Balanceamento Dinâmico de Carga para Funções Virtuais sobre Comutadores OpenFlow Heterogêneos

João Victor G. de Oliveira[1], **Pedro C. P. Bellotti**[1], **Rafael G. Motta**[1], **Roberto M. de Oliveira**[1], **Alex B. Vieira**[1], **Luciano J. Chaves**[1]

[1] Departamento de Ciência da Computação – Universidade Federal de Juiz de Fora
{joaoguimaraes, pedro.bellotti, rafael.motta, rmassi,
alex.borges, luciano.chaves}@ice.ufjf.br

Abstract. *Implementing a virtual network function by distributing rules among programmable SDN switches enables efficient NFV adoption in scenarios that require scalable packet processing. On that subject, we propose a dynamic load balancing mechanism for network functions virtualized over heterogeneous OpenFlow switches, along with two load balancing policies. The static policy distributes new traffic flows randomly among the switches. The dynamic policy considers the singularities of the switches (in hardware and software) in the initial flow allocation and in their subsequent reallocation, for better usage of available resources. We experimentally evaluated the mechanism under different load levels. The results confirm its efficiency; even in an overloaded scenario, there is a* $\approx 27\%$ *increase in the virtual function processing capacity.*

Resumo. *Implementar uma função virtual de rede através de regras distribuídas entre comutadores SDN programáveis possibilita a adoção eficiente do NFV em cenários que necessitem de processamento escalável de pacotes. Nesse sentido, propomos um arcabouço de balanceamento dinâmico de carga para funções virtualizadas sobre comutadores OpenFlow heterogêneos, acompanhado de duas políticas de balanceamento. A política estática distribui novos fluxos aleatoriamente entre os comutadores. A política dinâmica considera as especificidades dos comutadores (em hardware e em software) na alocação inicial dos fluxos e posterior realocação dos mesmos, para melhor aproveitamento dos recursos disponíveis. Avaliamos o arcabouço experimentalmente, sob diferentes níveis de carga. Os resultados confirmam a eficiência do arcabouço; mesmo em um cenário com sobrecarga de tráfego, há um aumento da capacidade de processamento da função virtual em* $\approx 27\%$.

1. Introdução

A crescente demanda por conectividade à Internet tem forçado a evolução das tradicionais arquiteturas de redes de computadores para um modelo mais sustentável, capaz de oferecer acesso ubíquo aos usuários sem aumentar significativamente o custo de implantação e operação dessas redes para os provedores de infraestrutura. Dentre as principais limitações da arquitetura tradicional está a dependência de *hardware* dedicado para a implementação de funções de rede, como roteadores, *firewalls*, *gateways*, balanceadores de carga, etc. [Chaves et al. 2015]. Este modelo requer um alto investimento para a aquisição e instalação de equipamentos, normalmente com recursos superdimensionados capazes de atender ao picos de tráfego e acomodar o crescimento da rede em curto prazo.

Com o objetivo de diminuir os custos e aumentar a flexibilidade, o paradigma de Virtualização das Funções de Rede (NFV, do Inglês *Network Functions Virtualization*) tem se mostrado uma alternativa atraente, recebendo atenção tanto da academia como da indústria [ETSI NFV 2012]. Através da separação entre o *software* responsável pela função de rede e o *hardware* onde ele é executado, o NFV permite o uso compartilhado de recursos computacionais de propósito geral para a execução de instâncias independentes de Funções Virtuais de Rede (VNFs, do Inglês *Virtual Network Functions*). A abordagem convencional consiste em implementar as VNFs como *software* rodando em máquinas virtuais (VMs) ou *containers* que são instanciados em servidores na nuvem. Dessa forma, as VNFs podem ser personalizadas e atualizadas com mais agilidade enquanto os recursos de *hardware* passam a ser melhor utilizados, visto que as VNFs podem ser alocadas dinamicamente de acordo com a demanda da rede [Yi et al. 2018].

Como um facilitador para a implantação do NFV, o paradigma das Redes Definidas por *Software* (SDN, do Inglês *Software Defined Networks*), juntamente com o protocolo OpenFlow, apresentam uma abordagem moderna para o gerenciamento de tráfego em uma rede de comutadores [Open Networking Foundation 2012]. A inteligência e o controle da rede são logicamente centralizados em *software* enquanto o plano de dados é fisicamente distribuído em *hardware* programável e otimizado para o encaminhamento de pacotes. Dessa forma, as tomadas de decisão são facilitadas pela visão global da rede, permitindo ao SDN direcionar os pacotes até as VMs para oferecer o encadeamento adequado do tráfego entre as VNFs e prover serviços de forma dinâmica através das Cadeias de Funções de Serviço (SFCs, do Inglês *Service Function Chains*) [Kreutz et al. 2015].

Entretanto, a flexibilidade do NFV vem acompanhada de novos desafios, dentre eles a escalabilidade [Wang et al. 2016]. Algumas VNFs requerem servidores de alto desempenho para lidar com intensas cargas de trabalho, principalmente quando precisam realizar inspeção individual de pacotes em grandes fluxos de dados. Mesmo com os recentes avanços tecnológicos, o processamento de pacotes por servidores de propósito geral ainda apresenta desempenho moderado quando comparado às plataformas de *hardware* de propósito específico [Nguyen et al. 2016]. Nesse contexto, uma das estratégias comumente adotada para aumentar a escalabilidade é replicar as VNFs e utilizar o SDN para balancear a carga de trabalho entre as instâncias disponíveis [Laghrissi e Taleb 2019].

Visto que a natureza de algumas VNFs está estritamente relacionada com a inspeção massiva de pacotes e possíveis ações sobre eles, [Chaves et al. 2017] propuseram uma abordagem diferente para a virtualização desta categoria de funções de rede: substituir o *software* em uma VM por um conjunto de regras programadas em comutadores OpenFlow, de modo a explorar o alto desempenho do *hardware* dedicado. Como um único comutador pode não ser suficiente para processar todo o tráfego, a escalabilidade desta VNF pode ser alcançada com a distribuição das regras de fluxo entre vários comutadores e o adequado balanceamento do tráfego entre eles.

Como principal contribuição deste trabalho, evoluímos a proposta original de [Chaves et al. 2017] para incorporar um plano de dados heterogêneo, que combina comutadores em *hardware* e em *software* na instanciação da VNF. Essa abordagem oferece flexibilidade ao provedor da infraestrutura: ele pode optar por comutadores OpenFlow com *hardware* dedicado ao processamento de pacotes para explorar o alto desempenho do dispositivo sem perder a programabilidade oferecida pelo mesmo; e/ou pode optar por

comutadores OpenFlow em *software*, que flexibiliza a instanciação da VNF em servidores de propósito geral, mesmo que isso implique em alguma redução na capacidade de processamento da função virtual. Um cenário heterogêneo como esse poderia ser adotado em um ambiente onde o provedor da infraestrutura mantenha um (ou alguns) comutadores físicos em operação, garantindo continuidade no oferecimento do serviço, mas instancie outros comutadores virtuais para atender demandas temporárias de tráfego.

Propomos também um arcabouço de balanceamento dinâmico de carga para distribuir adequadamente as regras de fluxos entre os comutadores heterogêneos que compõem a VNF. Este arcabouço identifica reativamente a chegada de novos fluxos e, com a ajuda do controlador SDN centralizado, verifica a disponibilidade de recursos para instalar as regras de processamento da função virtual no comutador interno apropriado. Como prova de conceito, nós desenvolvemos duas políticas para validar nosso balanceador em um cenário realista. A *política estática* distribui novos fluxos aleatoriamente entre os comutadores. Por outro lado, a *política dinâmica* considera as especificidades dos comutadores de *hardware* e *software* na alocação inicial dos fluxos e posterior realocação dos mesmos, de modo a extrair o máximo que cada tipo de infraestrutura pode oferecer.

Por fim, validamos o arcabouço proposto através de experimentos em um pequeno ambiente de testes e comprovamos a importância de se considerar a heterogeneidade dos comutadores no balanceamento da carga. Avaliamos três perfis de tráfego, variando a taxa de chegada de novos fluxos. Para o cenário com sobrecarga, os resultados mostram que a política dinâmica, quando comparada à política estática, é capaz de aumentar a capacidade de processamento da VNF em aproximadamente 27%, eliminando os bloqueios por indisponibilidade de recursos, e com impacto mínimo nos indicadores de QoS dos fluxos.

O restante deste artigo está organizado da seguinte maneira: a Seção 2 apresenta uma revisão da literatura enquanto a Seção 3 descreve o cenário adotado neste trabalho. A Seção 4 detalha o arcabouço de balanceamento dinâmico de carga juntamente com as duas políticas propostas. A Seção 5 apresenta e discute os resultados experimentas e as conclusões são expostas na Seção 6.

2. Trabalhos relacionados

O balanceamento de carga é a técnica que divide uma demanda por vários recursos computacionais em uma rede, como enlaces e servidores, de modo a otimizar a utilização dos mesmos. Este é um tópico atraente e bastante explorado na literatura, principalmente quando envolve os paradigmas SDN e NFV.

[Zhang et al. 2018] apresentam uma extensa revisão de literatura sobre soluções para o balanceamento de tráfego em *datacenters*, com destaque para os mecanismos centralizados de distribuição do fluxos pelos enlaces da infraestrutura de rede SDN. [Jamali et al. 2019] também discutem o balanceamento entre enlaces e comutadores numa rede SDN, utilizando técnicas de programação genética para encontrar rotas que evitem gargalos e que minimizem o custo das operações de configuração dos comutadores. Os trabalhos de [Laghrissi e Taleb 2019] e [Carpio et al. 2017] apresentam discussões sobre o problema de posicionamento de funções virtuais de rede num ambiente que adote NFV. Os autores dedicam atenção especial ao discutir soluções de posicionamento distribuído de VNFs que sejam cientes do balanceamento da carga nos enlaces, incluindo aspectos em relação à replicação de instâncias destas funções. Num contexto semelhante,

[An et al. 2014] propõem a virtualização do plano de dados do *gateway* de pacotes (P-GW) utilizado na arquitetura das redes móveis 4G. Preocupados com a escalabilidade desta VNF, os autores adotam um conjunto local de réplicas da VNF e o uso de um par de comutadores OpenFlow para distribuir o tráfego entre as instâncias ativas.

Nos trabalhos anteriores, a existência de uma infraestrutura SDN atua como um facilitador para a implementação das técnicas de balanceamento de carga, dado a fácil programabilidade destas redes. Entretanto, o SDN também pode ser usado para implementar funções virtuais de rede no contexto de NFV, principalmente aquelas que demandam de inspeção individual de pacotes. [Kaur et al. 2017] usam uma rede de comutadores OpenFlow para implementar um *firewall* (sem estado) distribuído, espalhando as regras de processamento de fluxos entre os comutadores da rede e eliminando o ponto único de falha da arquitetura tradicional. Outro cenário pode ser visto em [Rodrigues et al. 2015], onde os autores propõem a adoção de um comutador OpenFlow ligado à um controlador centralizado para substituir os equipamentos especializados de balanceamento de carga em nível de aplicação. Com a arquitetura proposta é possível implementar diferentes políticas para distribuir requisições HTTP entre servidores Web, oferecendo escalabilidade ao serviço.

O trabalho de [Chaves et al. 2017] propõe a virtualização do plano de dados do *gateway* P-GW unicamente através de comutadores SDN. Para isso, eles implementam as tarefas de responsabilidade do *gateway* como regras de fluxo OpenFlow, incluindo aquelas necessárias para a classificação individual dos fluxos de pacotes nos túneis usados pela rede 4G, além de limitadores de vazão e contabilização de estatísticas para fins de cobrança. Como a quantidade de regras cresce proporcionalmente com o número de fluxos ativos na rede e considerando as limitações no número máximo de regras em comutadores OpenFlow de prateleira, os autores sugerem que as regras que implementam o P-GW sejam distribuídas entre um conjunto de comutadores idênticos para aumentar a escalabilidade. Para isso, eles propuseram um mecanismo adaptativo para ativar/desativar os comutadores e mover as regras de fluxos entre eles de acordo com a demanda da rede.

Para flexibilizar a proposta de [Chaves et al. 2017], é interessante que a expansão de capacidade da VNF possa acontecer também com a agregação de comutadores heterogêneos, incluindo também comutadores virtualizados. É neste contexto que este trabalho apresenta um arcabouço para o balanceamento dinâmico de fluxos entre comutadores OpenFlow heterogêneos usados internamente na virtualização de uma única função de rede. Juntamente com o arcabouço, apresentamos duas políticas de balanceamento usadas para validar a importância de se considerar as características dos comutadores OpenFlow (que aqui fazem o papel da infraestrutura de virtualização) nas tomadas de decisão.

3. Cenário OpenFlow heterogêneo para VNFs

O cenário adotado neste artigo foi inspirado no trabalho de [Chaves et al. 2017]. Ele considera a implementação escalável de uma VNF genérica através de regras OpenFlow distribuídas entre diversos comutadores, que chamaremos aqui de *comutadores de função*. Este cenário é atrativo para a virtualização de funções de redes que lidam diretamente com a inspeção dos cabeçalhos dos pacotes para tomada seletiva de decisão. A motivação original para o mesmo é a virtualização do plano de dados de um *gateway* P-GW da arquitetura de redes 4G, onde existe ao menos uma regra OpenFlow instalada num comutador de função para cada fluxo ativo na rede. Neste caso, cada regra é responsável por identifi-

car o fluxo ao qual o pacote pertence, contabilizar o tráfego, monitorar a vazão e realizar as ações necessárias para o encapsulamento do pacote para que ele seja direcionado ao túnel correto dentro da arquitetura da rede. Sem perda de generalidade, o mesmo cenário poderia ser adotado na implementação de outras VNFs de natureza semelhante. Como exemplo, poderíamos considerar um *firewall*, onde cada regra OpenFlow instalada num comutador de função poderia identificar pacotes segundo algum padrão e tomar a decisão de encaminhar ou descartar o mesmo.

A quantidade de regras OpenFlow necessárias para implementar uma VNF pode aumentar consideravelmente com a complexidade da função e a carga de trabalho, então um único comutador de função pode não ser capaz de realizar essa tarefa sozinho. A solução imediata é combinar vários comutadores de função e dividir o trabalho entre eles. Para isso, precisamos também de novos comutadores OpenFlow responsáveis apenas por redirecionar os fluxos aos comutadores de função adequados. Daremos a estes novos comutadores o nome de *comutadores de redirecionamento*.

A Figura 1 ilustra a topologia dos comutadores de função e redirecionamentos adotada neste trabalho. Nesta topologia, o controlador em *software* centralizado é o responsável pela gerência do tráfego dentro da VNF e também pela instalação, atualização e remoção das regras de fluxo nos comutadores de função e de redirecionamento.

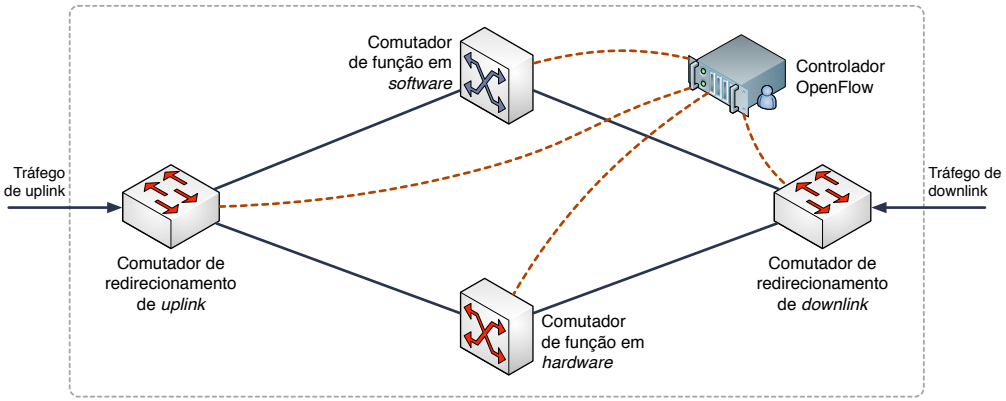

Figura 1. Comutadores heterogêneos implementando uma única VNF.

Um diferencial importante do cenário adotado neste trabalho em relação ao cenário proposto por [Chaves et al. 2017] é a utilização de comutadores de função com recursos distintos, caracterizando uma abordagem heterogênea composta por um comutador em *hardware* e por um comutador em *software*. Como discutido em [Costa et al. 2017], comutadores OpenFlow implementados em *hardware* são otimizados para o encaminhamento de pacotes, suportam altas cargas de trabalho e introduzem baixo atraso nos fluxos de dados. Entretanto, sofrem de uma limitação importante: o tamanho reduzido das tabelas de fluxo. As tabelas pequenas são consequência do alto custo e elevado consumo energético das Memórias de Conteúdo Ternário Endereçável (TCAM, do Inglês *Ternary Content-Addressable Memory*), utilizadas para a implementação eficiente das regras OpenFlow que envolvem máscaras nos campos de busca. Por outro lado, implementações em *software* de comutadores OpenFlow (como o *Open vSwitch*[1]), podem ser instanciadas em servidores de prateleira, são facilmente configuráveis e possuem tabelas de fluxos bem

[1] `https://www.openvswitch.org`

maiores. Entretanto, o atraso no encaminhamento de pacotes num ambiente virtualizado pode ser até uma ordem de grandeza maior em relação ao atraso de um comutador em *hardware*, o que limita consideravelmente sua capacidade de processamento.

A combinação entre comutadores de função heterogêneos pode ser benéfica quando o provedor da infraestrutura de virtualização mantenha um ou mais comutadores OpenFlow em *hardware* sempre disponíveis para a implementação da VNF, enquanto que novas instâncias temporárias de comutadores em *software* podem ser instanciadas sob demanda para atender aos picos de tráfego na rede. Entretanto, neste trabalho estamos considerando uma topologia estática com exatamente dois comutadores de função heterogêneos, sem instanciação sob demanda de outros comutadores.

Ainda sobre o cenário adotado, os comutadores de redirecionamento, individualmente identificados como *UpLink* (UL) e *DownLink* (DL), atuam como pontos de entrada e saída de tráfego na VNF. Dessa forma, esses comutadores podem estar sujeitos à grandes cargas de tráfego e por isso devem dispor de *hardware* de alto desempenho e um número comedido de regras de fluxo [Chaves et al. 2017]. Esses comutadores são responsáveis por distribuir o fluxos proveniente de ambas as direções entre os comutadores de função, que de fato implementam as tarefas da VNF. A decisão sobre o redirecionamento dos fluxos e o eventual balanceamento de carga entre os comutadores de função é o foco deste artigo, podendo ser aleatória ou baseada no conhecimento disponível acerca dos fluxos e dos recursos dos comutadores. Esses são aspectos centrais deste trabalho e serão discutidos em detalhes na Seção 4 a seguir.

4. Arcabouço para balanceamento dinâmico de carga

O balanceamento de carga tem por objetivo aumentar a capacidade da função virtual de rede através da otimização no uso dos recursos dos comutadores internos da VNF, evitando sobrecargas e bloqueio de fluxos na rede. A Figura 2 apresenta o funcionamento geral do balanceador de carga proposto neste trabalho para o cenário apresentado na Figura 1. Inicialmente, quando um novo fluxo chega nos comutadores de redirecionamento UL ou DL, uma mensagem *Packet-In* do OpenFlow é enviada ao controlador. Por sua vez, o controlador decide, com base na política em vigor, qual comutador de função (*hardware* ou *software*) receberá a regra para processar os pacotes deste fluxo. Neste trabalho, nós abordamos especificamente duas políticas para este fim, denominadas por política estática e política dinâmica. Essas políticas serão detalhadas nas subseções seguintes.

Uma vez escolhido o comutador de função, de acordo com a política em vigor, o controlador verifica se o mesmo possui espaço em sua tabela de fluxos para receber a nova regra. Em caso positivo, o controlador instala nesse comutador a regra de processamento da função virtual (e.g., uma regra para monitorar a vazão do fluxo) e instala nos comutadores UL e DL as regras específicas para redirecionar este fluxo para o comutador de função escolhido. Assim, fica claro que a política de balanceamento de carga adotada neste trabalho funciona por fluxo, garantindo que todos os pacotes de um mesmo fluxo sejam processados pelo mesmo comutador de função. Caso o comutador de função não possua espaço para novas regras em sua tabela de fluxos, o novo fluxo é então bloqueado.

A abordagem reativa do balanceador de carga (em que o primeiro pacote de cada fluxo é enviado para o controlador), permite a adoção da VNF de maneira transparente na rede, dispensando eventual sinalização entre o controlador e as aplicações

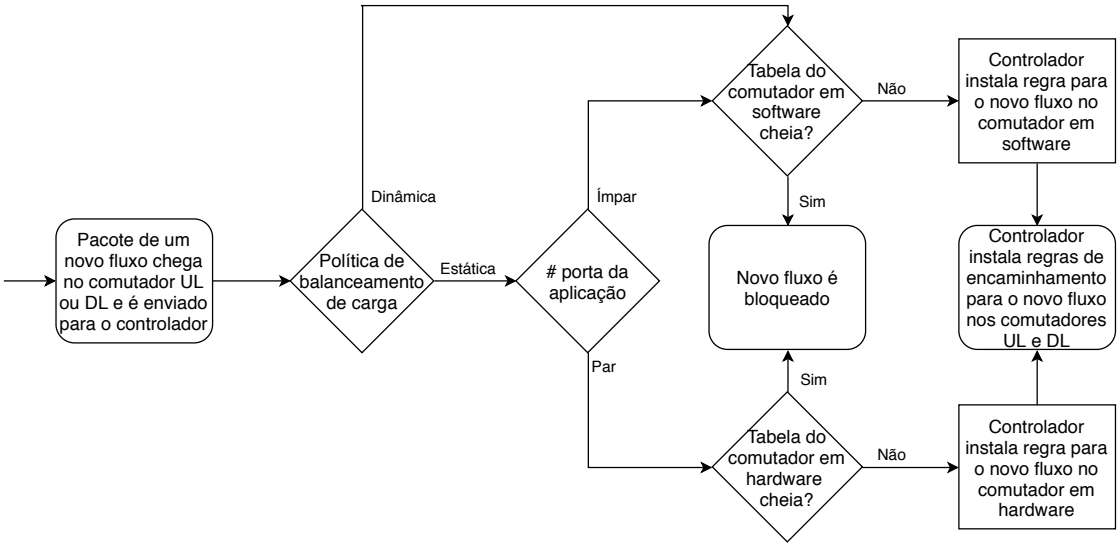

Figura 2. Algoritmo para balanceamento de novos fluxos de dados.

cliente/servidor. Além disso, o controlador OpenFlow e os comutadores de função e redirecionamento são todos locais para cada VNF, então o atraso incorrido pelo envio de um único pacote ao controlador torna-se desprezível.

A seguir, detalhamos as duas políticas de balanceamento propostas neste trabalho. A *política estática* é simples e serve como caso base de comparação. A *política dinâmica*, por outro lado, é uma política mais elaborada e leva em consideração as características dos fluxos e os recursos computacionais disponíveis nos comutadores de função da VNF.

4.1. Política estática

Pela política estática, os fluxos são balanceados de forma simples entre os comutadores de função em *hardware* e em *software*, levando em consideração apenas o número da porta da camada de transporte da aplicação cliente. Conforme ilustrado na Figura 2, fluxos originados em porta de número par são enviados para o comutador em *hardware* enquanto os fluxos com portas ímpares são enviados para o comutador em *software*. Esta política não leva em consideração os recursos computacionais disponíveis nos comutadores de função, apenas distribui o total de fluxos igualmente entre os comutadores a longo prazo.

Note que a utilização do número da porta da aplicação para a tomada de decisão não é a ideal. De fato, na prática, tal decisão não possui escalabilidade, demandando a instalação de inúmeras regras nos comutadores de redirecionamento UL e DL. Porém, essa política serve como caso base de comparação, e foi adotada neste trabalho apenas por conta da limitação de um único par de cliente/servidor no cenário de experimentação (detalhes na Seção 5.1). Em cenários com mais clientes e servidores, é possível realizar as decisões baseadas no número do IP do cliente, fazendo casamento com máscaras, o que reduziria consideravelmente o número de regras nos comutadores UL e DL.

4.2. Política dinâmica

A política dinâmica leva em consideração os recursos disponíveis nos elementos da função virtual. Ela tem como objetivo aumentar o desempenho da VNF, explorando melhor o tamanho da tabela de fluxos e a capacidade de processamento dos comutadores de função em *hardware* e em *software*. Conforme ilustrado na Figura 2, quando um novo

fluxo chega à rede, o controlador escolhe o comutador de função em *software* para receber a regra da função virtual. Neste caso, uma única regra instalada nos comutadores UL e DL é suficiente para redirecionar todos os fluxos para o comutador em *software*.

Comutadores em *software* geralmente possuem tabelas de fluxos grande o suficiente para acomodar todos os fluxos da VNF, eliminando os bloqueios por falta de espaço. Apesar desta vantagem, como citado na Seção 3, esses comutadores apresentam um atraso maior no encaminhamento de pacotes, além de uma capacidade total de processamento menor, quando comparados aos comutadores em *hardware*. Como consequência, comutadores em *software* podem atingir o limite da sua capacidade de processamento rapidamente, o que pode resultar em perdas excessivas de pacotes.

Para lidar com essa limitação, a Figura 3 ilustra o algoritmo de balanceamento de carga dinâmico para a realocação de fluxos ativos adotado neste trabalho. Periodicamente o controlador verifica a disponibilidade de recursos (ocupação da tabela de fluxos) no comutador de função em *hardware*. Havendo espaço para novas regras, o controlador constrói uma lista das regras ativas atualmente instaladas no comutador de função em *software* e as ordena de maneira decrescente em relação à vazão média individual de cada um dos respectivos fluxos. Em seguida, o controlador move as regras correspondentes aos fluxos de maior vazão do comutador em *software* para o comutador em *hardware*, de modo a ocupar todo o espaço disponível no comutador em *hardware*.

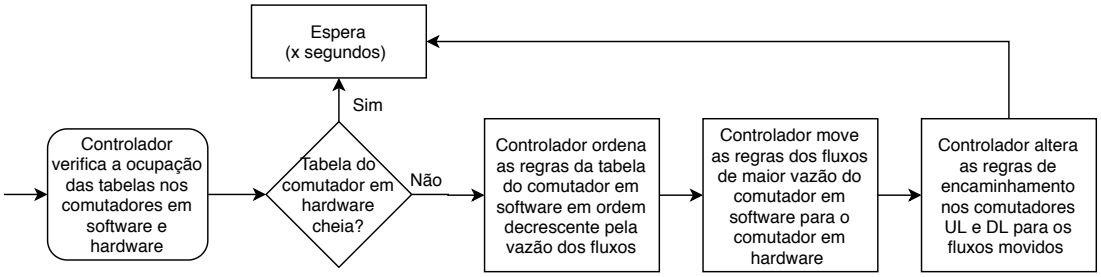

Figura 3. Algoritmo para balanceamento dinâmico de fluxos ativos.

A transferência dinâmica de cada fluxo é feita através da instalação no comutador de função em *hardware* de uma regra OpenFlow exatamente igual à existente no comutador de função em *software*. Na sequência, é necessário que regras específicas sejam instaladas nos comutadores de redirecionamento UL e DL para encaminhar os pacotes deste fluxo para o comutador de função em *hardware*. Após a instalação da nova regra e atualização dos comutadores de redirecionamento, a regra antiga pode ser finalmente removida do comutador de função em *software*. A ordem de execução dessas operações é importante, pois minimiza as perdas de pacotes durante a transição.

Em suma, a política dinâmica explora a maior capacidade de processamento do comutador em *hardware*, concentrando nele os fluxos de maior vazão mas sem extrapolar a capacidade de sua restrita tabela de fluxos. Como consequência, apenas os fluxos com menor vazão individual continuarão instalados no comutador em *software*, reduzindo sua demanda agregada por processamento e reduzindo eventuais perdas de pacotes. Observe que a política dinâmica não é direcionada para atender as necessidades individuais de QoS dos fluxos. O objetivo é aumentar a capacidade total da VNF, alocando os fluxos internamente de acordo com os recursos disponíveis nos comutadores de função.

5. Experimentos e resultados

Nesta seção apresentamos as configurações e a metodologia dos experimentos conduzidos (Seção 5.1), seguido da apresentação e discussão dos resultados (Seção 5.2). Os códigos do experimento e os arquivos com os resultados estão disponíveis para consulta em `https://github.com/pedrobellotti/artigoWGRS2020`.

5.1. Configurações do cenário e metodologia de avaliação

Para avaliarmos a topologia apresentada na Figura 1, utilizamos um ambiente de testes com 7 computadores dedicados. Para representar os comutadores internos da VNF foram utilizadas 4 máquinas independentes, cada uma executando uma instância do comutador virtual de código aberto *Open vSwitch*. Um quinto computador atua como controlador OpenFlow executando o *software POX*[2]. Por fim, um par de computadores foi utilizado como cliente e servidor, executando o *software Iperf*[3] para gerar tráfego na rede. Nesta topologia, todos os enlaces de comunicação entre as máquinas são *Gigabit Ethernet*.

Neste ambiente de testes, a heterogeneidade entre os comutadores em *hardware* (redirecionamento e função) e o comutador de função em *software* deve-se à diferença nas configurações dos equipamentos utilizados. Para os comutadores em *hardware* foram adotadas três máquinas com processador Intel Core i3-540, 2GB de memória e sistema operacional Ubuntu 18.04 LTS. Para o comutador de função em *software* foi adotada uma máquina antiga, com processador Intel Celeron D, 512 MB de memória e sistema operacional Lubuntu 16.04 LTS. Essa discrepância nas configurações resultou em uma capacidade máxima de processamento 2,5 vezes maior no comutador em *hardware* quando comparado ao comutador em *software*. Note que, ao se utilizar comutadores reais, essa diferença pode ser ainda maior. Para refletir também a diferença no tamanho das tabelas de fluxo, ajustamos o comutador de função em *hardware* para suportar no máximo 200 regras simultâneas, e deixamos a tabela do comutador de função em *software* com seu tamanho padrão. Tais valores foram definidos de acordo com as dimensões do experimento, de modo a analisar o impacto do tamanho das tabelas no bloqueio dos tráfegos.

Neste ambiente de testes não é possível incluir mais clientes/servidores, visto que todas as interfaces de redes nos comutadores UL e DL já estão em uso. Tal restrição implica em uma vazão teórica máxima de 1 Gbps em cada direção. Entretanto, este não é fator limitante para os experimentos, já que a capacidade reduzida de processamento do comutador em *software* (≈ 400 Mbps) e o tamanho reduzido da tabela de fluxos do comutador em *hardware* (200 regras) são, de fato, os gargalos deste cenário.

O *software Iperf* foi configurado com os parâmetros da Tabela 1 para gerar fluxos de dados UDP unidirecionais com duração uniforme e vazão exponencial. A taxa de chegada de novos fluxos segue uma distribuição de *Poisson*, onde três valores distintos de λ foram adotados para definir intensidades de carga *baixa*, *média* e *alta* nos experimentos. As diferentes cargas de trabalho foram usadas para avaliar o desempenho das duas políticas de balanceamento em situações adversas. Por fim, os atrasos de pacotes nos comutadores de função foram medidos utilizando uma versão modificada do software *UDPPing*[4], com sondas em paralelo ao tráfego gerado.

[2] `https://noxrepo.github.io/pox-doc/html/`
[3] `https://iperf.fr`
[4] `https://github.com/wangyu-/UDPping`

Tabela 1. Parâmetros utilizados na geração dos tráfegos.

	Distribuição	Parâmetros		
		Baixa	**Média**	**Alta**
Taxa de chegada de fluxos	*Poisson*	$\lambda = 3,\overline{3}$	$\lambda = 5$	$\lambda = 10$
Duração do fluxo	Uniforme	min = 5 s e max = 100 s		
Vazão do fluxo	Exponencial	média = 1024 Kbps		

Para cada combinação de *política de balanceamento* $\times$ *carga de trabalho*, foram realizadas 15 execuções de testes com sementes distintas para a geração dos parâmetros aleatórios, de modo a obter resultados diferentes em cada execução. Os resultados apresentados são a média das 15 execuções, considerando as análises no período estável do experimento, com um nível de confiança de 95%.

5.2. Análise dos resultados

Inicialmente, avaliamos o percentual de bloqueios de fluxos, considerando diferentes cargas de trabalho. Os bloqueios acontecem quando o comutador de função escolhido pela política de balanceamento não tem recursos para acomodar o novo fluxo. No cenário avaliado, isso acontece quando não há mais espaço disponível nas tabelas de fluxo do comutador de função em *hardware*. Para avaliar o bloqueio, consideramos apenas o cenário com a política de balanceamento estática. De fato, a política dinâmica não gera bloqueios, visto que novos fluxos são sempre redirecionados para o comutador de função em *software*, que possui tabela de fluxos com tamanho virtualmente grande.

A Figura 4 mostra o percentual de bloqueios ao se utilizar a política de balanceamento estática, em uma rede com tráfego em baixa, média e alta carga. Em baixa carga, o percentual de bloqueio do comutador em *hardware* é negligenciável. Porém, em uma rede com carga média de trabalho, pouco mais de 3% dos fluxos são bloqueados. Esse número pode superar 16% quando a rede está em alta carga de trabalho. Esse valor mostra a importância da política dinâmica e de políticas que exploram o conhecimento dos recursos computacionais disponíveis nos elementos de rede.

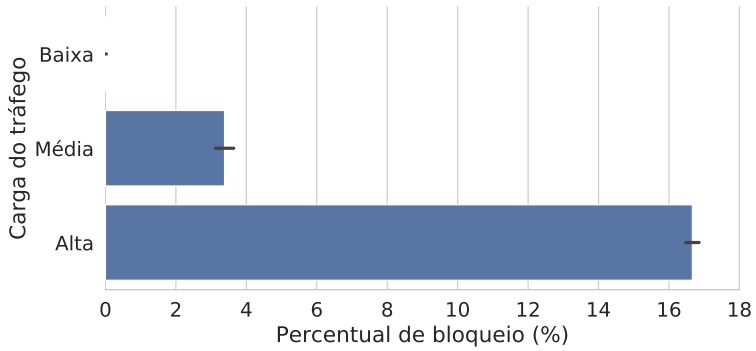

Figura 4. Percentual de bloqueio no comutador de função em *hardware*.

A vazão média da rede depende da demanda dos fluxos e, também, da política adotada pela VNF no balanceamento da carga. A Figura 5 apresenta a vazão média da

rede para as diferentes configurações dos experimentos realizados. Para as cargas baixa e média, não há diferença estatística significativa entre as políticas. Claramente, há recursos computacionais disponíveis nesses cenários, onde a política de balanceamento tem pouca influência. A necessidade do balanceamento de carga e a importância da política apropriada fica mais evidente com a alta carga de trabalho. A diferença média de vazão entre a política estática e a dinâmica, nesse cenário, é aproximadamente 27%.

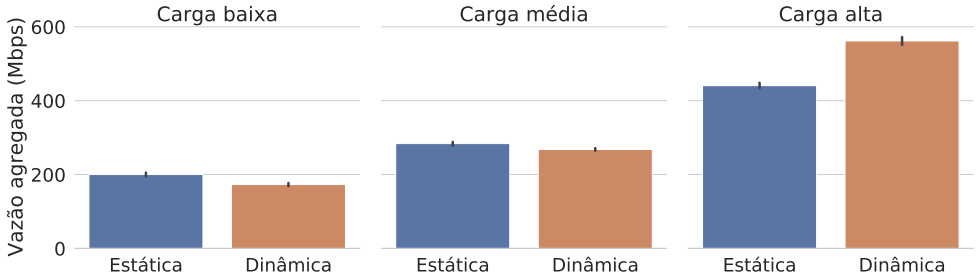

Figura 5. Vazão média da rede por política no intervalo estável.

A Figura 6 detalha a vazão da rede e mostra a carga atribuída a cada comutador de função. Para as cargas baixa e média, quando os comutadores de função têm recursos disponíveis em excesso, a política estática divide a demanda da rede entre os dois comutadores de modo estatisticamente igual. Cada comutador fica com uma vazão próxima a 100 Mbps. Nesses dois cenários, a política dinâmica apresentou maior vazão no comutador de função em *hardware*. Ou seja, quando há recursos disponíveis, a política dinâmica concentra a maioria dos fluxos no comutador em *hardware*, privilegiando o seu melhor desempenho no processamento de pacotes. O comutador em *hardware*, para as cargas baixa e média, ficou responsável por ≈ 140 e ≈ 200 Mbps de vazão, respectivamente.

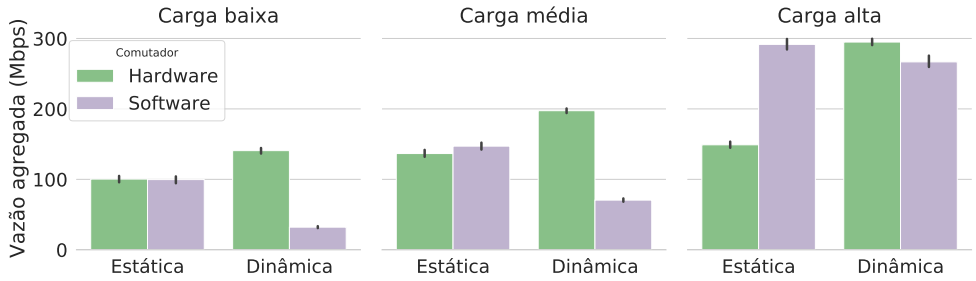

Figura 6. Vazão média por comutador no intervalo estável.

Por outro lado, quando os recursos de rede estão escassos (carga alta), a política estática continua atribuindo fluxos igualmente aos comutadores de função. Como o comutador em *hardware* tem limitação no tamanho da tabela, ele começa a negar serviço e eleva o percentual de bloqueio (Figura 4). Com isso, a vazão no comutador em *hardware* fica em ≈ 150 Mbps, o que é bem menor do que a vazão do comutador em *software*, que excede 290 Mbps. A política dinâmica evita os bloqueios e a disputa de recursos no comutador em *hardware*. Ela encaminha apenas os fluxos com maior vazão para este comutador, que tem maior poder de processamento, respeitando assim o seu limite no tamanho da tabela. Dessa forma, mesmo com uma quantidade menor de fluxos, o comutador em *hardware* alcança uma vazão de rede tão alta quanto o comutador em *software*.

A Figura 7 corrobora com a discussão anterior ao mostrar o número médio de regras instaladas nos comutadores de função. A distribuição das regras, em cargas baixa e média, é igualitária entre os comutadores pela política estática (assim como acontece com a vazão, já apresentada na Figura 6). Por outro lado, nessas cargas (média e baixa), a política dinâmica já explora prioritariamente o comutador em *hardware*. Mesmo com o número total de regras limitado em 200 neste comutador, a política dinâmica sempre tenta redirecionar os maiores fluxos para ele. Em carga alta, a limitação na tabela de fluxos do comutador em *hardware* torna-se crítica para ambas as políticas. Nesse caso, porém, os efeitos na vazão são mais evidentes, como demonstrado anteriormente.

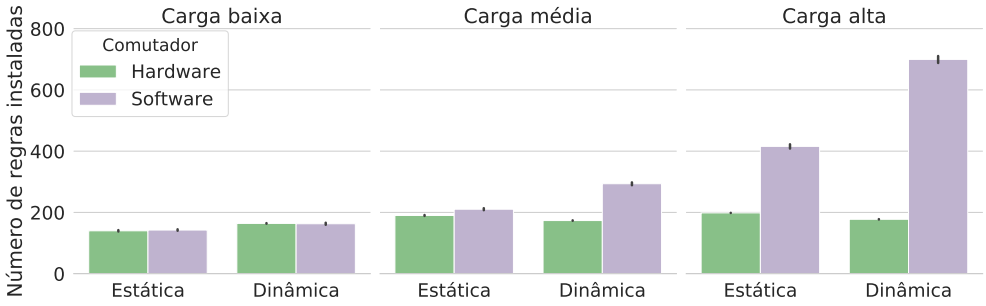

Figura 7. Número médio de regras instaladas por comutador no intervalo estável.

Notadamente, a política dinâmica apresenta uma vantagem com relação ao melhor aproveitamento dos recursos da VNF, resultando no aumento de sua vazão. Analisamos também um outro indicador de qualidade: a latência fim-a-fim dos pacotes. A Figura 8 apresenta a função de distribuição cumulativa dos atrasos para os pacotes de sondagem em ambos os comutadores, apenas no cenário de maior demanda (i.e., carga alta). Por essa figura, a política estática apresenta uma leve vantagem sobre a política dinâmica, apenas para os fluxos direcionados ao comutador de função em *hardware*. De fato, para 50% dos pacotes de sondagem, a política estática apresenta até 0,6 ms de atraso neste comutador. Já na política dinâmica, para esse mesmo percentual, o atraso é de até 0,75 ms.

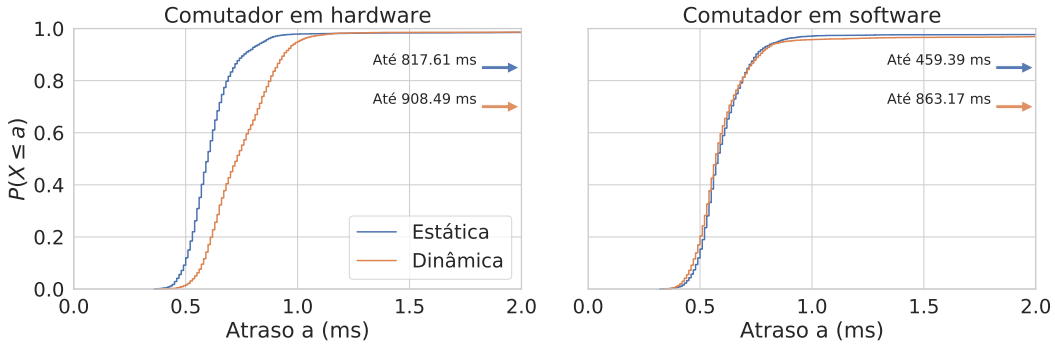

Figura 8. Atraso fim-a-fim no intervalo estável com carga alta (pior caso).

A diferença entre esses atrasos é consequência da natureza das políticas. Pela política estática, o comutador de função em *hardware* bloqueia alguns fluxos. Logo, a vazão agregada é menor neste comutador, as filas ficam pequenas e o atraso diminui. Por outro lado, pela política dinâmica, apenas os maiores tráfegos são atribuídos ao comutador

em *hardware*. Por conta disso, a vazão agregada aumenta e as filas ficam maiores, impactando no atraso. Além disso, existe um atraso decorrente do processamento de pacotes pelo *Open vSwitch* no comutador de função em *hardware* que não existiria em um comutador físico. Portanto, os atrasos indicados para este comutador na Figura 8 representam o pior caso. É esperado que, em uma topologia com um comutador em *hardware* real, os atrasos deste comutador sejam menores do que os atrasos do comutador em *software*.

Por fim, observamos um número muito pequeno de pacotes sendo recebidos fora de ordem nas duas políticas, como apresentado na Tabela 2. Isso demonstra que a qualidade da rede, neste quesito, é semelhante para ambas as políticas. Novamente, a pequena diferença existe por conta da natureza das políticas. Mesmo com a remoção das regras antigas sendo feita após a instalação de novas regras para evitar perda de pacotes na política dinâmica, pacotes antigos já enviados pelo comutador em *software* podem chegar depois dos pacotes novos enviados pelo comutador em *hardware*, devido a diferença no tempo de processamento dos comutadores. A consequência natural é um pequeno aumento no número de pacotes recebidos fora de ordem. Porém, mesmo no cenário de carga alta, as perdas de pacote são desprezíveis para todos os fluxos em ambas as políticas.

Tabela 2. Percentual de pacotes fora de ordem por fluxo de dados.

	Política estática			**Política dinâmica**		
	Carga baixa	**Carga média**	**Carga alta**	**Carga baixa**	**Carga média**	**Carga alta**
Mínimo	0	0	0	0	0	0
Mediana	0	0	0	0	0	0
Média	0	4.88×10^{-8}	4.62×10^{-7}	0	0	1.28×10^{-5}
Máximo	0	3.95×10^{-4}	1.01×10^{-3}	0	0	2.88×10^{-2}

6. Conclusões e trabalhos futuros

Este trabalho apresenta uma nova política dinâmica de balanceamento de carga, de modo a otimizar a capacidade de processamento de uma VNF implementada sobre comutadores OpenFlow heterogêneos, preservando a qualidade do serviço. Validamos a política proposta por meio de experimentação em um pequeno, mas realista, ambiente de testes. Os resultados obtidos foram contrastados com outra política mais simples, também implementada para servir como base de comparação. Observamos que a política dinâmica de balanceamento de carga consegue atingir uma vazão $\approx 27\%$ maior que a abordagem simples, com percentual de bloqueio nulo e melhor aproveitamento dos recursos disponíveis nos comutadores. Dessa forma, toda a demanda da rede foi atendida com qualidade, atingindo os objetivos inicialmente propostos neste trabalho.

Os resultados alcançados para esta pequena topologia mostram a viabilidade da proposta. Como trabalhos futuros, pretendemos avaliar estas e outras políticas de balanceamento de carga em um cenário maior através de simulações. Pretendemos também propor um novo mecanismo capaz de realizar a instanciação de comutadores de função em *software* para aumentar a capacidade da VNF dinamicamente, apenas quando necessário.

Agradecimentos

Os autores agradecem o apoio da CAPES, CNPq e FAPEMIG.

Referências

An, X., Kiess, W., e Perez-Caparros, D. (2014). Virtualization of cellular network EPC gateways based on a scalable SDN architecture. Em *IEEE GLOBECOM*.

Carpio, F., Dhahri, S., e Jukan, A. (2017). VNF placement with replication for load balancing in NFV networks. Em *IEEE ICC*.

Chaves, L. J., Eichemberger, V. M., Garcia, I. C., e Madeira, E. R. M. (2015). Integrating OpenFlow to LTE: some issues toward Software-Defined Mobile Networks. Em *IFIP NTMS*.

Chaves, L. J., Garcia, I. C., e Madeira, E. R. M. (2017). An adaptive mechanism for LTE P-GW virtualization using SDN and NFV. Em *IEEE CNSM*.

Costa, L. C., Vieira, A. B., de Britto e Silva, E., Macedo, D. F., Gomes, G., Correia, L. H. A., e Vieira, L. F. M. (2017). Performance evaluation of OpenFlow data planes. Em *IFIP/IEEE IM*.

ETSI NFV (2012). Network functions virtualisation: An introduction, benefits, enablers, challenges & call for action. White Paper.

Jamali, S., Badirzadeh, A., e Siapoush, M. S. (2019). On the use of the genetic programming for balanced load distribution in software-defined networks. *Digital Communications and Networks*. DOI: 10.1016/j.dcan.2019.10.002.

Kaur, S., Kaur, K., e Gupta, V. (2017). Implementing OpenFlow based distributed firewall. Em *IEEE InCITe*.

Kreutz, D., Ramos, F. M. V., Verissimo, P., Rothenberg, C. E., Azodolmolky, S., e Uhlig, S. (2015). Software-defined networking: A comprehensive survey. *Proceedings of the IEEE*, 103(1):14–76.

Laghrissi, A. e Taleb, T. (2019). A survey on the placement of virtual resources and virtual network functions. *IEEE Communications Surveys & Tutorials*, 21(2):1409–1434.

Nguyen, X.-N., Saucez, D., Barakat, C., e Turletti, T. (2016). Rules placement problem in OpenFlow networks: A survey. *IEEE Communications Surveys & Tutorials*, 18(2):1273–1286.

Open Networking Foundation (2012). Software-Defined Networking: The new norms for networks. ONF White Paper.

Rodrigues, C. P., Costa, L. C., Vieira, M. A. M., Vieira, L. F. M., Macedo, D. F., e Vieira, A. B. (2015). Avaliação de balanceamento de carga web em redes definidas por software. Em *SBRC*.

Wang, C., Spatscheck, O., Gopalakrishnan, V., Xu, Y., e Applegate, D. (2016). Toward high-performance and scalable network functions virtualization. *IEEE Internet Computing*, 20(6):10–20.

Yi, B., Wang, X., Li, K., k. Das, S., e Huang, M. (2018). A comprehensive survey of network function virtualization. *Computer Networks*, 133:212–262.

Zhang, J., Yu, F. R., Wang, S., Huang, T., Liu, Z., e Liu, Y. (2018). Load balancing in data center networks: A survey. *IEEE Communications Surveys & Tutorials*, 20(3):2324–2352.