Uso de SDN no Balanceamento de Carga em Redes com Suporte a Múltiplos Caminhos

Natália Q. de Oliveira e Ronaldo M. Salles

***Resumo*— A atual arquitetura das redes de computadores é formada de protocolos projetados para serem usados de maneira isolada, provendo soluções específicas, resultando na principal limitação das redes atuais: a complexidade. *Software Defined Networking* (SDN) é uma tecnologia emergente onde o plano de encaminhamento e o plano de controle estão claramente separados. A inteligência da rede SDN é centralizada em controladores, os quais mantém uma visão global da rede. Este artigo propõe um algoritmo de balanceamento de fluxos, implementado em uma aplicação chamada de LBX, a qual define o caminho de custo mínimo baseado no algoritmo *Dijkstra*, em uma topologia com suporte a múltiplos caminhos.**

***Palavras-Chave*— SDN, Redes de Computadores, Balanceamento de Carga**

***Abstract*— The current architecture of computer networks is formed by protocols designed to be used in isolation, providing specific solutions and resulting in major limitation of current networks: complexity. Software Defined Networking (SDN) is an emerging technology where the forwarding plane and the control plane are clearly separated. The intelligence of an SDN network is centralized in controllers, which maintains a global view of the network. This paper proposes a flow balancing algorithm, implemented in an application called LBX, which sets the minimum cost path based on Dijkstra algorithm in a topology with support for multiple paths.**

***Keywords*— SDN, Computers Network, Load Balancing**

I. Introdução

A explosão de dispositivos móveis, virtualização e serviços em nuvem guiam a indústria de redes a reavaliar as arquiteturas das redes tradicionais. O constante crescimento das redes de computadores causaram problemas de desempenho, congestionamento e interrupção de serviços ocasionados por uma sobrecarga dos sistemas. A solução mais utilizada para amenizar esses problemas é o balanceamento de carga. Com o balanceamento, o tráfego é reencaminhado por vários caminhos alternativos a fim de descongestionar os recursos da rede. A necessidade de balanceamento de carga é uma das principais questões que atraem grande quantidade de pesquisa [1], [2] e [3].

Através da tecnologia SDN, acrônimo do inglês *Software Defined Networking*, temos um novo conceito de arquitetura de redes, onde há uma visão e gerência global da rede, além de outros benefícios como: a independência de fornecedores de equipamentos de redes e espaço para inovação, além da redução de complexidade através de automatização. Essa tecnologia propõe a separação completa do plano de controle e do plano de dados, de maneira que o plano de controle torna-se programável, possibilitando assim que o funcionamento da rede passe a ser definido no nível de aplicação. A principal ideia é permitir que os desenvolvedores de *softwares* possam contar com os recursos da rede da mesma forma como fazem com os recursos de armazenamento e computação. A fim de concretizar isso, foi proposto o protocolo de código aberto *OpenFlow* [4], o qual já foi adotado pela *Google*, como apresentado em [5], para soluções de melhoria de engenharia de tráfego em seus próprios *backbones*. Assim, com essa separação dos planos de dados e controle, a empresa teve a possibilidade de escolher o *hardware* específico baseado nas características que atendam às suas necessidades, enquanto é capaz de inovar no desenvolvimento das suas soluções. A partir desde controle centralizado, a *Google* foi capaz de desenvolver soluções mais específicas para seus problemas e que atendam às suas necessidades com a facilidade de serem customizadas.

A principal contribuição deste artigo é a proposta de um algoritmo de seleção de caminho de custo mínimo baseado no algoritmo de *Dijkstra*, para distribuição de carga em redes SDN com suporte à múltiplos caminhos. Os fluxos serão identificados através do valor do campo ToS, acrônimo do inglês (*Type of Service*), presente em cada datagrama analisado, com isso, será possível balancear os fluxos a partir das aplicações identificadas dentro da rede SDN. Todos os caminhos de custo mínimo calculados pelo algoritmo, serão armazenados em um banco de dados, para que possam ser utilizados para recalcular novos caminhos disjuntos de custo mínimo na rede. O algoritmo proposto é capaz de calcular caminhos disjuntos sem interseção de enlaces para o balanceamento dos fluxos, assim poderemos utilizar todo o recurso da rede, fazendo com que os nós e enlaces não fiquem sobrecarregados ou subutilizados.

Por meio de experimentos avaliaremos os resultados a partir de uma visão centralizada e global da rede, além do comportamento do algoritmo proposto. Foram realizados experimentos isolados em dois cenários distintos (C_1 e C_2), onde comparamos os benefícios e desvantagens do balanceamento de carga, da visão centralizada da rede, do balanceador de fluxos proposto e da solução ECMP (*Equal Cost Multipath Routing*) implementada nos equipamentos pelo fabricante *Cisco*.

II. Trabalhos Relacionados

A. Roteamento de Múltiplos Caminhos

O Roteamento de Múltiplos Caminhos, acrônimo em inglês *Multipath Routing*, é uma técnica de roteamento que utiliza múltiplos caminhos alternativos para o tráfego de dados dentro

Natália Q. de Oliveira e Ronaldo M. Salles¸ Departamento de Sistemas e Computação, Instituto Militar de Engenharia - IME, Praça General Tibúrcio, 80 - 22290-270 - Rio de Janeiro - RJ, Brasil, E-mails: queiroz.nati@gmail.com, salles@ime.eb.br.

de uma rede de computadores [2]. Esta técnica pode produzir benefícios, tais como: Balanceamento de Carga, Customização de Aplicações e Resiliência.

O roteamento por múltiplos caminhos facilita a distribuição de tráfego entre uma origem e um destino, a forma como esses pacotes são divididos é uma questão importante. Segundo [2] o encaminhamento de tráfego através de múltiplos caminhos é realizado basicamente através de dois componentes principais: a divisão de tráfego (*traffic splitting*) e a seleção do caminho (*path selection*). O componente da divisão de tráfego divide o tráfego de dados em unidades menores e distribui essas unidades independentemente por vários caminhos baseado em um componente da seleção do caminho. Caso o processador de encaminhamento esteja ocupado, cada unidade de tráfego é enfileirada em uma saída por um componente de seleção de caminho. O componente da seleção do caminho é responsável por escolher um caminho para cada pacote.

B. Modelos de Encaminhamento de Múltiplos Caminhos

Vários modelos de encaminhamento de múltiplos caminhos realizam a distribuição de carga nas mais diferentes maneiras nas redes de computadores atuais. Cada modelo apresenta diferentes vantagens e deficiências, devido à diferença em seus componentes internos. De acordo com [2] os modelos de distribuição de carga existentes podem ser classificados como não-adaptativos e adaptativos, como apresentados na Figura 1.

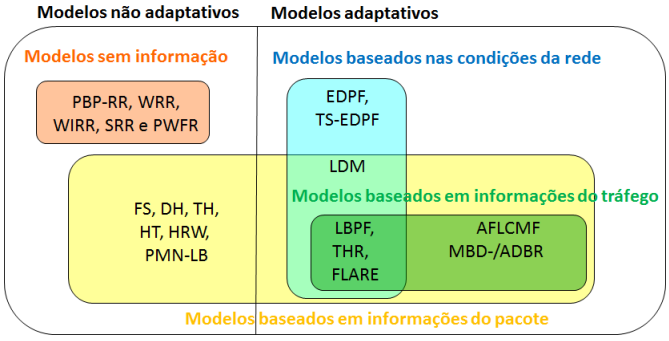


Fig. 1. Modelos de classificação da distribuição de carga. Adaptado de [2]

Os modelos pertencentes à classe "Modelos sem informação", tomam a decisão de distribuição de tráfego sem levar em consideração nenhuma informação externa e os pertencentes à classe "Modelos baseados em Informações do Pacote" necessitam de informações obtidas a partir do cabeçalho do pacote. Os modelos da classe "Modelos sem informação" não coletam nenhuma informação sobre o tráfego ou sobre as condições da rede. A reordenação dos pacotes é o principal problema dos modelos dessa classe. Selecionar o mesmo caminho para todos os pacotes com o mesmo endereço de destino, pode solucionar esse problema. Em [6], os autores propõem um algoritmo para balanceamento de carga para redes de *datacenters* com topologia *fat-tree*. Em [7], é abordado o problema no gerenciamento de QoS, acrônimo do inglês, (*Quality of Service*) em uma rede SDN, o qual também é um problema real nas redes de computadores atuais. No trabalho de [8], é analisado o *overhead* do tráfego excessivo em ambientes de *datacenters* e mostra este comportamento aplicado em uma rede SDN. Já a abordagem de [9], analisa como SDN impacta na performance de uma aplicação logicamente centralizada. Em [10] é retratado um levantamento exaustivo na literatura sobre soluções de suporte a múltiplos caminhos.

Com o levantamento bibliográfico vimos que a maioria dos algoritmos de seleção de caminho em redes com suporte à múltiplos caminhos, não levam em consideração uma visão centralizada da rede e sim a visão parcial do nó, onde o algoritmo está sendo executado. Isso faz com que cada nó realize a sua própria seleção de caminho e envio de fluxos dentro da rede. Além disso, cada nó sofre um alto nível de processamento para poder computar seus algoritmos e definir a seleção do caminho dentro da rede. A solução desses algoritmos são complexas e limitadas para se implementar e administrar, cada fornecedor tem a sua própria solução para tratar de problemas específicos, não encontrando assim espaço para inovação.

III. SDN e OpenFlow

SDN é uma arquitetura de rede de computadores emergente onde o controle de rede está dissociado do encaminhamento de pacotes e que pode ser diretamente programável [11].

A Figura 2 retrata a arquitetura da tecnologia SDN. Esta arquitetura é composta por 3 camadas camadas. Na camada inferior, ou camada de infraestrutura, temos os equipamentos de rede com o protocolo *OpenFlow*. Esses equipamentos possuem somente o plano de encaminhamento de pacotes e que executam as ações dos controladores. Na camada de controle, temos os controladores baseados em software que possuem a visão global da rede e que gerenciam as políticas de encaminhamento de pacotes na arquitetura SDN. Na camada superior, ou camada de aplicação, temos um conjunto de aplicações (APIs), que torna, possível a criação, customização ou implementação dos mais comuns serviços de redes incluindo: roteamento, segurança, controle de acesso, gerenciamento de largura de banda, engenharia de tráfego, balanceamento de carga, qualidade de serviço, entre outros.

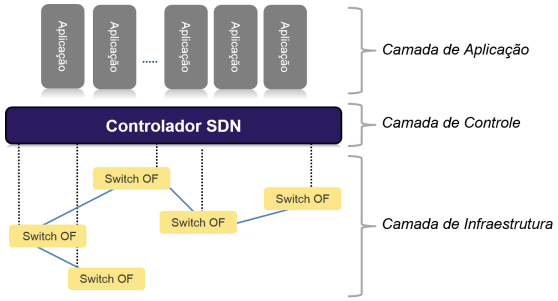


Fig. 2. Arquitetura SDN

De acordo com [4], o protocolo *OpenFlow* é a primeira interface padrão de comunicação definida entre o controlador e as camadas de encaminhamento de uma arquitetura de redes definidas por software. O *Open Networking Foundation* [10] é

responsável pela padronização do protocolo *OpenFlow*, a fim de garantir a interoperabilidade entre dispositivos de rede e software de controle de diferentes fornecedores. O protocolo especifica primitivas básicas que podem ser usadas por uma aplicação de software externo para programar o plano de encaminhamento dos dispositivos de redes, assim como o conjunto de instruções em uma CPU que podem programar um sistema de computador. O protocolo do *OpenFlow* é implementando como interface de comunicação entre os dispositivos de rede e a camada SDN de controle.

IV. Algoritmo Proposto

O Algoritmo 1 apresenta o pseudocódigo do algoritmo proposto neste artigo. Os fluxos identificados em cada datagrama será balanceado pelo enlace de maneira justa, respeitando a necessidade das aplicações definidas pelo administrador da rede. O algoritmo proposto será implementado em uma aplicação de balanceamento de fluxos chamada de LBX (*Load Balance X*), projetada para uma rede SDN com suporte a múltiplos caminhos. Sua principal característica é a possibilidade da seleção de caminhos disjuntos de custo mínimo, os quais não tenham interseção de enlaces.

Para um novo fluxo analisado dentro da rede SDN, o algoritmo primeiramente identifica a origem, o destino e o identificador do fluxo obtidos através do cabeçalho da mensagem analisada. Logo após, é verificado no banco de dados se existe algum caminho já calculado para determinado fluxo. Caso não exista, o caminho é calculado, utilizando *Djikstra*, e devidamente armazenado no banco de dados. Caso já exista um caminho, o algoritmo verifica se existe um novo caminho disjunto, através do método CalculateLBX(path), desconsiderando a interseção dos enlaces do caminho previamente calculado. Caso exista, o fluxo é enviado pelo novo caminho e também é armazenado no banco de dados. Caso contrário, o fluxo é encaminhado pelo caminho já existente.

```
src = packet-in(flow);
dst = packet-in(flow);
flowID = packet-in(flow);
path = QueryDB(sr,dst,flowID);
if path ≠ null then
    newPath = CalculateLBX(path);
else
    CalculateDijkstra();
end
```

Algoritmo 1: Algoritmo Proposto

O Algoritmo 2 apresenta o pseudocódigo do método CalculateLBX(path). A diferença entre o método CalculateLBX(path) e o *Djikstra* é a saída. No método CalculateLBX(path) retorna um novo caminho de custo mínimo, desconsiderando a interseção dos enlaces do caminho previamente calculado.

V. Simulação e Resultados

Foram elaborados dois cenários distintos (C_1 e C_2). O cenário C_1 apresenta uma topologia mínima com 3 *switches*,

```
distance[source] ← 0; distance[u] ← ∞ , para cada u ≠ 0, u ∈ V);
insert [u] with key d[u] into priority queue PQ, for each u ∈ V;
remove adjRemove into priority queue PQ, for each u ∈ V;
while PQ ≠ null do
    u ← ExtractMin(PQ);
    foreach v adjacent to u do
        if distance[v] > distance[u] + edgeW[u,v] + 1
        then
            distance[v] ← distance[u] + edgeW[u,v] + 1;
            p[v] ← distance[u];
        else
        end
    end
end
```

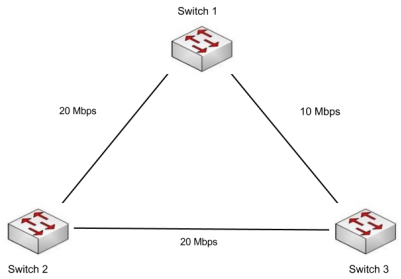
Algoritmo 2: Método CalculateLBX(path)

com duas possíveis escolhas de caminho para cada origem e destino. Durante os testes, foram realizados 5 experimentos distintos com a sua particularidade, a partir de vários fluxos gerados da origem *Switch* 1 com destino para o *Switch* 2. O cenário C_2, apresenta uma topologia com 6 *switches* com três possíveis escolhas de caminho entre a origem *Switch* 1 e o destino *Switch* 6. Para os testes, foram realizados 3 experimentos distintos, a partir de vários fluxos gerados da origem *Switch* 1 com destino para o *Switch* 6. Em ambos os cenários cada enlace da topologia foi configurado com uma banda específica, 100 ms de delay e 1 % de perda de pacotes.

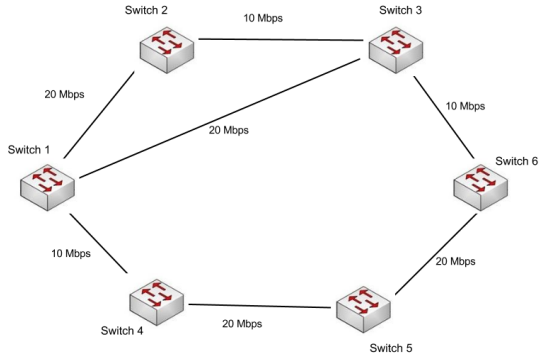
Os fluxos foram gerados através da ferramenta *Iperf* [12], a partir dessa ferramenta podemos testar e medir o *throughput*, que é a taxa de entrega de pacotes da rede e o *packet-loss*, que é a taxa de perda de pacotes. A distribuição utilizada para os fluxos gerados durantes as simulações foi a uniforme. A fim de congestionar a rede SDN, nos experimentos destes cenários foram enviados 15 fluxos UDP de 1 Mbps e 30 fluxos TCP de 64 *bits* cada, entre a origem o e destino. Os fluxos TCP e UDP foram enviados aleatoriamente para assim avaliarmos a distribuição dos fluxos e o desempenho da rede. Notamos que enviando 15 fluxos UDP e 30 fluxos UDP o enlaces entre a origem e o destino ficavam congestionados. Assim definimos esta quantidade de fluxos para os experimentos a seguir.

A. Cenário C_1

No Cenário C_1 foram realizados 5 experimentos distintos. No experimento 01 avaliamos a rede SDN sem a realização do balanceamento de carga. No experimento 02 avaliamos a rede SDN utilizando a aplicação do balanceamento de fluxos LBX. No experimento 03 avaliamos o balanceamento de carga simulando o envio de 2 aplicações separadamente. Para isso, foram enviados somente 2 tipos de ToS nos cabeçalhos dos fluxos *OpenFlow*. No experimento 04 avaliamos o balanceamento de carga levando em consideração que os caminhos utilizados tenham o mesmo valor de banda total do Experimento 01. No experimento 05 avaliamos a mesma condição do experimento



(a) Topologia do cenário C_1



(b) Topologia do cenário C_2

Fig. 3. Topologias do Ambiente de Teste

03, porém consideração que o caminho utilizado tenha uma banda menor, neste caso 20 Mpbs em vez de 30 Mpbs.

A Tabela I apresenta um comparativo sobre os valores de *throughput* por segundo nos experimentos 01, 02, 03, 04 e 05. Percebemos que os valores do experimento 01 são bastante inferiores aos dos experimentos 02 e 03, os quais utilizam a aplicação do balanceamento de fluxos LBX. Notamos que nos experimentos onde há o balanceamento de fluxos LBX, os valores do *throughput* são melhores mostrando assim que a taxa de transferência da rede SDN é maior com a utilização do balanceamento de carga. Podemos observar que os índices nos experimentos 04 e 05, os quais utilizam o balanceamento de fluxos LBX ainda são mais significativos aos do experimento 01 o qual não possui balanceamento de carga, mesmo utilizando a mesma banda total (20 Mbps) em todos os experimentos. A taxa de *throughput* foi avaliada em Mpbs. Os resultados apresentados apresentam a média de diversas simulações.

Tabela II apresenta um comparativo sobre os valores de *packet-loss* por segundo obtidos nos experimentos 01, 02, 03, 04 e 05. Os valores foram bastantes discrepantes em comparação do experimento 01 aos experimentos 02 e 03. Notamos que a porcentagem de *packet-loss* nos experimentos 02 e 03, os quais utilizam o balanceamento de fluxos são significantemente menores do que os comparados com o experimento 01, sem o balanceamento de fluxo. Por não possuir balanceamento de carga, os valores do *packet-loss* são bastante altos em comparação com os experimentos 02 e 03. Nos experimentos com balanceamento de carga ocorre uma diminuição de *packet-loss* e o aumento do *throughput*

TABELA I

Tabela throughput nos experimentos 01, 02, 03, 04 e 05.

Seg	Exp01	Exp02	Exp03	Exp04	Exp05
1	2,2	8,1	10,3	10,2	13,8
2	2,5	8,5	10,7	10,7	13,6
3	2,1	8,2	10,3	10,5	13,6
4	2,5	8,5	10,5	10,6	13,1
5	2,6	8,9	11,1	10,8	13,3
6	2,8	8,3	10,4	10,7	13,7
7	2,7	8,4	10,6	10,7	13,8
8	2,5	8,2	10,1	10,4	13,5
9	2,3	8,8	10,4	10,2	13,8
10	2,2	8,9	10,8	10,8	13,4

na rede, tornando assim a rede menos sub-utilizada e menos congestionada. Analisamos que a porcentagem de *packet-loss* nos experimentos 04 e 05, os quais utilizam o balanceamento de fluxo ainda são significantemente menores do que os comparados com o experimento 01, mesmo utilizando a mesma banda total (20 Mbps) nos experimentos. A taxa de *packet-loss* foi avaliada em %. Os resultados apresentados apresentam a média de diversas simulações.

TABELA II

Tabela packet-loss nos experimentos 01, 02, 03, 04 e 05.

Seg	Exp01	Exp02	Exp03	Exp04	Exp05
1	30	4,5	3,8	3	2,9
2	40	5,5	4,4	3,4	3
3	23	4	3,5	2,8	2,6
4	21	3,6	3,8	2,9	2,5
5	36	4	5,3	2,6	13,8
6	45	5,2	4,6	4,6	4,4
7	28	4,5	2,9	2,9	2,5
8	24	4,1	3,8	2,9	2,5
9	32	4,2	3,5	2,9	2,4
10	50	4	3,5	3,6	3

B. Cenário C_2

No Cenário C_2 foram realizados 3 experimentos distintos. Novamente no experimento 01 avaliamos a topologia sem a realização do balanceamento de carga, no experimento 02 avaliamos a utilização do balanceamento de fluxos proposto e por fim no experimento 03 avaliamos o balanceamento de carga com a utilização da solução ECMP, a qual é implementado pela empresa *Cisco* em seus equipamentos de rede.

A solução ECMP é implementada através do modelo DH (*Direct Hashing*). O DH realiza o balanceamento da carga baseado na realização de uma função *hash* para as rotas (*Equal Cost Multipath Routing*). A fim de obter um caminho, ele executa um algoritmo de *hash* módulo-*K* que leva em consideração o identificador do pacote *X*, obtido através das informações do cabeçalho do pacote. Assim ele aplica uma função *hash h(X)* e extrai o módulo do número dos múltiplos caminhos *mod(h(X))*. A Tabela III apresenta um comparativo sobre os valores de *throughput* por segundo obtidos nos experimentos 01, 02 e 03. A taxa de *throughput* foi avaliada em Mpbs. Os resultados apresentados apresentam a média de diversas simulações.

TABELA III

Tabela throughput experimentos 01, 02 e 03.

Seg	Exp01	Exp02	Exp03
1	1,2	12,2	11,5
2	1,3	12,3	11
3	1,1	12,6	11,4
4	1,5	12,1	11,1
5	1,2	12	11
6	1,8	12,1	11,4
7	1,7	12,8	11,2
8	1,5	12,7	11,3
9	1,3	12,4	11,1
10	1,62	12,3	11,8

A aplicação LBX mostrou-se estável para a realização do balanceamento de fluxos, pois ela não realiza o *hash* do cabeçalho dos fluxos, como a solução do ECMP. Ele os encaminha baseado nas informações de origem, destino e identificador de fluxo armazenadas no banco de dados da aplicação LBX. A solução ECMP, implementada pela *Cisco*, distribui os pacotes pertencentes a um determinado fluxo por *N* caminhos distintos, o que implica na necessidade de reordenação desses pacotes na chegada ao destino desse fluxo. Neste caso torna-se necessário a implementação de um método para o controle da ordenação desses pacotes, já que eles podem ser enviados fora de ordem. O balanceador de fluxos LBX, avaliado no experimento 02 deste cenário, seleciona o caminho de custo mínimo baseado no algoritmo de *Dijkstra* utilizando informações previamente armazenadas no banco de dados da aplicação e distribui os fluxos por caminhos disjuntos, não sendo necessário a reordenação dos pacotes no seu destino. Com isso, temos vantagens na rede SDN, pois não se torna necessária nenhuma reordenação dos pacotes no seu destino.

A Tabela IV apresenta um comparativo sobre os valores de *packet-loss* obtidos nos experimentos 01, 02 e 03. A taxa de *packet-loss* foi avaliada em %. Os resultados apresentados apresentam a média de diversas simulações.

TABELA IV

Tabela packet-loss experimentos 01, 02 e 03.

Seg	Exp01	Exp02	Exp03
1	64	4,6	6,8
2	82	5,8	6
3	55	4,2	6,4
4	48	4,1	5,9
5	66	4,5	5,3
6	75	4,2	5,7
7	88	4	5,9
8	66	4,5	5
9	82	4,4	5
10	41	4,3	5,3

VI. CONCLUSÕES

Com a implementação do algoritmo proposto e através dos experimentos foi possível destacar os benefícios da tecnologia SDN. Através das logs da tabela do banco de dados e dos resultados do servidor *Iperf* vimos que o balanceamento de fluxos ocorreu de fato em ambos os cenários abordados. Observamos que o campo ToS, utilizado como campo chave para identificarmos o fluxo dentro da rede SDN, atendeu as expectativas, pois os fluxos divergentes foram trafegados ao longo dos múltiplos caminhos e os fluxos iguais seguiram os mesmo caminho dentro da topologia.

Todos os caminhos de custo mínimo selecionados na topologia foram devidamente arquivados e monitorados pela aplicação LBX na busca de novos caminhos disjuntos para o balanceamento dos fluxos. Vimos que a falta do balanceamento de carga continua sendo um problema atual e também em redes SDN, porém através de SDN temos espaço para inovação no assunto e liberdade de construir novas soluções através de APIs abertas e customizadas. Com esse grande benefício observamos que com SDN podemos programar diretamente a rede, fazendo que a mesma se comporte de acordo com as políticas e regras definidas a partir de uma visão centralizada. Assim podemos melhor definir, modificar e tratar o problema de balanceamento de carga nas redes atuais.

Com os experimentos realizados nos cenários C_1 e C_2 observamos que a aplicação proposta neste trabalho mostrou ótimos valores com o suporte da tecnologia de redes definidas por *software*. A expectativa é de que os mesmos experimentos realizados em redes maiores e reais apresentem também resultados favoráveis. Além da possibilidade do identificador de fluxo ser também aproveitado para outra tomada de decisão como prioridade de tráfego em uma rede SDN, com suporte a múltiplos caminhos.

REFERÊNCIAS

[1] R. Roy e B. Mukherjee, "Degraded-service-aware multipath provisioning in telecom mesh networks" *Optical Fiber Communication Conference*, pp 1–3, 2008.

[2] S. Prabhavat, H. Ansari, N. Kato, "Effective delay-controlled load distribution over multipath networks"*Parallel and Distributed Systems*, IEEE Transactions on, 22:1730–1741, 2011.

[3] Singh, R. Kumar, N. S. Chaudhari, and K. Saxena. "Load balancing in IP/MPLS networks: a survey.*Communications and Network*, 2012."

[4] N. McKeown, T. Anderson, H. Balakrishnan, G. Parulkar, L. Peterson, J. Rexford, S. Shenker e J. Turner, "Openflow: enabling innovation in campus networks". *ACM SIGCOMM Computer Communication Review*, 38:69–74, 2008.

[5] S. Jain, A. Kumar, S. Mandal, J. Ong, L. Poutievski, A. Singh, S. Venkata, J. Wanderer, J. Zhou, M. Zhu, J. Zolla, U. Holzle, S. Stuart, e A. Vahdat, "B4: experience with a globally-deployed software defined wan". *SIGCOMM Computer Communication Review*, 43:3–14, 2013.

[6] A. Tolk, S. Diallo, I. Ryzhov, L. Yilmaz, S. Buckley, J. and Miller, "A simulation and emulation study of sdn-based multipath routing for fat-tree data center networks", 2014.

[7] E. Chemeritskiy e R. Smeliansky, "On qos management in sdn by multipath routing". *In Science and Technology Conference (Modern Networking Technologies) (MoNeTeC)*, 2014 First International, pp 1–6. IEEE.

[8] A. R. Curtis, J. C. Mogul, J. Tourrilhes, P. Yalagandula, P. Sharma, S. and Banerjee, "Devoflow: Scaling flow management for high-performance networks".*In ACM SIGCOMM Computer Communication Review*, volume 41, pp 254–265, 2011. ACM.

[9] D. Levin, A. Wundsam, B. Heller, N. Handigol e A. Feldmann, "Logically centralized: state distribution trade-offs in software defined networks". *In Proceedings of the first workshop on Hot topics in software defined networks*,pp 1–6, 2012. ACM.

[10] Qadir, Junaid, et al. "Exploiting the power of multiplicity: a holistic survey of network-layer multipath."*IEEE Communications Surveys & Tutorials*v. 17, n. 4, p. 2176-2213, 2015.

[11] O. N. Foundation, "The new norm for networks". *ofn white paper*. https://www.opennetworking.org/sdn-resources/sdnlibrary/whitepapers, 2012.

[12] Iperf. http://iperf.fr/. Acesso em março de 2016.