θ e Δ pela quantidade de fluxos n: $T = \Theta * \Delta * n$. Substituindo a variável quantidade de fluxos (n) pela Equação 5.1 (com o devido arredondamento), temos:

$$T = \Theta * \Delta * \lfloor D/\Theta \rfloor \tag{5.4}$$

Por exemplo, para obter uma demanda média de 1 Mbps utilizando uma taxa máxima de 80 kbps é necessário manter aproximadamente 12 fluxos ativos (portanto uma demanda aproximada de 80 kbps$*$12 = 960 kbps). Como um fluxo novo inicia a cada 250 milésimos, é preciso iniciar fluxos com duração de 3 segundos. Para um fluxo durar 3 segundos a uma taxa de 80 kbps, o tamanho deve ser 30 kB (240 kb). Similarmente,

caso deseje-se obter uma taxa agregada de 2 Mbps são necessários 25 fluxos de 80 kbps cada. Nesse caso a duração média é 6 segundos, que é obtida com um tamanho de 60 kB (480 kb).

Tabela 5.1: Significado de cada variável e constante.

Constante	*Significado*	*Valor*
Θ	Taxa de cada fluxo	80 kbps
Δ	Intervalo entre fluxos	250 ms
Variável		*Unidade*
D	Demanda necessária	kbps
n	Quantidade de fluxos ativos ao mesmo tempo	
t	Duração de cada fluxo	s
T	Tamanho de cada fluxo	kB

Fonte: Os Autores

Capacidade e latência. O Mininet permite definir a capacidade e latência dos próprios enlaces usando a ferramenta do kernel do Linux TC (Traffic Control) (KUZNETSOV, 1999). Nos experimentos realizados, a capacidade foi definida para 1 Mbps e a latência variou entre 50 e 150 ms. A escolha de portas no método proposto foi configurada para começar a ser realizada quando a utilização dos enlaces atinge 90%. O valor foi escolhido através de experimentos preliminares, oferecendo uma margem de segurança para evitar perda de pacotes.

5.2 Métodos comparados

O protótipo foi comparado com duas alternativas: encaminhamento sem nenhuma forma de balanceamento e encaminhamento com balanceamento tradicional ECMP (Equal-Cost Multi-Path). Ambas utilizam o controlador para pré-configurar os caminhos da rede. No caso do encaminhamento sem balanceamento, o controlador seleciona os menores caminhos para cada par de origem e destino e codifica-os nos dispositivos. Após, cada fluxo é encaminhado diretamente usando a regra genérica definida na tabela (ou seja, não aplica-se a especialização de regras).

A implementação do ECMP é derivada do programa do controlador para o método proposto. O controlador utiliza as informações sobre a topologia e aplica um algoritmo de menor caminho adaptado para encontrar caminhos de custo igual. Nesse caso, os dispositivos são configurados em tabelas de grupos de seleção do OpenFlow, as quais permitem que seja escolhida uma entre diversas opções de portas de saída. Especificamente, quando um fluxo chega pela primeira vez, o switch envia-o para uma tabela de grupos configurada

para selecionar uma entre diversas portas em *round-robin*. A seguir, a ação de especialização de regras cria uma regra específica para o fluxo, de forma a mantê-lo na mesma porta pelo tempo de duração de um *flowlet*.

Por motivos de simplicidade, a proposta não será comparada com o trabalho DUCE. Para comparar com o DUCE, seria necessário modificar ele para tratar balanceamento via fluxos ao invés de pacotes. Além disso, DUCE utiliza matrizes de tráfego para criar a configuração do roteamento, isto depende da escalabilidade do ambiente e será tratado em trabalhos futuros.

5.3 Avaliações

As avaliações foram divididas em duas categorias principais, as quais consideram se o menor caminho é suficiente para comportar todas as demandas (sem congestionamento no menor caminho, Subseção 5.3.1) e ou se ele é insuficiente (com congestionamento no menor caminho, Subseção 5.3.2). Para cada categoria, são investigados dois cenários, totalizando 4 tipos de experimentos distintos.

Os cenários sem congestionamento analisam o efeito da disponibilidade de recursos. São verificadas duas situações: ampla disponibilidade de recursos e escassez de recursos. O segundo caso indica que o menor caminho comporta a demanda, mas com uma utilização alta no(s) seu(s) enlace(s).

Os cenários com congestionamento examinam a consequência de como é configurado o roteamento. Verificam-se dois casos: um com os custos dos enlaces otimizados para a demanda e outro com custos não-otimizados. As características do tráfego gerado para cada cenário estão descritas na Tabela 5.2.

Os experimentos tiveram 3 variações (uma para cada método) e foram repetidos 10 vezes. Em todos, foi medido o tempo médio de término de fluxos (*Flow Completion Time* – FCT). Os valores exibidos são as médias de todos os valores de FCT obtidos entre todas as execuções realizadas, com intervalo de confiança em 99%. Nos cenários com congestionamento foi realizada uma avaliação mais detalhada. Nos mesmos, além do FCT, foram analisados a quantidade de fluxos ativos e o atraso para o inicio dos fluxos. Todas as avaliações foram realizadas com carga de tráfego uniforme (os fluxos possuem a mesma demanda). Essa escolha tem por objetivo analisar o comportamento da variação da demanda isoladamente e impedir que o desempenho do ECMP seja afetado negativamente por fluxos de tamanhos variados.

Tabela 5.2: Configurações utilizadas para geração de tráfego em cada cenário.

Cenário	*Fonte*	*Destino*	*Demanda (kbps)*	*Tamanho dos fluxos (kB)*	*FCT esperado (s)*
Sem congestionamento no menor caminho					
Recursos amplos	A	D	900	28	2,8
	B	C	600	18	1,8
	B	D	300	9	0,9
Recursos escassos	A	D	1000	30	3
	B	C			
	B	D			
Com congestionamento no menor caminho					
Rotas otimizadas	A	D	3000	90	9
Rotas não otimizadas	A	B	2000	60	6
	C	D	1000	30	3

Fonte: Os Autores

5.3.1 Sem congestionamento no menor caminho

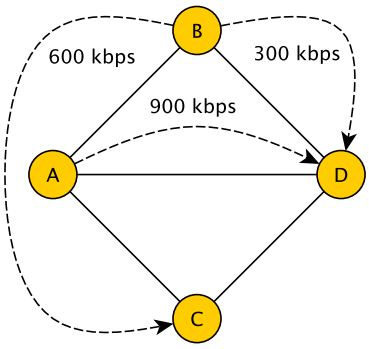
Esta subseção demonstra os dois cenários em que o menor caminho seria suficiente para lidar com as demandas. O primeiro analisa o comportamento em uma situação com ampla disponibilidade de recursos, ou seja, balanceamento de carga não deve causar congestionamento em nenhum caminho. O segundo é uma situação em que os recursos são escassos, pois a demanda é próxima do limite tratável pelo menor caminho.

Ampla disponibilidade de recursos. A Figura 5.2 exibe um caso onde as demandas entre cada par são suficientemente baixas para não congestionar nenhum caminho independente do método escolhido. As setas tracejadas representam as demandas.

A Figura 5.3 apresenta os resultados deste cenário. Os três métodos avaliados ficaram muito próximos em relação ao FCT, indicando que balanceamento de carga não possui nenhum ganho ou sobrecarga quando a disponibilidade de recursos é ampla. Como as demandas não superam as capacidades, o encaminhamento sem balanceamento deve possuir FCT médio próximo ao esperado. O método proposto comporta-se como o encaminhamento sem balanceamento. Ele não divide a carga pois nenhuma demanda atinge o limite de utilização para iniciar a escolher caminhos alternativos. Por sua vez, por causa da configuração da rede, o ECMP divide a carga de A para D em três caminhos, a carga de B para C em dois caminhos e a carga de B para D em somente um caminho. Devido a essas divisões, o enlace de B para D recebe carga originada das três fontes. Apesar disso, a utilização não entra em congestionamento, pois o total agregado é 900 kbps.

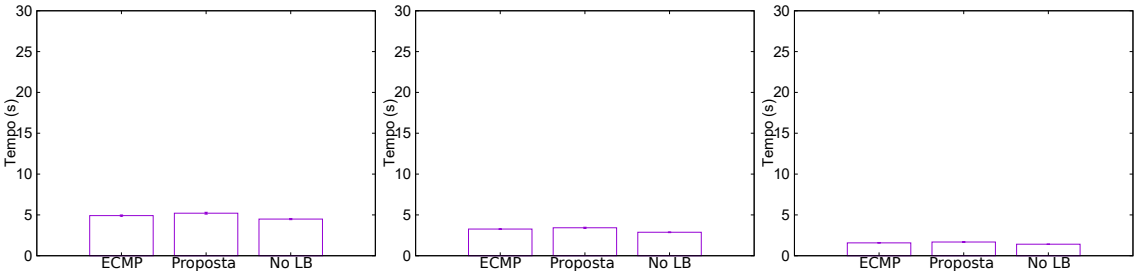
Os três métodos possuem FCT muito próximos aos valores esperados, sendo o

Figura 5.2: Efeito dos mecanismos de balanceamento de carga quando os recursos são amplos e as demandas não são suficientes para congestionar o menor caminho. As setas tracejadas representam as demandas entre cada par.



Fonte: Os Autores

Figura 5.3: FCT médios para o cenário sem congestionamento e recursos amplos. *Quanto menor o tempo, melhor.*



(a) FCT médio entre A e D (b) FCT médio entre B e C (c) FCT médio entre B e D

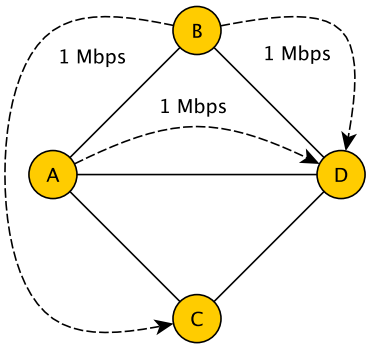
Fonte: Os Autores

encaminhamento sem balanceamento marginalmente melhor do que o resto. Acreditamos que tal se deve ao fato de ser um método simples, que não possui sobrecarga. O ECMP e o método proposto ficaram muito próximos para diferenciar, o que indica que, em uma situação deste tipo, faz pouca diferença o método de balanceamento aplicado.

Escassez de recursos. Neste cenário as demandas foram aumentadas até o limite de capacidade suportado pelo menor caminho, causando uma escassez de recursos (Figura 5.4). A configuração do cenário é tal que o caso ótimo ocorre somente se todas as demandas forem alocadas em caminhos únicos específicos. Qualquer desvio acarretará congestionamento. Em outras palavras, o desempenho do encaminhamento sem balanceamento é ótimo para esse caso. Idealmente, balanceamento de carga não deve distribuir carga de uma forma que prejudique o desempenho.

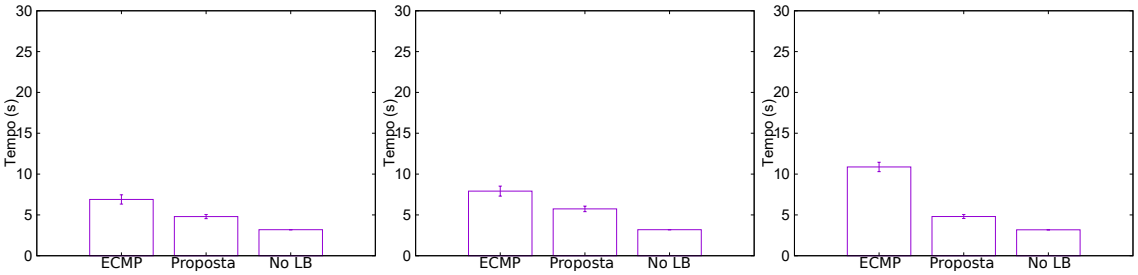
A Figura 5.5 demonstra o comportamento dos métodos. Conforme esperado, o melhor FCT médio foi no encaminhamento sem balanceamento. No caso das estratégias de balanceamento o desempenho degrada pois ambos distribuem o tráfego de maneira

Figura 5.4: Desempenho quando as demandas situam-se no limite, causando escassez de recursos, mas sem congestionar o menor caminho. As setas tracejadas representam as demandas entre cada par.



Fonte: Os Autores

Figura 5.5: FCT médios para o cenário sem congestionamento e recursos escassos. Nesse caso, *quanto menor o tempo, melhor*.



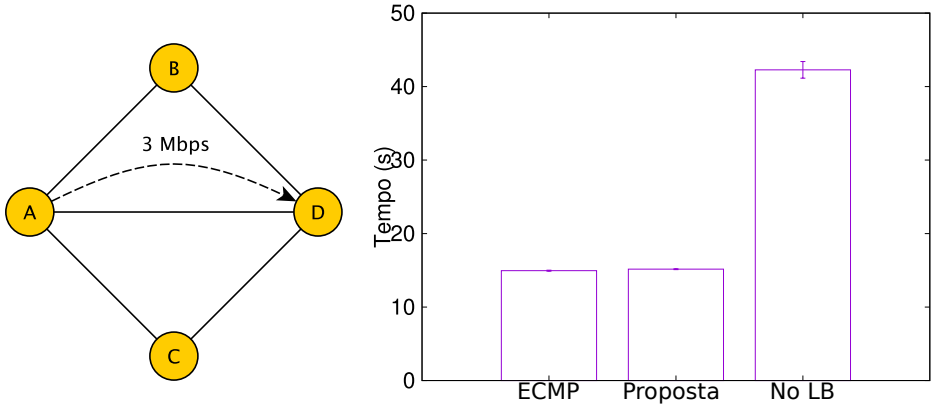
(a) FCT médio entre A e D (b) FCT médio entre B e C (c) FCT médio entre B e D

Fonte: Os Autores

inadequada. O pior foi no ECMP pela sua maneira de dividir o tráfego. Ele faz a mesma divisão do cenário anterior, porém agora a quantidade de tráfego agregada no enlace entre B e D atinge 1,8 Mbps, que é quase duas vezes a sua capacidade. Em consequência, ocorre congestionamento e o FCT médio aumenta. O método proposto situa-se entre os dois. A redução do FCT em relação ao ECMP foi de até 55%, porém o aumento em relação ao método sem balanceamento foi de no mínimo 79%. Isso indica que ele evita dividir o tráfego desnecessariamente, porém não atinge o desempenho ideal porque variações nas medições de utilização ocasionalmente levam a uma má escolha de caminhos alternativos.

Atribuímos o principal problema deste cenário à falta de visão global. Um controlador nesse caso poderia auxiliar o balanceamento ao perceber que uma má escolha está causando congestionamento desnecessariamente. Tal observação indica que há espaço a ser investigado no projeto de estratégias híbridas explorando um compromisso no aumento da interação com o controlador. Em particular, ao invés de utilizar somente uma pré-configuração, seria possível verificar quais os ganhos e perdas que ocorrem ao au-

Figura 5.6: Desempenho com uma configuração otimizada.



(a) Seta tracejada representa a demanda.

(b) FCT para cenário com congestionamento e com custos otimizados para demanda. *Quanto menor o tempo, melhor.*

Fonte: Os Autores

mentar gradualmente a frequência com que o controlador adquire estado global da rede e modifica as configurações de roteamento.

5.3.2 Com congestionamento no menor caminho

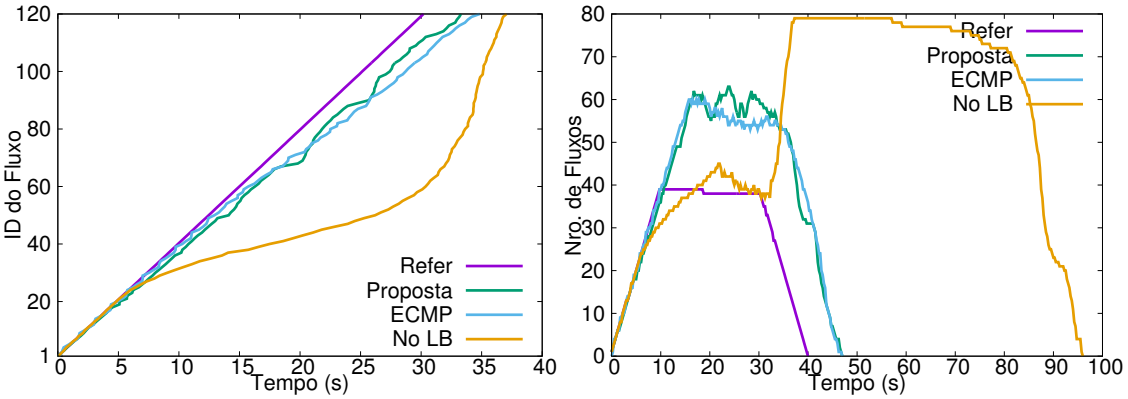
Esta subseção demonstra os cenários em que o menor caminho não é suficiente para tratar as demandas. Como observação geral destaca-se que, conforme esperado, o ganho das estratégias é muito superior ao encaminhamento sem balanceamento. Ademais, nesse caso há indícios de que a estratégia proposta pode apresentar ganhos significativos em relação ao ECMP.

Foram analisados dois cenários. No primeiro a configuração é otimizada para tratar a demanda, no segundo não. O objetivo do primeiro é demonstrar que a proposta não acarreta sobrecarga em relação ao caso ótimo do ECMP, enquanto o objetivo do segundo foi mostrar o potencial de ganho da proposta.

Custos otimizados para demanda. Neste cenário, conforme observado na Figura 5.6(a), a demanda está de acordo com o caso ótimo para o ECMP, quando considerando os custos dos enlaces definidos anteriormente (Figura 5.1). Em outras palavras, a divisão antevista para o ECMP consegue evitar completamente o congestionamento.

A Figura 5.6(b) apresenta os resultados deste cenário. No encaminhamento sem balanceamento o FCT situa-se próximo a 180% maior do que os métodos que distribuem a carga. Os métodos que utilizam balanceamento ficaram muito próximos. Isto é um indicativo de que a sobrecarga do método proposto é desprezível. Apesar do ganho signi-

Figura 5.7: Comportamento dos fluxos para cenário com congestionamento e custos otimizados para demanda. Nesse caso, a curva de referência indica o comportamento esperado caso não houvesse congestionamento. Em (a), a análise mostra o tempo para cada um dos fluxos começar a enviar dados. Em (b), a análise mostra os fluxos ativos (todos que concluíram a etapa de conexão e começaram a enviar dados).



(a) Tempo para fluxos iniciarem. *Quando mais próximo da referência, melhor.*

(b) Quantidade de fluxos ativos. *Quando menor a área, melhor.*

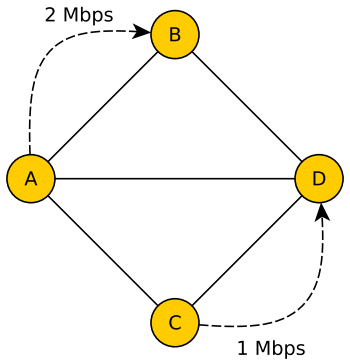
Fonte: Os Autores

ficativo em utilizar balanceamento de carga, os valores de FCT médios ficaram acima do esperado, indicando que houve possível atraso para os fluxos iniciarem e atingirem a taxa máxima.

Para estudar esse caso, verificamos o comportamento dos fluxos ao longo do tempo. A Figura 5.7 demonstra a dinâmica de inicialização e término de fluxos. Primeiro verificamos, na Figura 5.7(a) o momento em que o servidor recebe o primeiro pacote de dados (logo, após a conclusão da etapa de conexão). Nesse caso, incluímos uma curva de referência que indica qual seria o comportamento em um cenário sem congestionamento. No caso dos métodos com balanceamento, é possível verificar que há um pequeno atraso em relação à referência. No início o desvio é pequeno, porém seu efeito acumula causando uma penalização geral. O efeito da penalização pode ser observado no gráfico ao lado (Figura 5.7(b)). Nela, verificamos a quantidade de fluxos ativos ao longo do tempo. Conforme a quantidade de fluxos aumenta, inevitavelmente uma pequena quantidade de perdas ocorre gerando, alongando os demais fluxos. Isso faz com que mais fluxos fiquem ativos ao mesmo tempo, porém com uma taxa de envio reduzida (Figura 5.7(b)).

No caso do método sem balanceamento, o atraso para grande parte dos fluxos é visivelmente maior. É possível verificar uma tendência à divergência seguida por convergência no tempo de inicio de fluxos (Figura 5.7(a)) e uma queda visível na quantidade de fluxos ativos (Figura 5.7(b), entre 20s e 35s). Isso indica que uma grande quantidade de fluxos não consegue sequer iniciar a conexão (perde um pacote SYN ou SYN/ACK).

Figura 5.8: Desempenho com uma configuração não-otimizada.



Fonte: Os Autores

Com isso, os fluxos dobram o valor do *timeout* e tentam recomeçar a tentativa de conexão. É esperado que essa operação se repita enquanto houver fluxos com uma alta taxa de envio inibindo os fluxos novos. Quando os fluxos antigos começam a acabar, uma grande quantidade de fluxos encontra espaço para iniciar. Porém, dessa vez a taxa de envio de cada é reduzida, alongando o tempo de término.

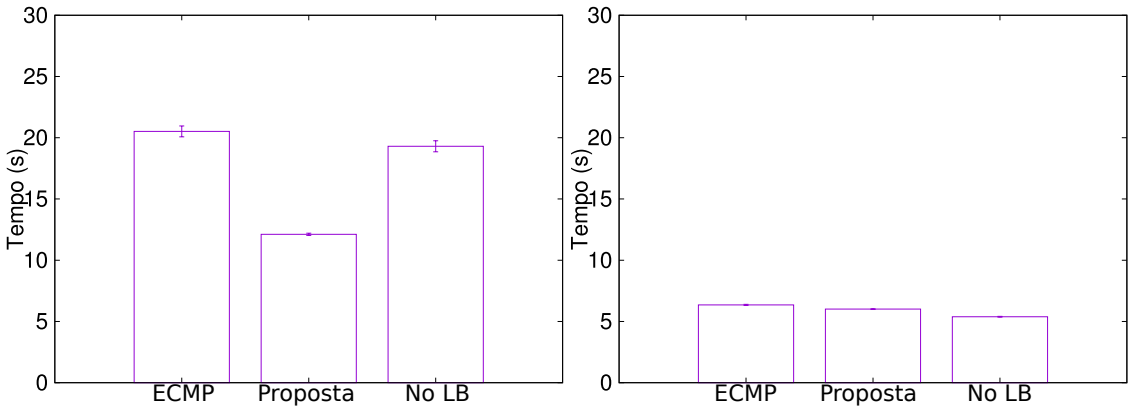
Vale destacar que a demanda é três vezes acima da capacidade do menor caminho, necessitando o uso dos três caminhos para ser tratada. Por causa disto, o encaminhamento sem balanceamento é consideravelmente afetado.

Custos não otimizados para demanda. Nesse caso comparamos duas demandas topologicamente isoladas ocorrendo ao mesmo tempo, conforme a Figura 5.8. Na demanda de A para B há congestionamento, enquanto na demanda de C para D não há. O ECMP e o método sem balanceamento enviam todo o tráfego através de somente um caminho (isso ocorre em virtude dos pesos pré-definidos, conforme Figura 5.1). Por causa disso, o enlace entre A e B é congestionado, pois recebe uma demanda que é o dobro de sua capacidade. A proposta não sofre este problema, pois trata o tráfego de A utilizando o caminho direto para B e o que atravessa D.

O objetivo deste cenário foi analisar a perda de desempenho do balanceamento de carga quando a demanda difere do comportamento estimado com as matrizes de tráfego (Figura 5.8). Isto significa que os custos dos enlaces estão definidos de uma forma que não distribui adequadamente o tráfego entre os caminhos possíveis.

Devido a isso, o FCT do ECMP foi tão alto quanto o encaminhamento sem balanceamento. O método proposto possui o menor FCT médio (40% menor na média, conforme a Figura 5.9). Isso acontece pois ele reage dinamicamente às variações de tráfego. Ao contrário dos demais, que dependem de configurações baseadas em conhecimento

Figura 5.9: FCT médios para o cenário com congestionamento e configuração não otimizada. *Quanto menor o tempo, melhor.*



(a) FCT médio entre A e B (b) FCT médio entre C e D

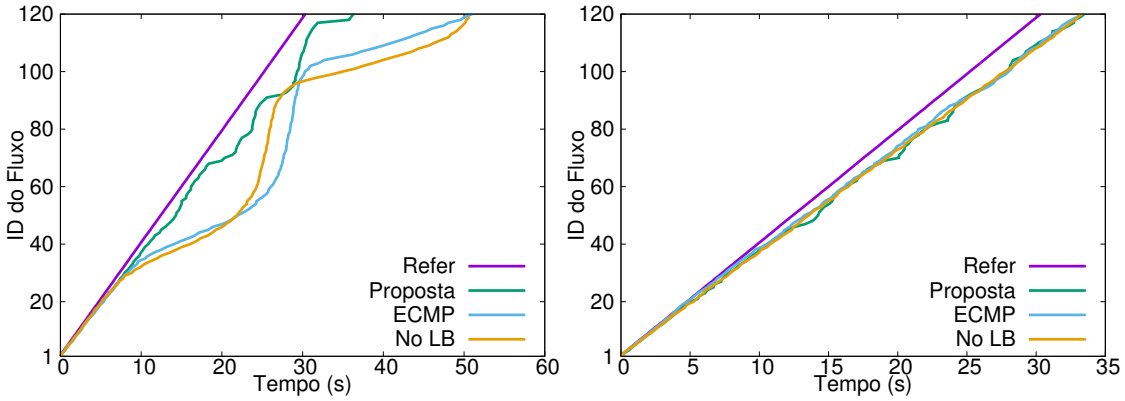
Fonte: Os Autores

prévio.

A Figura 5.10 compara as demandas com e sem congestionamento para este cenário (recorde que ambas são demandas que ocorrem paralelamente, mas não compartilham os mesmos enlaces). É possível observar que para as demandas com congestionamento (fluxos entre A e B), o comportamento da proposta é similar ao da referência. Por sua vez, o ECMP e o método sem balanceamento neste caso comportam-se de maneira similar ao observado no método sem balanceamento na Figura 5.7, quando este último estava congestionado. Além disso, perceba que para a demanda não congestionada (fluxos entre C e D), a demanda é similar à referência. Essas observações indicam que, neste caso, a proposta conseguiu lidar com o congestionamento e superar as demais. Esses resultados indicam que a proposta possui uma clara vantagem em relação às demais.

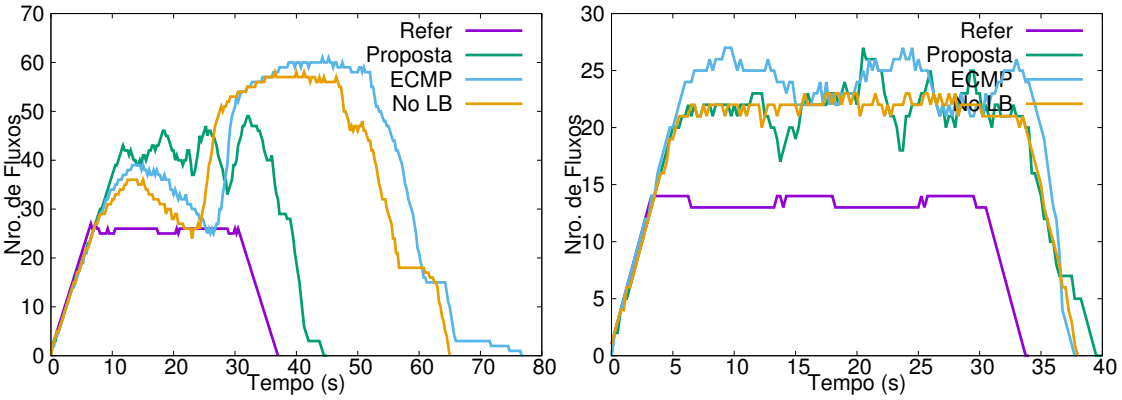
Os cenários apresentados nesta seção buscaram capturar um conjunto de comportamentos limite esperados em uma rede. Apesar de ainda serem bastante preliminares, os resultados indicam que a proposta pode beneficiar o balanceamento em casos onde a carga comporta-se de forma distinta ao esperado pelas matrizes de tráfego. A seguir discutimos como essa proposta pode ser expandida e suas as potencias direções futuras.

Figura 5.10: Comportamento dos fluxos para cenário com congestionamento e custos não otimizados para demanda. A curva de referência indica o comportamento esperado caso não houvesse congestionamento. Em (a) e (b), a análise mostra o tempo para cada um dos fluxos começar a enviar dados. Em (c) e (d), a análise mostra os fluxos ativos (todos que concluíram a etapa de conexão e começaram a enviar dados).



(a) Tempo para fluxos iniciarem entre A e B. *Quando mais próximo da referência, melhor.*

(b) Tempo para fluxos iniciarem entre C e D. *Quando mais próximo da referência, melhor.*



(c) Quantidade de fluxos ativos entre A e B. *Quando menor a área, melhor.*

(d) Quantidade de fluxos ativos entre C e D. *Quando menor a área, melhor.*

Fonte: Os Autores

6 CONCLUSÃO

Neste trabalho, investigamos o uso de uma arquitetura híbrida para melhorar o balanceamento de carga em redes WAN. A arquitetura proposta combina controle central e reação local com o objetivo de prover caminhos globalmente ótimos e minimizar o tempo de reação a variações no volume de tráfego. Nessa arquitetura, o controle global pré-configura a rede uma única vez e os dispositivos utilizam somente informações locais para escolher as portas para encaminhar fluxos novos.

A proposta possibilita obter uma latência de reação próxima a zero, pois a reação ocorre no tempo em que chegam fluxos nos dispositivos, dependendo somente da velocidade em que processam pacotes. A sobrecarga de controle é mínima, pois o controlador atua somente uma vez. Entretanto, devido aos dispositivos considerarem somente o estado local, eles não possuem nenhuma visibilidade para evitar escolhas que levam a congestionamento em enlaces que situam-se adiante no caminho. A proposta possui uma sobrecarga significativa no tamanho das tabelas, pois o mecanismo de encaminhamento utiliza um conjunto complexo de regras que aumenta conforme o numero de portas e destinos.

Comparamos o método proposto com o ECMP e encaminhamento sem balanceamento. As avaliações foram dividas em cenários com e sem congestionamento no menor caminho. As seguintes observações foram constatadas pelos resultados obtidos:

- **Cenários sem congestionamento.** Em situações com ampla disponibilidade de recursos, qualquer escolha é boa o suficiente e nenhuma possui ganho ou sobrecarga aparente. Quando a demanda cresce e os recursos tornam-se escassos, em casos onde não há congestionamento no menor caminho, os algoritmos de balanceamento podem causar sobrecarga em relação ao encaminhamento via menor caminho. Essa observação está de acordo com análises de outros autores (LIU et al., 2014). Identificamos que isso ocorre principalmente por causa de más escolhas nos custos dos enlaces (no caso do ECMP) ou nas decisões locais (no caso da abordagem proposta), possivelmente justificando uma interação maior com o controlador.
- **Cenários com congestionamento.** Quando o menor caminho é insuficiente para lidar com a demanda, as estratégias de balanceamento mostram ganho considerável. Nesses casos, o algoritmo proposto tem uma redução de até 40% nos tempos médios de término dos fluxos em relação ao ECMP quando a configuração não está de acordo com a demanda. Quando os custos estão otimizados para o cenário de con-

gestionamento específico, o desempenho é o mesmo, indicando que a sobrecarga de processamento de pacotes da proposta é desprezível.

Atualmente, estamos investigando formas de reduzir a quantidade de regras necessárias para implementar o mecanismo. Uma possibilidade é adaptar as tabelas de grupo do OpenFlow, pois permitem definir uma lista de portas e um algoritmo que escolhe uma porta entre essa lista. Um dos algoritmos padrão é o *fast failover*, que escolhe a primeira porta que atende a um critério, similar ao algoritmo de escolha neste trabalho. Uma vantagem disso é reduzir a quantidade de regras nas tabelas principais do pipeline, porém isto dependerá da escalabilidade das tabelas de grupo.

Além disso, estamos explorando alternativas para ampliar a escala do ambiente utilizado para experimentação. Estamos analisando variações do switch em software utilizado que permitem obter desempenho maior. Isso permitirá criar cenários baseados em topologias reais (KNIGHT et al., 2011), executar uma quantidade maior de demandas e utilizar tráfego real baseado em matrizes de tráfego (ZHANG, 2004; UHLIG et al., 2006). Isto viabilizaria realizar comparação com o DUCE e outros trabalhos relacionados.

Adicionalmente, iremos estudar novas escolhas de projeto que permitam uma interação maior com um controlador logicamente centralizado. A principal direção de pesquisa atual é desenvolver uma arquitetura que realiza reconfiguração periódica. Isso permitirá tratar os pontos que foram discutidos anteriormente nas premissas, como, por exemplo, falhas na rede. Utilizar reconfiguração periódica por meio do controlador possibilitaria também tratar situações de congestionamento que não são resolvíveis somente com a reação local.

REFERÊNCIAS

AGARWAL, S.; NUCCI, A.; BHATTACHARYYA, S. Measuring the shared fate of igp engineering and interdomain traffic. In: **13TH IEEE International Conference on Network Protocols (ICNP'05)**. [S.l.: s.n.], 2005. p. 10 pp.–.

AL-FARES, M. et al. Hedera: Dynamic flow scheduling for data center networks. In: **NSDI**. [S.l.: s.n.], 2010. v. 10, p. 19–19.

ALIZADEH, M. et al. Conga: Distributed congestion-aware load balancing for datacenters. In: **Proceedings of the 2014 ACM Conference on SIGCOMM**. New York, NY, USA: ACM, 2014. (SIGCOMM '14), p. 503–514. Available from Internet: <http://doi.acm.org/10.1145/2619239.2626316>.

APPLEGATE, D.; BRESLAU, L.; COHEN, E. Coping with network failures: Routing strategies for optimal demand oblivious restoration. In: **Proceedings of the Joint International Conference on Measurement and Modeling of Computer Systems**. New York, NY, USA: ACM, 2004. (SIGMETRICS '04/Performance '04), p. 270–281. Available from Internet: <http://doi.acm.org/10.1145/1005686.1005719>.

APPLEGATE, D.; COHEN, E. Making routing robust to changing traffic demands: algorithms and evaluation. **IEEE/ACM Transactions on Networking (TON)**, IEEE Press, v. 14, n. 6, p. 1193–1206, 2006.

AWDUCHE, D. et al. **Overview and principles of Internet traffic engineering**. [S.l.], 2002.

AZAR, Y. et al. Optimal oblivious routing in polynomial time. In: **Proceedings of the Thirty-fifth Annual ACM Symposium on Theory of Computing**. New York, NY, USA: ACM, 2003. (STOC '03), p. 383–388. Available from Internet: <http://doi.acm.org/10.1145/780542.780599>.

BENSON, T. et al. Microte: Fine grained traffic engineering for data centers. In: **Proceedings of the Seventh COnference on Emerging Networking EXperiments and Technologies**. New York, NY, USA: ACM, 2011. (CoNEXT '11), p. 8:1–8:12. Available from Internet: <http://doi.acm.org/10.1145/2079296.2079304>.

BUSCH, C.; SURAPANENI, S.; TIRTHAPURA, S. Analysis of link reversal routing algorithms for mobile ad hoc networks. In: . ACM, 2003. (SPAA). Available from Internet: <http://doi.acm.org/10.1145/777412.777446>.

CHARRON-BOST, B. et al. Time complexity of link reversal routing. **ACM Trans. Algorithms**, ACM, v. 11, n. 3, 2015. ISSN 1549-6325. Available from Internet: <http://doi.acm.org/10.1145/2644815>.

CHEN, F. et al. Engineering traffic uncertainty in the openflow data plane. In: **IEEE INFOCOM 2016 - The 35th Annual IEEE International Conference on Computer Communications**. [S.l.: s.n.], 2016. p. 1–9.

CURTIS, A. R. et al. Devoflow: Scaling flow management for high-performance networks. In: **Proceedings of the ACM SIGCOMM 2011 Conference**. New York, NY, USA: ACM, 2011. (SIGCOMM '11), p. 254–265. Available from Internet: <http://doi.acm.org/10.1145/2018436.2018466>.

ELWALID, A. et al. Mate: Mpls adaptive traffic engineering. In: **Proceedings IEEE INFOCOM 2001. Conference on Computer Communications. Twentieth Annual Joint Conference of the IEEE Computer and Communications Society (Cat. No.01CH37213)**. [S.l.: s.n.], 2001. v. 3, p. 1300–1309 vol.3. ISSN 0743-166X.

ERIKSEN, M. A. Trickle: A userland bandwidth shaper for unix-like systems. In: **USENIX Annual Technical Conference, FREENIX Track**. [S.l.: s.n.], 2005. p. 61–70.

FELDMANN, A. et al. Deriving traffic demands for operational ip networks: Methodology and experience. **IEEE/ACM Transactions on Networking (ToN)**, IEEE Press, v. 9, n. 3, p. 265–280, 2001.

FERNANDES, Z. L. K. E. L. **OpenFlow 1.3 Software Switch**. 2012. Available from Internet: <https://github.com/CPqD/ofsoftswitch13>.

FORTZ, B.; THORUP, M. Internet traffic engineering by optimizing ospf weights. In: **Proceedings IEEE INFOCOM 2000. Conference on Computer Communications. Nineteenth Annual Joint Conference of the IEEE Computer and Communications Societies (Cat. No.00CH37064)**. [S.l.: s.n.], 2000. v. 2, p. 519–528 vol.2.

GAFNI, E.; BERTSEKAS, D. Distributed algorithms for generating loop-free routes in networks with frequently changing topology. **Transactions on Communications**, IEEE, v. 29, n. 1, 1981.

HONG, C.-Y. et al. Achieving high utilization with software-driven wan. In: **Proceedings of the ACM SIGCOMM 2013 Conference on SIGCOMM**. New York, NY, USA: ACM, 2013. (SIGCOMM '13), p. 15–26. Available from Internet: <http://doi.acm.org/10.1145/2486001.2486012>.

JAIN, S. et al. B4: Experience with a globally-deployed software defined wan. In: **Proceedings of the ACM SIGCOMM 2013 Conference on SIGCOMM**. New York, NY, USA: ACM, 2013. (SIGCOMM '13), p. 3–14. Available from Internet: <http://doi.acm.org/10.1145/2486001.2486019>.

KANDULA, S. et al. Walking the tightrope: Responsive yet stable traffic engineering. In: **Proceedings of the 2005 Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications**. New York, NY, USA: ACM, 2005. (SIGCOMM '05), p. 253–264. Available from Internet: <http://doi.acm.org/10.1145/1080091.1080122>.

KANDULA, S. et al. Dynamic load balancing without packet reordering. In: . New York, NY, USA: ACM, 2007. v. 37, n. 2, p. 51–62. Available from Internet: <http://doi.acm.org/10.1145/1232919.1232925>.

KANDULA, S. et al. Calendaring for wide area networks. In: **Proceedings of the 2014 ACM Conference on SIGCOMM**. New York, NY, USA: ACM, 2014. (SIGCOMM '14), p. 515–526. Available from Internet: <http://doi.acm.org/10.1145/2619239.2626336>.

KATTA, N. et al. Hula: Scalable load balancing using programmable data planes. In: **Proceedings of the Symposium on SDN Research**. New York, NY, USA: ACM, 2016. (SOSR '16), p. 10:1–10:12. Available from Internet: <http://doi.acm.org/10.1145/2890955.2890968>.

KNIGHT, S. et al. The internet topology zoo. **IEEE Journal on Selected Areas in Communications**, IEEE, v. 29, n. 9, p. 1765–1775, 2011.

KODIALAM, M.; LAKSHMAN, T.; SENGUPTA, S. Efficient and robust routing of highly variable traffic. In: CITESEER. **In Proceedings of Third Workshop on Hot Topics in Networks (HotNets-III**. [S.l.], 2004.

KOPONEN, T. et al. Onix: A distributed control platform for large-scale production networks. In: **OSDI**. [S.l.: s.n.], 2010. v. 10, p. 1–6.

KUMAR, A. et al. Bwe: Flexible, hierarchical bandwidth allocation for wan distributed computing. In: **Proceedings of the 2015 ACM Conference on Special Interest Group on Data Communication**. New York, NY, USA: ACM, 2015. (SIGCOMM '15), p. 1–14. Available from Internet: <http://doi.acm.org/10.1145/2785956.2787478>.

KUZNETSOV, A. N. **Traffic control**. 1999. Available from Internet: <http://lartc.org/manpages/tc.txt>.

LANTZ, B.; HELLER, B.; MCKEOWN, N. A network in a laptop: Rapid prototyping for software-defined networks. In: **Proceedings of the 9th ACM SIGCOMM Workshop on Hot Topics in Networks**. New York, NY, USA: ACM, 2010. (Hotnets-IX), p. 19:1–19:6. Available from Internet: <http://doi.acm.org/10.1145/1868447.1868466>.

LI, Y.; HARMS, J.; HOLTE, R. A simple method for balancing network utilization and quality of routing. In: **Proceedings. 14th International Conference on Computer Communications and Networks, 2005. ICCCN 2005.** [S.l.: s.n.], 2005. p. 71–76.

LIU, J. **Routing Along DAGs**. Thesis (PhD) — UCB/EECS-2011-155, UC Berkeley, 2011. Available from Internet: <http://www2.eecs.berkeley.edu/Pubs/TechRpts/2011/EECS-2011-155.html>.

LIU, J. et al. Ensuring connectivity via data plane mechanisms. In: . USENIX, 2013. (NSDI). Available from Internet: <http://dl.acm.org/citation.cfm?id=2482626.2482639>.

LIU, X. et al. Multipath routing from a traffic engineering perspective: How beneficial is it? In: **2014 IEEE 22nd International Conference on Network Protocols**. [S.l.: s.n.], 2014. p. 143–154.

MATNI, N.; TANG, A.; DOYLE, J. C. A case study in network architecture tradeoffs. In: **Proceedings of the 1st ACM SIGCOMM Symposium on Software Defined Networking Research**. New York, NY, USA: ACM, 2015. (SOSR '15), p. 18:1–18:7. Available from Internet: <http://doi.acm.org/10.1145/2774993.2775011>.

MICHAEL, N.; TANG, A. Halo: Hop-by-hop adaptive link-state optimal routing. **IEEE/ACM Trans. Netw.**, IEEE Press, Piscataway, NJ, USA, v. 23, n. 6, p. 1862–1875, dec. 2015. Available from Internet: <http://dx.doi.org/10.1109/TNET.2014.2349905>.

Open Networking Foundation. **OpenFlow Switch Specification Version 1.5.0 (Protocol version 0x06)**. 2014. Technical Report. Technical Report.

RASLEY, J. et al. Planck: Millisecond-scale monitoring and control for commodity networks. In: **Proceedings of the 2014 ACM Conference on SIGCOMM**. New York, NY, USA: ACM, 2014. (SIGCOMM '14), p. 407–418. Available from Internet: <http://doi.acm.org/10.1145/2619239.2626310>.

ROTHENBERG, C. E. et al. Revisiting routing control platforms with the eyes and muscles of software-defined networking. In: **Proceedings of the First Workshop on Hot Topics in Software Defined Networks**. New York, NY, USA: ACM, 2012. (HotSDN '12), p. 13–18. Available from Internet: <http://doi.acm.org/10.1145/2342441.2342445>.

ROUGHAN, M.; THORUP, M.; ZHANG, Y. Traffic engineering with estimated traffic matrices. In: **Proceedings of the 3rd ACM SIGCOMM Conference on Internet Measurement**. New York, NY, USA: ACM, 2003. (IMC '03), p. 248–258. Available from Internet: <http://doi.acm.org/10.1145/948205.948237>.

RUI, Z.; MCKEOWN, N. Designing a predictable internet backbone network. In: **Proceedings of Third Workshop on Hot Topics in Networks (HotNets-III**. [S.l.: s.n.].

Ryu SDN Framework Community. **RYU network operating system**. 2015. Available from Internet: <https://osrg.github.io>.

SCHLINKER, B. et al. Engineering egress with edge fabric: Steering oceans of content to the world. In: **Proceedings of the ACM Conference on SIGCOMM**. New York, NY, USA: ACM, 2017. (SIGCOMM '17).

SCHMID, S.; SUOMELA, J. Exploiting locality in distributed sdn control. In: **Proceedings of the Second ACM SIGCOMM Workshop on Hot Topics in Software Defined Networking**. New York, NY, USA: ACM, 2013. (HotSDN '13), p. 121–126. Available from Internet: <http://doi.acm.org/10.1145/2491185.2491198>.

SRIDHARAN, A.; GUÉRIN, R.; DIOT, C. Achieving near-optimal traffic engineering solutions for current ospf/is-is networks. **IEEE/ACM Trans. Netw.**, IEEE Press, Piscataway, NJ, USA, v. 13, n. 2, p. 234–247, abr. 2005. Available from Internet: <http://dx.doi.org/10.1109/TNET.2005.845549>.

TIRUMALA, A. et al. Iperf: The tcp/udp bandwidth measurement tool. **https://sourceforge.net/projects/iperf2/**, 2005.

UHLIG, S.; BONAVENTURE, O. Implications of interdomain traffic characteristics on traffic engineering. **Transactions on Emerging Telecommunications Technologies**, Wiley Online Library, v. 13, n. 1, p. 23–32, 2002.

UHLIG, S. et al. Providing public intradomain traffic matrices to the research community. **SIGCOMM Comput. Commun. Rev.**, ACM, New York, NY, USA, v. 36, n. 1, p. 83–86, jan. 2006. Available from Internet: <http://doi.acm.org/10.1145/1111322.1111341>.

VALLET, J.; BRUN, O. Online ospf weights optimization in ip networks. **Computer Networks**, Elsevier, v. 60, p. 1–12, 2014.

WANG, H. et al. Cope: Traffic engineering in dynamic networks. In: **Proceedings of the 2006 Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications**. New York, NY, USA: ACM, 2006. (SIGCOMM '06), p. 99–110. Available from Internet: <http://doi.acm.org/10.1145/1159913.1159926>.

WANG, P. et al. Expeditus: Congestion-aware load balancing in clos data center networks. In: **Proceedings of the Seventh ACM Symposium on Cloud Computing**. New York, NY, USA: ACM, 2016. (SoCC '16), p. 442–455. Available from Internet: <http://doi.acm.org/10.1145/2987550.2987560>.

XU, D.; CHIANG, M.; REXFORD, J. Link-state routing with hop-by-hop forwarding can achieve optimal traffic engineering. **IEEE/ACM Transactions on Networking**, v. 19, n. 6, p. 1717–1730, Dec 2011.

XU, K.; ZHANG, Z.-L.; BHATTACHARYYA, S. Profiling internet backbone traffic: Behavior models and applications. In: **Proceedings of the 2005 Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications**. New York, NY, USA: ACM, 2005. (SIGCOMM '05), p. 169–180. Available from Internet: <http://doi.acm.org/10.1145/1080091.1080112>.

YEGANEH, S. H.; GANJALI, Y. Kandoo: A framework for efficient and scalable offloading of control applications. In: **Proceedings of the First Workshop on Hot Topics in Software Defined Networks**. New York, NY, USA: ACM, 2012. (HotSDN '12), p. 19–24. Available from Internet: <http://doi.acm.org/10.1145/2342441.2342446>.

YEGANEH, S. H.; TOOTOONCHIAN, A.; GANJALI, Y. On scalability of software-defined networking. **IEEE Communications Magazine**, v. 51, n. 2, p. 136–141, February 2013. ISSN 0163-6804.

ZHANG, C. et al. On optimal routing with multiple traffic matrices. In: **Proceedings IEEE 24th Annual Joint Conference of the IEEE Computer and Communications Societies.** [S.l.: s.n.], 2005. v. 1, p. 607–618 vol. 1.

ZHANG, Y. **Abilene topology and traffic dataset**. [S.l.], 2004. Available from Internet: <http://www.cs.utexas.edu/yzhang/research/AbileneTM/>.