Avaliação de Balanceamento de Carga Web em Redes Definidas por Software

Cristiane P. Rodrigues[1], Leonardo C. Costa[1]
Marcos Augusto M. Vieira[2], Luiz Filipe M. Vieira[2], Daniel F. Macedo[2], Alex B. Vieira[1]

[1]Departamento de Ciência da Computação
Universidade Federal de Juiz de Fora (UFJF) – Juiz de Fora, MG – Brasil

[2]Departamento de Ciência da Computação
Universidade Federal de Minas Gerais (UFMG) – Belo Horizonte, MG – Brasil

{cristiane.rodrigues,leonardocosta}@ice.ufjf.br, alex.borges@ufjf.edu.br

{mmvieira,lfvieira,damacedo}@dcc.ufmg.br

Abstract. *The decoupling between the control and data planes, the main idea from Software Defined Networks (SDNs), opens a new horizon to perform traditional tasks such as load balancing. In this sense, we present an SDN architecture in which different policies may be used to balance loads on Web servers. In this work we simulate system behavior when using three policies: random, round-robin style, and one that consider the Web server's workload. Our results show that for traditional Web loads, where there are several small requests, the policies do not differ significantly. For loads equivalent to Web 2.0, such as pictures and large files, the policy that considers the load presents an improvement of up to 11% in response time. Surprisingly, for heavy loads, which is becoming the most common load, the load-based policy has the worst result. In this case, the reaction time required by the Web application and SDN is much lower than the data collection time on Web servers.*

Resumo. *A separação entre camadas de controle e dados, ideia base de Redes Definidas por Software, abre um novo horizonte para realizar tarefas tradicionais como o balanceamento de carga. Nesse sentido, apresentamos uma arquitetura SDN na qual diferentes políticas podem ser utilizadas para realizar balanceamento de carga em servidores Web. Neste trabalho simulamos o comportamento do sistema ao se utilizar uma política aleatória, uma estilo round-robin e uma que leva em consideração a carga de trabalho dos servidores Web. Nossos resultados mostram que para cargas tradicionais Web, onde há várias requisições de pequeno tamanho para cada uma delas, as políticas não apresentam diferenças significativas. Para cargas equivalentes a Web 2.0, como fotos e arquivos grandes, a política que leva em consideração a carga apresenta uma melhora de até 11% no tempo de resposta. Surpreendentemente, para cargas pesadas, cada vez mais comuns, a política que leva em consideração a carga apresenta o pior resultado entre as três políticas consideradas. Nesse caso, o tempo de reação requisitado pela aplicação Web e SDN é muito inferior ao tempo de coleta de dados de carga nos servidores Web.*

1. Introdução

O balanceamento de carga tornou-se um mecanismo importante para serviços em redes de computadores. Com a inclusão de novas tecnologias, a Internet vem se tornando algo complexo, onde serviços devem ser capazes de lidar com milhares de requisições simultaneamente. Para suportar grandes demandas e manter um tempo de resposta suficientemente rápido para os clientes, é necessário que os recursos da rede sejam bem utilizados. Resumindo, cada servidor ou equipamento deve processar uma quantidade de trabalho compatível com sua capacidade.

Atualmente, uma determinada configuração de serviços em redes consiste em um balanceador de carga que recebe requisições recém chegadas e as distribui para múltiplos servidores pertencentes a uma rede. Comumente, esses balanceadores são sistemas (*hardware*/*software*) altamente especializados para determinado tipo de serviço. Por serem altamente especializados, esses balanceadores são equipamentos muito caros (facilmente superando 50 mil dólares), possuem regras rígidas de funcionamento e necessitam de administradores capacitados para sua operação [Uppal and Brandon 2010].

Redes Definidas por Software (SDN) são um novo paradigma que vem mudando a forma de como criar, modificar e gerenciar redes de computadores [Feamster et al. 2014]. Essa nova abordagem tem atraído a atenção tanto da indústria quanto da comunidade acadêmica. Com ela, diversos serviços de rede estão sendo repensados, de forma a torná-los mais flexíveis. O balanceamento de carga pode ser implementado empregando conceitos de SDN.

A separação entre camadas de controle e dados, ideia base de Redes Definidas por Software, abre um novo horizonte para realizar tarefas tradicionais como o balanceamento de carga. Por exemplo, podemos criar um balanceador de carga onde as regras de balanceamento são definidas de acordo com as várias aplicações existentes. Por utilizar regras adaptadas a cada aplicação, a distribuição de tráfego poderá ser mais eficiente do que empregarmos um balanceador proprietário. Além disso, por empregar *hardware commodity*, a solução SDN poderá ter um custo menor.

Na literatura, encontramos alguns esforços na direção de propor balanceamento de carga utilizando Redes Definidas por Software [Sherwood et al. 2009, Handigol et al. 2009, Uppal and Brandon 2010, Wang et al. 2011, Ragalatha et al. 2013, Zhou et al. 2014]. Os trabalhos citados propõem o balanceamento de carga através da construção de uma arquitetura SDN com o padrão OpenFlow, porém nem todos eles avaliam o comportamento do balanceador proposto. E os poucos que realizam tal avaliação, o fazem em ambiente muito específico, sem se preocupar com o perfil de carga a qual serão submetidos os serviços a serem balanceados.

Considerando esse contexto, nós propomos, neste trabalho, um arcabouço de balanceamento de carga em SDN no qual diferentes políticas podem ser utilizadas para realizar tal tarefa. Nós avaliamos o funcionamento desse arcabouço de balanceamento de carga utilizando um serviço *Web* como foco, e mostramos os benefícios alcançados e as limitações quando o balanceador tem que lidar com diferentes perfis de carga.

De uma forma mais detalhada, avaliamos o comportamento do sistema ao se utilizar três políticas de balanceamento diferentes: uma política aleatória, uma estilo *round-robin* e uma que leva em consideração a carga de trabalho dos servidores *Web*. Nós

também variamos a carga de trabalho, refletindo três cenários comuns a servidores *Web*.

Nossos resultados mostram que, para cargas tradicionais *Web* onde há várias requisições de pequeno tamanho, as políticas não apresentam diferenças significativas entre si. Nesse caso, as requisições são pequenas e de rápida resposta, e assim não há tempo suficiente para que políticas sofisticadas de balanceamento de carga convirjam para um resultado melhor que uma abordagem simples, como a *round-robin*. Para cargas equivalentes a *Web* 2.0, como *upload/download* de fotos e arquivos grandes, a política que leva em consideração a carga dos servidores *Web* apresenta uma melhora de até 11% no tempo de resposta, quando comparada às demais políticas. Surpreendentemente, para cargas *Web* pesadas, cada vez mais comuns na Internet, a política que leva em consideração a carga apresenta o pior resultado entre as três políticas consideradas. Nesse caso, o tempo de reação requisitado pela aplicação *Web* é muito inferior ao tempo de atualização dos dados de carga nos servidores *Web*. Assim, rajadas de requisições são direcionadas a um mesmo servidor que, erroneamente, foi interpretado como o menos sobrecarregado pelo controlador SDN.

O restante desse artigo está organizado como segue: na Seção 2 mostramos o estado da arte com relação às propostas de balanceamento de carga em SDN. Na seção 3 apresentamos nossa proposta e discutimos as decisões de projeto do nosso arcabouço de balanceamento de carga. As seções 4 e 5 mostram nossa metodologia de avaliação e os resultados obtidos com nosso balanceador de carga. Finalmente, a seção 6 apresenta nossas conclusões e trabalhos futuros.

2. Trabalhos Relacionados

O balanceamento em redes de computadores é uma técnica usada para distribuir a carga de trabalho entre vários enlaces ou computadores [Uppal and Brandon 2010]. Desta forma, mais tarefas poderão ser realizadas na mesma quantidade de tempo e, em geral, todas as requisições poderão ser respondidas em menos tempo. Além disso, o balanceamento de carga traz vantagens como o aumento da escalabilidade, pois poderá existir flexibilidade em adicionar mais recursos à rede. Isso acontece de forma rápida e transparente aos usuários finais. Traz ainda como vantagem o aumento do desempenho pela utilização dos recursos de forma inteligente e maior disponibilidade, já que se um dos servidores falhar, a carga poderá ser redistribuída aos outros servidores existentes sem comprometer o processamento das requisições [Korner and Kao 2012].

Balanceamento de carga é um tema de pesquisa atrativo, uma vez que um balanceador dedicado pode se tornar um ponto de falha e congestionamento [Wang et al. 2011]. Além disso, algumas das soluções existentes são específicas para determinados serviços, o que as tornam pouco flexíveis com relação à visão da rede como um todo. Atualmente, diversos trabalhos em balanceamento de carga têm empregado os conceitos de SDN.

Grande parte da atenção em SDN tem sido voltada para o padrão OpenFlow [Guedes et al. 2012]. O OpenFlow é um dos principais protocolos relacionados a SDN e, de fato, é o que possibilita a implementação desse novo paradigma. Com a padronização do OpenFlow, vendedores e operadores de rede começam a dar uma importância maior a ele, reforçando por consequência o paradigma SDN. Por exemplo, em 2011, empresas como Facebook, Yahoo, Google e Microsoft formaram a Open Network-

ing Foundation (ONF). Essa organização tem como objetivo promover a tecnologia OpenFlow, trazendo para o mercado normas e soluções SDN [ONF 2014].

Alguns trabalhos recentes em balanceamento de carga apresentam propostas baseadas em SDN, entretanto não apresentam uma avaliação aprofundada das suas propostas [Handigol et al. 2009, Uppal and Brandon 2010, Wang et al. 2011, Ragalatha et al. 2013, Zhou et al. 2014]. De fato, alguns destes são trabalhos em desenvolvimento. Praticamente todos compartilham de semelhanças nas arquiteturas propostas: usam padrão OpenFlow, controlador baseado em NOX, e um esquema de reescrita tanto de cabeçalho de pacotes quanto da tabela de fluxos do switch OpenFlow.

Em alguns casos, o balanceamento de carga é efeito colateral da flexibilidade em se adicionar recursos de forma transparente à rede [Handigol et al. 2009]. Nesse trabalho, os autores desenvolveram um esquema que monitora a realidade da rede (recursos disponíveis) e distribui a carga de trabalho entre esses serviços. Para realizar essa distribuição, existem módulos específicos que monitoram os servidores existentes e seu estado. Esses módulos ativamente modificam as regras no controlador para realizar a distribuição de carga.

Outros trabalhos procuram realizar um balanceamento dinâmico de carga. Por exemplo, Ragalatha et al. [Ragalatha et al. 2013] implementam um balanceador onde a carga de trabalho é calculada e distribuída entre os servidores existentes em tempo de execução. Wang et al. [Wang et al. 2011] fazem o balanceamento dinâmico instalando regras curingas nos *switches*. Eles propõem algoritmos para computar essas regras e ajustá-las de acordo com mudanças no sistema. Isso ocorre sem quebras nas conexões já existentes.

Korner et al. [Korner and Kao 2012] também se preocupam com balanceamento de carga dinâmico. Porém, os autores tratam a rede de forma diferente dos trabalhos discutidos anteriormente. Nesse caso, o controlador NOX é utilizado juntamente com o FlowVisor, que divide os recursos de rede e utiliza um controlador diferente para cada divisão. Assim, cada fatia da rede pode ter seu próprio balanceamento, independente das demais. Isso provê maneiras de alocar recursos e balancear diferentes serviços, independentes um do outro.

Apesar dos trabalhos apontados mostrarem estratégias diversas para realizar balanceamento de carga em SDN, não ficam claros os benefícios alcançados. Alguns deles não realizam uma avaliação consistente, pois utilizam cargas simplistas quando comparados a cenários mais próximos à realidade, como os descritos em [Jeon et al. 2012, Polfliet et al. 2012, Bovenzi et al. 2011]. Dessa forma, esses trabalhos apresentam resultados que podem não ser conclusivos.

Assim, destacamos as seguintes contribuições de nosso trabalho: primeiro, apresentamos uma maneira simples e flexível para realizar balanceamento de carga em SDN. Nós discutimos a sua implementação e mostramos soluções para capturar informações da rede, sem quebrar a premissa de divisão entre dados e controle. Segundo, nós apresentamos uma avaliação do mecanismo implementado. Nós mostramos cenários próximos à realidade e mostramos as limitações existentes para balanceamento de carga em SDN.

3. O mecanismo de balanceamento de carga proposto

A proposta do presente trabalho é a criação e avaliação do balanceamento de carga em uma rede SDN com OpenFlow. Nossa proposta se concentra no balanceamento de cargas *Web*. Porém, a arquitetura proposta é flexível, permitindo a coexistência de serviços e políticas de balanceamento, e podendo ser empregada em outros serviços além da *Web*.

A Figura 1 apresenta a arquitetura proposta nesse trabalho. Os servidores hospedam os servidores *Web* da arquitetura. Esses servidores se ligam ao mundo externo por um único endereço IP, ao qual também iremos nos referir como endereço IP externo do serviço. Quando um cliente faz uma nova requisição para o endereço IP externo do serviço, a requisição é encaminhada para um *switch* OpenFlow. Esse *switch* tem regras específicas e encaminha a carga a um dos servidores do sistema, balanceando assim a carga total de serviço. As regras de balanceamento que o *switch* executa são definidas pelo controlador OpenFlow.

Mais detalhadamente, o funcionamento do sistema ocorre como descrito a seguir: quando o primeiro pacote de uma nova solicitação chega ao *switch*, ele é repassado ao controlador. O controlador, após a escolha de qual servidor do conjunto de servidores que respondem ao serviço irá atender a requisição, cria uma entrada na tabela de fluxo do *switch*, com as ações de encaminhar o pacote para a porta de saída específica. Além disso, é feita a reescrita dos campos de endereços MAC e IP de destino, substituindo os endereços MAC e IP do endereço de serviço pelos endereços MAC e IP do servidor escolhido. Nesse momento, o controlador também instala regras para permitir o fluxo de retorno do servidor para o cliente. Neste caso serão modificados os campos de origem do pacote. Para a resposta à requisição, os endereços IP e MAC de origem do pacote, que são os do servidor, serão substituídos pelos endereços IP externos de serviço. Com isso, o cliente não sabe para qual servidor foi enviada sua requisição e de qual servidor ele obteve resposta.

Nesse trabalho, utilizamos três políticas para realizar o balanceamento das requisições, a saber: Round-Robin, Aleatória e Baseada na carga. As políticas foram implementadas na linguagem Python. As políticas implementadas para o balanceamento de carga são detalhadas a seguir:

1. **Aleatória:** O controlador escolhe aleatoriamente para qual servidor enviar a solicitação. Essa escolha é uniformemente distribuída entre os servidores existentes.
2. **Round-Robin:** Para cada fluxo novo, um servidor é escolhido alternativamente para receber a solicitação do cliente, a partir do primeiro servidor. Assim, as requisições são divididas uniformemente entre todos os servidores.
3. **Baseada na carga:** Essa política envia a solicitação para o servidor com a menor carga média de CPU. Essa carga média é calculada pelo sistema operacional através da média do último minuto de uso da CPU. A carga é obtida no arquivo de sistema /PROC/LOADAVG de cada servidor Linux. Os servidores enviam as suas cargas para a máquina onde se encontra o controlador, através de comunicação estabelecida via Sockets Python, por uma rede externa à rede SDN.

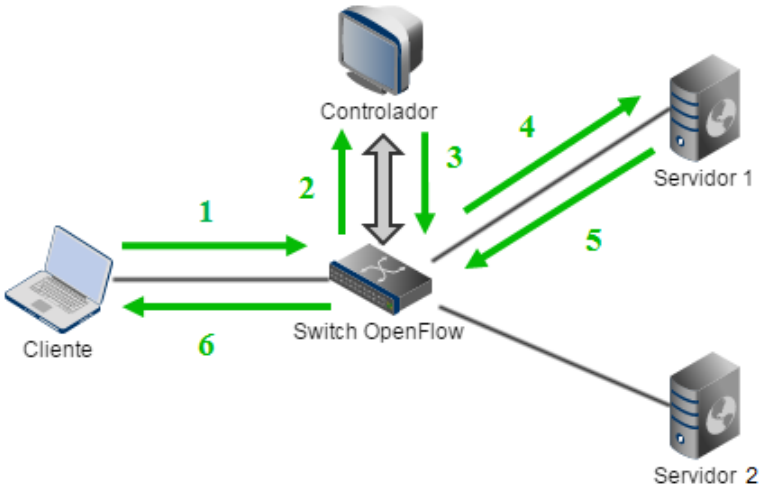

Figura 1. Arquitetura da rede proposta: (1) A requisição do cliente é enviada ao *switch*; (2) Como não há entrada especificada na tabela de fluxos, o primeiro pacote da requisição é repassado ao controlador; (3) O controlador escolhe o servidor de destino, cria uma entrada correspondente na tabela de fluxos e devolve o pacote ao *switch*; (4) A requisição é enviada ao servidor escolhido; (5) A requisição é atendida pelo servidor, que envia a resposta de volta ao *switch*; (6) A resposta da requisição é enviada ao cliente.

4. Metodologia de Avaliação

4.1. Ambiente de testes

A Figura 1 mostra a arquitetura SDN criada para a realização dos testes do esquema de balanceamento de carga proposto. A arquitetura foi criada dentro de uma máquina física, e é composta por máquinas virtuais e um *switch* virtual. Utilizamos duas máquinas virtuais como servidores *Web* de nossa arquitetura. Esses servidores têm duas interfaces de rede cada. Uma das interfaces está conectada à rede definida por *software*.

Para realizar a comunicação direta entre os servidores e controlador, sem violar o princípio básico de separação dos planos de controle e dados existente em Redes Definidas por Software, uma rede externa à SDN foi criada. Essa rede externa foi criada utilizando uma segunda placa de rede contida na máquina física, tornando possível a comunicação direta com os servidores externamente à rede SDN.

Utilizamos o Open vSwitch (OvS) [Pfaff et al. 2009] como elemento de chaveamento de nossa arquitetura. O OvS é um *switch* virtual multi-camadas com suporte OpenFlow e de código aberto, construído para funcionar em ambientes virtualizados. Ele suporta a criação de VLANS e GRE tunneling, além de fornecer conectividade entre as máquinas virtuais e as interfaces físicas. Através da interface física, é criada uma ponte para que possam ser criadas novas interfaces virtuais que se conectarão às máquinas servidoras.

O controlador SDN empregado foi o POX [POX 2014]. Ele foi instalado no mesmo equipamento onde se encontra o *switch* OpenFlow. No controlador foram im-

plementadas as três políticas de escalonamento avaliadas nesse trabalho. Conforme mencionamos, o controlador recebe informações sobre os servidores em uma interface à parte da SDN. Por essa interface, o controlador recebe a carga de CPU de cada servidor.

Os cliente de testes e os dois servidores *Web* foram instanciados como três máquinas virtuais. Essas máquinas estão em execução dentro da máquina física onde se encontra o *switch* OpenFlow, e são gerenciadas através do VirtualBox. Os servidores possuem sistema operacional Linux Ubuntu Server 12.04, com 1 GB de memória. Em cada servidor foi instalado o servidor *Web* Apache versão 2.2. O Apache, que é o servidor *Web* mais utilizado atualmente, foi escolhido por ser de fácil uso e por atender bem aos requisitos da arquitetura. Não houve necessidade de configurações adicionais no Apache. Já a máquina cliente possui sistema operacional Ubuntu 14.04, com 512 MB de memória. Nela foi instalada a ferramenta que foi responsável por enviar as requisições ou cargas: o Httperf [Mosberger and Jin 1998].

4.2. Cargas de trabalho

Para enviar as cargas, a partir da máquina cliente, foi utilizada a ferramenta Httperf. O Httperf é uma ferramenta que tem como propósito medir o desempenho de um servidor *Web*. Ela gera diversas cargas de trabalho HTTP a partir de parâmetros pré-definidos. Ao final de sua execução, o Httperf sumariza as estatísticas gerais de desempenho do servidor avaliado, como o tempo total de conexão, por exemplo.

Os testes foram realizados em três cenários distintos. Em cada um deles utilizamos as três políticas de balanceamento de carga descritas na Seção 3. O objetivo dos testes é obter os tempos médios de conexão, através de experimentos, para diferentes taxas de chegada de requisições por segundo, para a posterior avaliação de desempenho dos algoritmos em cada situação. Definimos um experimento como sendo cada execução do Httperf com uma taxa de chegada pré-definida, que retornará um valor de tempo total de conexão. Esse experimento é executado 10 vezes para cada valor de taxa de chegada, e a média dos tempos de conexão é então calculada. As taxas de chegada são um conjunto de valores determinísticos, definidos como parâmetro no Httperf. Várias taxas foram utilizadas nos experimentos em cada cenário, porém as mais representativas em termos gráficos foram escolhidas para ilustrar o comportamento de cada política de balanceamento.

O primeiro cenário retrata uma carga *Web* leve. Intuitivamente, tentamos retratar o acesso a uma página comum da *Web* com textos e figuras pequenas [Bovenzi et al. 2011, Williams et al. 2005, Goseva-Popstojanova et al. 2004]. Nesse sentido, o acesso é feito a um pequeno arquivo HTML. Cada acesso a esse pequeno arquivo gera 20 requisições de 10 KB cada, totalizando 200 KB. Para cada experimento geramos 1.000 conexões, com taxas de chegada variando entre 100 e 1000 requisições por segundo.

O segundo cenário retrata uma carga típica *Web* 2.0. Nesse cenário, usuários enviam e recuperam de servidores *Web* arquivos nitidamente maiores que simples páginas. Por exemplo, fotos e arquivos de texto com cerca de 1 MB de dados [Jeon et al. 2012, Polfliet et al. 2012, Nagpurkar et al. 2008]. Assim, avaliamos o balanceador de carga utilizando requisições a um arquivo (i.e. JPG) de 1 MB. Novamente, cada experimento tem 1.000 conexões e as taxas de chegada variam entre 10 e 1.000.

Por fim, no terceiro cenário, testamos o ambiente com um arquivo de vídeo de 100

MB. Esse cenário de carga pesada é cada vez mais comum na Internet. Observamos que, cada vez mais, usuários Internet estão acessando vídeos de longa duração e com qualidade alta [Mitra et al. 2011, Summers et al. 2012]. Nesse cenário, cada experimento tem 100 conexões, cada uma com uma requisição pelo vídeo. Variamos as taxas de chegada entre 1 e 100 conexões por segundo.

Os resultados que apresentamos, como já dissemos, são valores médios de 10 execuções de cada experimento, com intervalos de confiança de 95%.

5. Resultados

Nesta parte do trabalho serão mostrados os resultados dos testes realizados para cada um dos cenários que foram citados na seção anterior. Os gráficos apresentam a relação entre o tempo médio de duração das conexões e a taxa de requisições enviadas pelo cliente. O propósito de analisar essa relação é verificar o impacto do aumento da taxa sobre o tempo de processamento das conexões.

5.1. Primeiro cenário

No primeiro cenário, os testes foram realizados com um arquivo HTML. Em cada um dos 10 experimentos são enviadas 1000 conexões com 20 requisições cada. A Figura 2 apresenta o tempo médio de resposta (e intervalo de confiança) a uma requisição, enquanto variamos a taxa de chegada nos servidores.

Para cargas leves, as três políticas de balanceamento apresentam resultados semelhantes entre si. De fato, para todas as taxas avaliadas, os intervalos de confiança das políticas apresentam interseção entre si. Intuitivamente, as cargas leves são atendidas muito rapidamente. Assim, qualquer um dos servidores que for delegado a atender uma nova requisição estará com sobra de recursos computacionais.

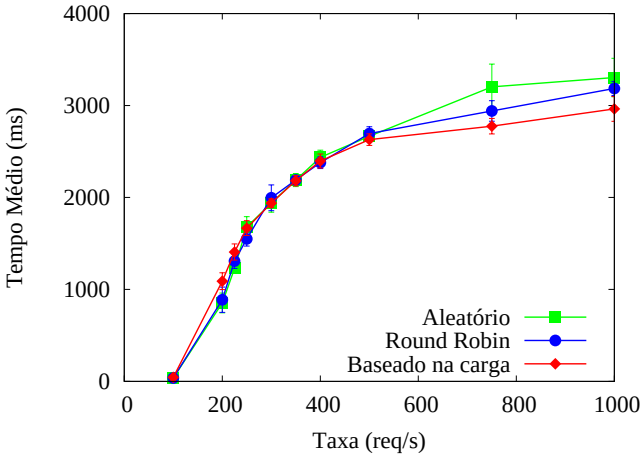

Figura 2. Tempo médio por taxa para o arquivo HTML

Com taxas mais elevadas (e.g. 500 a 750 req./s), há uma leve diferença entre as políticas. Nesse cenário, a política baseada na carga é estatisticamente diferente da abordagem aleatória. Mais precisamente, para taxas de 750 req./s, as abordagens baseada na carga, *round-robin* e aleatória apresentam tempos médios de resposta de 2775, 2941 e

3202 ms respectivamente. Para essa carga, as recusas de conexão pelo servidor *Web* são baixas para todas as políticas e variam entre 0,67% a 1,8%.

Com o aumento da carga, observamos também um aumento no percentual de requisições recusadas pelos servidores. Além disso, com uma taxa alta, a utilização dos servidores pode mudar rapidamente. Infelizmente, o mecanismo Linux que provê estatísticas de carga de um servidor apresenta médias do último minuto avaliado. Assim, um servidor, com baixa carga no último minuto, pode receber uma rajada de conexões. Isso faz com que o tempo de resposta médio e a taxa de recusas piorem, se comparados com taxas menores. De fato, como observamos na figura, para a taxa de 1.000 req./s, a política baseada na carga tem resultados equivalentes à política aleatória. Porém, a taxa de recusa de conexões chega a 7,5% para a política baseada na carga dos servidores, contra 0,65% da política *round-robin*.

5.2. Segundo cenário

No segundo cenário, os testes foram realizados com um arquivo JPG. Novamente, em cada um dos 10 experimentos são realizadas 1000 conexões, com uma requisição a cada arquivo.

De acordo com a Figura 3, para uma taxa baixa de requisições, todas as três políticas possuem resultados equivalentes. Nessa situação (taxa baixa de chegada), os servidores *Web* do sistema tem recursos suficientes para atender as requisições. Taxas baixas não geram rajadas a um único servidor (o que pode acontecer pela política aleatória e por carga).

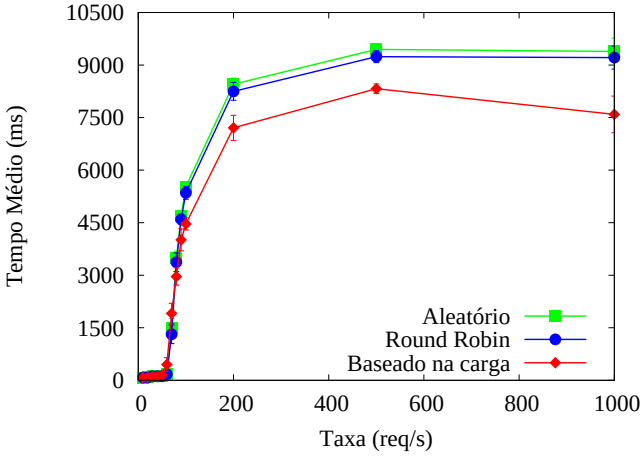

Figura 3. Tempo médio por taxa para o arquivo JPG

Para taxas a partir de 90 req./s, a política baseada na carga passa a ter melhor desempenho que as demais. Por exemplo, para uma taxa de 500 req./s, a política baseada na carga é 12% melhor que a aleatória. Mais detalhadamente, a Figura 4 apresenta a distribuição de probabilidade acumulada do tempo de resposta para taxas de chegada de 500 req./s e 20 execuções de experimentos. Por essa figura, fica claro verificar que as políticas Round-Robin e aleatória são equivalentes. Entretanto, a política baseada na carga pode apresentar respostas até 1s mais rápidas que as demais.

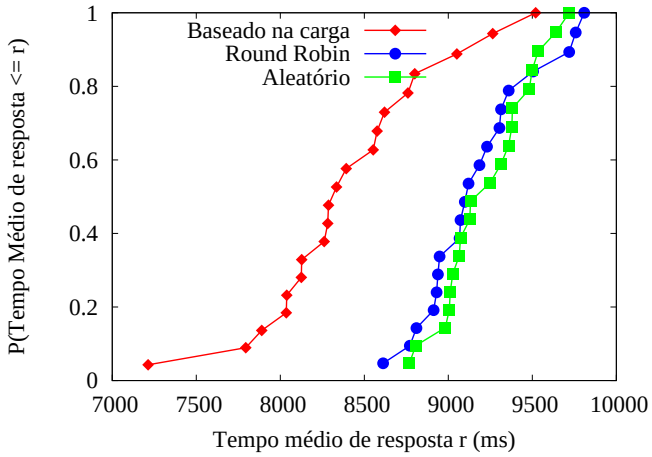

Figura 4. Tempo de Resposta para Taxas de Chegada de 500 req./s

Novamente, para taxas muito altas, observamos rajadas de requisições. Isso faz com que os servidores rejeitem novos clientes e aumentem assim a taxa de recusa. No cenário atual, o percentual de recusa é consideravelmente maior, quando comparado ao primeiro cenário. Por exemplo, para a política baseada na carga, uma taxa de 1000 req./s gerou 13,16% de recusas.

5.3. Terceiro cenário

O terceiro cenário de testes apresenta cargas pesadas a serem processadas pelo sistema. Nesse cenário, realizamos 100 conexões em cada execução do experimento. Cada conexão requisita um arquivo de vídeo de 100MB.

A Figura 5 apresenta os tempos médios de resposta para taxas de chegada variando de 1 a 100 req./s. Para praticamente todas as taxas avaliadas, as três políticas não são equivalentes. Porém, ao contrário dos outros dois cenários, a política baseada na carga tende a ser a pior.

Nesse cenário, as requisições demoram a ser processadas pelo servidor *Web*. Vídeos grandes são executados em minutos, ao contrário dos poucos segundos observados nas cargas dos dois cenários anteriores. Os servidores *Web* utilizados têm recursos limitados, e assim, saturam-se rapidamente. Como consequência, tão logo algumas poucas requisições consumam os recursos dos servidores *Web*, qualquer tentativa de balanceamento de carga se mostra ineficiente.

Além disso, nesse cenário, a letargia em atualizar o valor da carga média de trabalho da CPU pelo sistema operacional de cada um dos servidores gera resultados ainda piores que o esperado. Como, no início dos experimentos, o servidor que apresenta o menor valor médio de carga de trabalho será escolhido, toda nova requisição será direcionada para esse servidor. Porém, como uma requisição de vídeo demora a ser processada, ela ficará impactando o servidor até que o sistema operacional informe ao controlador da rede uma atualização de sua carga de trabalho. Vale ressaltar que a atualização do valor da carga da CPU é lenta, pois este valor refere-se à média de cargas do último minuto. Como, neste caso, apenas um servidor será responsável por atender a quase todas as requisições, e o outro servidor praticamente não terá nenhuma requisição para atender,

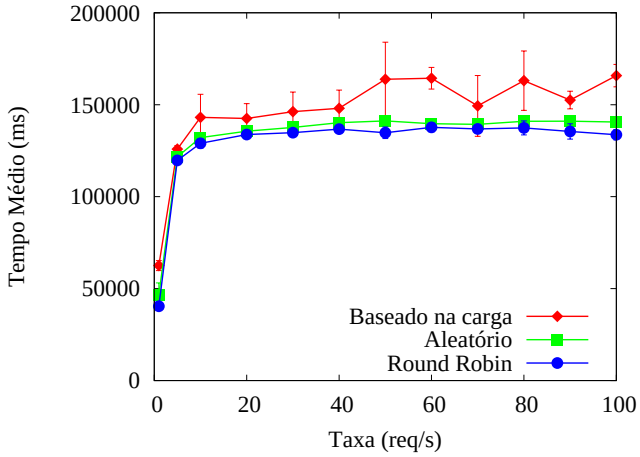

Figura 5. Tempo médio por taxa para o arquivo de vídeo

os tempos médios de resposta ficarão maiores para esse cenário.

6. Considerações Finais

A consolidação do paradigma de Redes Definidas por Software vem mudando a forma de gerenciar e projetar as redes de computadores, permitindo que elas possam evoluir e melhorar mais rapidamente em relação ao que ocorre atualmente. Dessa forma, com SDN foi possível criar um ambiente de teste para estudar um tema que tem um importante papel nas redes de computadores: o balanceamento de carga.

Este trabalho propõe e avalia o uso de SDN para o balanceamento de carga. Foram avaliadas três políticas de balanceamento de carga, a saber: política aleatória, política *round-robin* e uma política ciente da carga dos servidores de aplicação. As soluções propostas foram avaliadas em três cenários *Web*.

No primeiro cenário, equivalente a um cenário *Web* típico onde páginas são requisitadas a um servidor, as três políticas de balanceamento apresentam desempenhos semelhantes. As requisições consomem poucos recursos e são atendidas rapidamente. Assim, qualquer servidor pode atender requisições com recursos suficientes. Para cargas equivalentes a *Web* 2.0, como fotos e arquivos grandes, a política que leva em consideração a carga apresenta uma melhora de até 11% no tempo de resposta. Isso representa respostas aos clientes *Web* até 1 segundo mais rápidas. Surpreendentemente, para cargas pesadas, cada vez mais comuns, a política que leva em consideração a carga apresenta o pior resultado entre as três políticas consideradas. Nesse caso, o tempo de reação requisitado pela aplicação *Web* é muito inferior ao tempo de atualização de dados da carga nos servidores *Web*. Essa letargia por parte do sistema operacional faz com que rajadas de requisições sejam encaminhadas a um único servidor, aumentando também o tempo médio de resposta das requisições.

A arquitetura proposta pode ter melhores resultados a partir de duas frentes: primeiro, podemos refinar o esquema de medição de carga disponível no sistema operacional do servidor. Esse esquema deve ser rápido e reagir a novas cargas tão rápido quanto mudam as condições da rede. Segundo, podemos mesclar as políticas de balanceamento de carga e adicionar mecanismos para evitar rajadas.

Também pretendemos avaliar a proposta, e novas políticas de balanceamento de carga, em um ambiente de produção. Nesse sentido, estamos criando um arcabouço SDN equivalente ao proposto nesse trabalho, utilizando os recursos existentes em um *campus* universitário. Podemos apontar também como trabalho futuro extensões ao balanceamento de carga, como por exemplo, criar políticas de economia de energia entre os servidores existentes.

Agradecimentos

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

Referências

Bovenzi, A., Cotroneo, D., Pietrantuono, R., and Russo, S. (2011). Workload characterization for software aging analysis. In *Proceedings - International Symposium on Software Reliability Engineering, ISSRE*, pages 240–249.

Feamster, N., Rexford, J., and Zegura, E. (2014). The road to SDN: an intellectual history of programmable networks. In *ACM SIGCOMM Computer Communication Review*, volume 44, pages 87–98. ACM New York, NY, USA.

Goseva-Popstojanova, K., Mazimdar, S., and Singh, A. D. (2004). Empirical study of session-based workload and reliability for web servers. In *Proceedings - International Symposium on Software Reliability Engineering, ISSRE*, pages 403–414.

Guedes, D., Vieira, L. F. M., Vieira, M. M., Rodrigues, H., and Nunes, R. V. (2012). Redes definidas por software: uma abordagem sistêmica para o desenvolvimento de pesquisas em redes de computadores. In *Minicurso do XXX Simpósio Brasileiro de Redes de Computadores e Sistemas Distribuídos - SBRC 2012*, Ouro Preto, Brasil.

Handigol, N., Seetharaman, S., Flajslik, M., McKeown, N., and Johari, R. (2009). Plug-n-serve: load-balancing web traffic using openflow. In *Proceedings of demo at ACM SIGCOMM*.

Jeon, M., Kim, Y., Hwang, J., Lee, J., and Seo, E. (2012). Workload characterization and performance implications of large-scale blog servers. *ACM Transactions on the Web*, 6(4).

Korner, M. and Kao, O. (2012). Multiple service load-balancing with openflow. In *Proceedings of the IEEE 13th Conference on High Performance Switching and Routing*. IEEE Publishers, Belgrado, Sérvia.

Mitra, S., Agrawal, M., Yadav, A., Carlsson, N., Eager, D., and Mahanti, A. (2011). Characterizing web-based video sharing workloads. *ACM Transactions on the Web*, 5(2).

Mosberger, D. and Jin, T. (1998). httperf: A tool for measuring web server performance. In *Proceedings of the 1998 Internet Server Performance Workshop*, volume 26, pages 31–37.

Nagpurkar, P., Horn, W., Gopalakrishnan, U., Dubey, N., Jann, J., and Pattnaik, P. (2008). Workload characterization of selected jee-based web 2.0 applications. In *Proceedings of IEEE International Symposium on Workload Characterization 2008*, pages 109–118.

ONF (2014). What is onf? Disponível em: https://www.opennetworking.org /images/stories/downloads/about/onf-what-why.pdf. Acessado em: Novembro de 2014.

Pfaff, B., Pettit, J., Koponen, T., Amidon, K., Casado, M., and Shenker, S. (2009). Extending networking into the virtualization layer. In *8th ACM Workshop on Hot Topics inNetworks (HotNets-VIII). New York, NY, USA*.

Polfliet, S., Ryckbosch, F., and Eeckhout, L. (2012). Studying hardware and software trade-offs for a real-life web 2.0 workload. In *ICPE'12 - Proceedings of the 3rd Joint WOSP/SIPEW International Conference on Performance Engineering*, pages 181–192.

POX (2014). Pox wiki. Disponível em: https://openflow.stanford.edu/display/ONL/POX+Wiki. Acessado em: Novembro de 2014.

Ragalatha, P., Challa, M., and K, S. K. (2013). Design and implementation of dynamic load balancer on openflow enabled sdns. In *Research Paper in International Organization of Scientific Research Journal of Engineering (IOSRJEN), ISSN: 2250-3021*, volume 3, Issue 8, pages 32–41.

Sherwood, R., Gibb, G., Yap, K.-K., Appenzeller, G., Casado, M., McKeown, N., and Parulkar, G. (2009). Flowvisor: A network virtualization layer. In *Technical Report Openflow-tr-2009-1, OpenFlow*. Stanford University.

Summers, J., Brecht, T., Eager, D., and Wong, B. (2012). Methodologies for generating http streaming video workloads to evaluate web server performance. In *ACM International Conference Proceeding Series*.

Uppal, H. and Brandon, D. (2010). Openflow based load balancing. In *Proceedings of CSE561: Networking Project Report*. University of Washington, Spring.

Wang, R., Butnariu, D., and Rexford, J. (2011). Openflow-based server load balancing gone wild. In *Hot-ICE'11 Proceedings of the 11th USENIX conference on Hot topics in management of internet, cloud, and enterprise networks and services*. USENIX Association Berkeley, CA, USA.

Williams, A., Arlitt, M., Williamson, C., and Barker, K. (2005). Web workload characterization: Ten years later. In *Web Content Delivery*, volume 2 of *Web Information Systems Engineering and Internet Technologies Book Series*, pages 3–21. Springer US.

Zhou, Y., Ruan, L., Xiao, L., and Liu, R. (2014). A method for load balancing based on software defined network. In *Advanced Science and Technology Letters*, volume 45, pages 43–48.