# Monitoramento de Desempenho com *Middleboxes* em Redes Definidas por Software

Ethel B. Gondim, Antônio J. Pinheiro e Divanilson R. Campelo

***Resumo*— O gerenciamento de desempenho de aplicações é frequentemente dificultado por *middleboxes* em razão de sua variedade e capacidade de alterar o tráfego que os atravessa. Com o advento das Redes Definidas por Software, surgem novas possibilidades para o gerenciamento de desempenho a partir da programabilidade dos dispositivos e do controle centralizado do tráfego. Este trabalho propõe uma arquitetura que objetiva mitigar desafios impostos pelos *middleboxes* ao monitoramento de desempenho nessas redes. Em particular, é apresentado e validado um protótipo que identifica o tempo de resposta e a disponibilidade de aplicações na presença de dois *middleboxes*, um balanceador de carga e um IPS.**

***Palavras-Chave*— SDN, Gerenciamento de Desempenho de Aplicações, *Middleboxes***

***Abstract*— Performance management is frequently hampered by the presence of middleboxes due to their variety and capacity of modifying the traffic that traverses them. With the advent of Software-Defined Networking (SDN), new possibilities for performance management arise from the programmability of devices and the centralized control of traffic. This paper proposes an architecture that aims at mitigating the challenges posed by middleboxes in performance monitoring in SDN networks. In particular, it is presented and validated a prototype that identifies the response time and the availability of applications in the presence of two middleboxes, a load balancer and an IPS.**

***Keywords*— SDN, Application Performance Management, Middleboxes**

## I. Introdução

O Gerenciamento de Desempenho de Aplicações (do Inglês, *Application Performance Management*, APM) tem por objetivo prover visibilidade do comportamento dos sistemas críticos ao funcionamento de uma organização. As soluções de APM têm sido cada vez mais presentes no mercado, o que evidencia uma dependência crescente dos negócios em sistemas computacionais. Ainda que elas tenham evoluído consideravelmente nos últimos anos, sua interpretação acerca do comportamento das aplicações é frequentemente dificultada pela presença de *middleboxes*.

Os *middleboxes*, definidos como dispositivos físicos ou virtuais que atuam sobre o tráfego com propósitos que vão além do roteamento/encaminhamento tradicional de pacotes, têm uma presença marcante na Internet atualmente [1], e frequentemente superam a quantidade de roteadores comuns em redes corporativas [2]. Nessa categoria, enquadram-se NATs, *firewalls* IP e de aplicação, *gateways* de aplicação, *proxies*, *caches* e balanceadores de carga, dentre outros. É notável a diversidade dentre os *middleboxes*, seja de funcionalidades, comportamentos, fabricantes ou mesmo de formas de implementação. Eles podem ser implementados em localizações arbitrárias e das mais variadas formas, como *appliances* físicos/dedicados, máquinas virtuais, coleções de processos, funcionalidade em um controlador SDN, dentre outros [3]. Tais características tornam extremamente complexo o monitoramento de desempenho em redes com *middleboxes*.

A grande diversidade e complexidade de dispositivos presentes nas redes foram parte da motivação para o surgimento das Redes Definidas por Software (do Inglês, *Software-Defined Networking*, SDN). Em SDN, propõe-se a separação do plano de controle do plano de encaminhamento dos dados. O plano de controle passa a contar com controladores que recebem os registros dos diversos fluxos de tráfego que passam pela rede, controlando-a de forma centralizada [4]. Além disso, a arquitetura de SDN propõe uma maior programabilidade dos dispositivos presentes nas redes. Isso implica a existência de um padrão de comunicação com os dispositivos, como o OpenFlow [5], e também uma maior presença de Interfaces de Programação de Aplicações (do Inglês, *Application Programming Interfaces*, APIs).

Tanto o controle centralizado dos fluxos quanto a programabilidade dos dispositivos propostos em SDN podem ser aproveitados para o gerenciamento. Este artigo propõe uma nova arquitetura de solução de monitoramento de desempenho de aplicações para SDN, utilizando-se dessas características da seguinte forma: as informações dos fluxos são verificadas de forma centralizada no(s) controlador(es) da rede, e o comportamento dos middleboxes é verificado regularmente por meio de APIs, seguindo uma definição padronizada de estados. O principal objetivo da arquitetura proposta é contornar as limitações que a grande presença de *middleboxes* traz às soluções de monitoramento.

Para a avaliação da arquitetura, foi desenvolvido um protótipo de solução de APM capaz de identificar o tempo de resposta e a disponibilidade de aplicações em uma rede SDN na presença de *middleboxes*. Esse protótipo foi desenvolvido em ambiente emulado no Mininet, um emulador de SDN que simula uma coleção de *hosts*, *switches*, roteadores e enlaces em uma única máquina virtual [6]. O seu funcionamento foi avaliado e validado para dois *middleboxes open-source*, um balanceador de carga e um Sistema de Prevenção de Intrusões (do Inglês, *Intrusion Prevention System*, IPS). Espera-se que o protótipo seja uma base para compor uma solução de APM completa e aplicável a ambientes complexos.

Nas seções seguintes, é apresentada uma breve revisão do

Ethel B. Gondim - Departamento de Ciência da Computação, Universidade de Brasília (UnB), Brasília-DF, Brasil (e-mail:ethelbg@aluno.unb.br).

Antônio J. Pinheiro e Divanilson R. Campelo - Centro de Informática (CIn), Universidade Federal de Pernambuco (UFPE), Recife-PE, Brasil (e-mails: ajp@cin.ufpe.br, dcampelo@cin.upfe.br).

estado da arte de monitoramento e *middleboxes* para SDN, e são descritas a arquitetura da solução, a representação de estados de *middleboxes* proposta, e o protótipo de monitoramento. É descrita na sequência a validação desse protótipo, com a descrição de sua avaliação e dos resultados. Por fim, são apresentados a conclusão e os trabalhos futuros propostos.

## II. Revisão do Estado da Arte

As Redes Definidas por Software têm recebido interesse crescente da indústria e da comunidade científica nos últimos anos. Como o conceito de SDN se aplica diretamente apenas às camadas 2 e 3 do modelo OSI, a maioria das propostas e inovações em SDN se dão nesse contexto. Há diversas propostas de arquitetura de redes SDN, de padrões similares ou complementares ao OpenFlow, e de integração de redes SDN com redes tradicionais. Já para as camadas de 4 a 7 do modelo OSI, segmento em que se concentram a maior parte das funcionalidades dos *middleboxes* e a maioria das métricas que compõem o gerenciamento de desempenho de aplicações, o desenvolvimento tem sido mais tímido.

Há algumas publicações que propõem *middleboxes* próprios para SDN, como [7], [8] e [9], que apresentam modelos de balanceadores de carga próprios para SDN, implementados a partir da codificação de uma funcionalidade do controlador SDN. Além disso, há exemplos de soluções comerciais de *middleboxes* que são compatíveis com SDN, como o BIG-IP [10], fabricado pela F5, o NetScaler [11], desenvolvido pela Citrix, e o FastView [12], da Radware. Existem também propostas para melhorar a configuração dos *middleboxes* em SDN, como [3], [13], [14] e [15]. Em [3], é proposto um *framework* para redes com *middleboxes* em SDN, contendo um mecanismo para se exercer controle unificado sobre os fatores que influenciam as operações de *middleboxes*. Esse mecanismo possibilitaria o gerenciamento de ambientes complexos que contenham *middleboxes*. Em [13], é proposto um protocolo para configuração dinâmica para SDN de dois exemplos comuns de *middleboxes*, os NATs e os *firewalls*. Em [14], [15] e [16], são propostos novos modelos de desenvolvimento de *middleboxes* que poderiam ser aplicados em SDN.

Já em relação ao gerenciamento envolvendo *middleboxes* para SDN, foram encontrados poucos trabalhos publicados. O artigo [17] discute uma proposta para melhorar o gerenciamento de *middleboxes* em SDN sem trazer limitações de localização ou de implementação de funcionalidades a partir dos padrões de SDN já disponíveis. Em [3], como explicado, também há algum trabalho no sentido de melhorar o gerenciamento de *middleboxes*. No entanto, o tema de gerenciamento de desempenho de aplicações em redes SDN com *middleboxes* não foi abordado em nenhuma publicação encontrada, e encontrou-se apenas uma solução de APM de mercado compatível com SDN, o Riverbed SteelCentral [18], que possui código fechado. Assim, ainda há desafios e oportunidades pouco explorados nesse segmento, como os mecanismos de obtenção de dados para soluções de APM em SDN e o próprio monitoramento do comportamento dos *middleboxes*.

## III. Arquitetura de Monitoramento

A arquitetura proposta, ilustrada na Figura 1, consiste em um protótipo de solução de APM para SDN que obtém dados de dois tipos de elementos da rede: os controladores SDN, de onde são obtidos os dados de fluxos das redes por onde trafega a aplicação monitorada; e os *middleboxes*, de onde são obtidas métricas que definem os seus estados a partir de APIs. O protótipo foi implementado como um *software* Python executado sobre o *host* centralizador na plataforma Mininet. Os dados de todos os fluxos da rede são capturados e analisados executando-se o *software* livre Wireshark [19] no controlador SDN. As informações de estados dos *middleboxes* são recebidas utilizando-se o formato *JavaScript Object Notation* (JSON), que permite que sua transmissão seja rápida e introduza baixo *overhead* à rede [20].

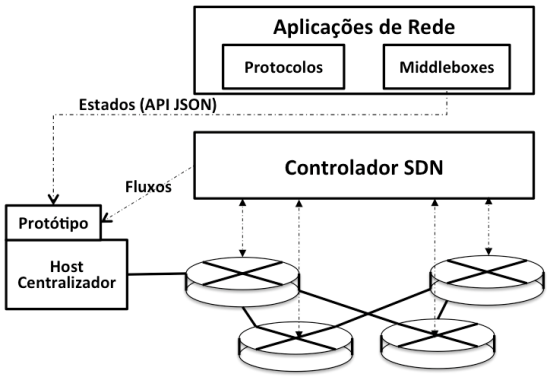


Fig. 1. Arquitetura da solução proposta.

A definição de estados dos *middleboxes* é realizada para simplificar o seu monitoramento e torná-lo agnóstico a fabricantes. De forma semelhante a [3], define-se aqui o estado de um *middlebox* como um conjunto suficiente de métricas para o monitoramento proposto no protótipo. Como o protótipo deve monitorar aplicações, o objetivo não é visualizar de forma extensiva as características de um *middlebox*, mas sim a atuação do *middlebox* sobre o tráfego de uma aplicação. A cada nova conexão trafegada, o *middlebox* envia para o *host* centralizador a tupla **ID-Conexão** (*IP-origem, IP-destino, Porta-origem, Porta-destino*), o horário de início da conexão, e as informações que indicam a atuação do *middlebox* sobre ela. A Tabela I ilustra o modelo de abstração considerado para representar o estado para alguns dos tipos de *middleboxes* mais comuns.

As informações recebidas são filtradas a partir dos IPs dos servidores e das portas de destino da conversa, e então são processadas pelo *software* para que ele compute duas das métricas mais relevantes para se indicar o desempenho de uma aplicação: seu tempo de resposta e sua disponibilidade. Além disso, é computada a quantidade de dados trocados (*bytes*) em cada conexão, e são disponibilizadas métricas específicas a cada *middlebox*, conforme indicado na Tabela I.

## IV. Avaliação

Para a validação da arquitetura proposta, o protótipo de monitoramento de desempenho foi desenvolvido em ambiente

TABELA I

ALGUNS DOS *middleboxes* MAIS COMUNS E SUAS ABSTRAÇÕES DE ESTADO, EM PROPOSTA SEMELHANTE À DE [3].

| *Middlebox* | Abstração de Estado |
|---|---|
| *Firewall* | ID-Conexão, horário, status (bloqueada/liberada) |
| NAT | ID-Conexão, horário, mapeamento de endereços |
| Balanceador de carga | ID-Conexão, horário, servidor que atendeu à requisição, status do *pool* de servidores, política de balanceamento |
| IDS/IPS | ID-Conexão, horário, nível de alerta/bloqueio |
| *Cache/Proxy* | ID-Conexão, horário, política de substituição dos dados, status (encontrado/não encontrado em cache) |

TABELA II

AVALIAÇÕES REALIZADAS PARA VALIDAR O PROTÓTIPO.

| | Balanceador | IPS |
|---|---|---|
| T1 | Cada *host* cliente faz 100 requisições HEAD. Não há indisponibilidades. | Cada *host* cliente faz 100 requisições head. Ambiente disponível e sem bloqueios. |
| T2 | Cada *host* cliente faz 100 requisições GET de um arquivo de 3900 *bytes*. Não há indisponibilidades. | Cada *host* cliente faz 100 requisições GET de um arquivo de 3900 *bytes*. Ambiente disponível e sem bloqueios. |
| T3 | Cada *host* cliente faz 300 requisições GET de um arquivo de 3900 *bytes*. Servidores são indisponibilizados na sequência h4 $\rightarrow$ h5 $\rightarrow$ h6, e retornam em sequência igual. | Cada *host* cliente faz 100 requisições HEAD. Conexões dos *hosts* h1 e h2 são bloqueadas pelo IDS (serviço indisponível para h1 e h2). Restante do ambiente disponível. |

Mininet 2.1.0, disponibilizado em [6] como uma máquina virtual de sistema operacional Ubuntu 14.04 LTS capaz de emular todos os componentes de uma rede SDN: controlador(es), *hosts*, *switches*, roteadores, etc. Ela possui todos os binários do OpenFlow 1.0 e é compatível com Open vSwitch 1.3. A máquina virtual do Mininet foi configurada com 2 CPUs virtuais e 8 GB de memória RAM. Além disso, foram utilizados alguns utilitários livres adicionais: Wireshark 1.10.6, para captura e análise de pacotes [19]; HTTPing 1.5.8, para realizar requisições HTTP e medir seus tempos de resposta [21]; e CGIHTTPServer, um servidor Web em Python capaz de lidar com requisições HTTP do tipo *Common Gateway Interface* (CGI) [22].

Foram realizadas avaliações da capacidade do protótipo de monitoramento de desempenho em redes com dois tipos de *middleboxes open-source*: um balanceador de carga executado sobre o POX 0.3.0 (controlador para OpenFlow 1.0 desenvolvido em Python [23]), e um IPS Snort 2.9.7.0 [24]. Para o envio das informações necessárias pela API, foram necessárias algumas modificações: no balanceador, foi feita uma pequena alteração em seu código para envio dos dados da Tabela I; no IPS, essas informações eram extraídas continuamente a partir de seus *logs*.[2] Foram realizados três testes para cada *middlebox*, descritos na Tabela II. Esses testes foram configurados em *scripts* sequenciais, em que são definidas as requisições HTTP realizadas por cada *host*, e os momentos em que um determinado link ou dispositivo fica disponível/indisponível.

Para validação dos resultados, as métricas obtidas pelo protótipo são comparadas a um conjunto de informações que denominamos **dados de validação**:

1) O próprio *script* de testes para o controle da disponibilidade, do status do *pool* de servidores e da política de balanceamento;
2) Capturas nas interfaces do ambiente emulado no Mininet utilizando o Wireshark para validar a quantidade de *bytes* trocados;
3) Os resultados do HTTPing para verificar o tempo de resposta de cada requisição.

[2]Idealmente, seria possível extrair essas informações a partir das APIs de fabricantes. Em [25], são listadas algumas das APIs mais relevantes para SDN e para *Network Functions Virtualization (NFV)*, o que ilustra a viabilidade da extração de várias métricas por APIs em dispositivos de mercado.

### A. Testes com o balanceador de carga

Para os três testes com o balanceador, foi emulada a topologia ilustrada na Figura 2. As Tabelas III, IV e V exibem os resultados de tempo de resposta médio e de disponibilidade média para as conexões de cada *host* cliente (h1, h2 e h3) e para o total de conexões, em comparação com os dados de validação[2].

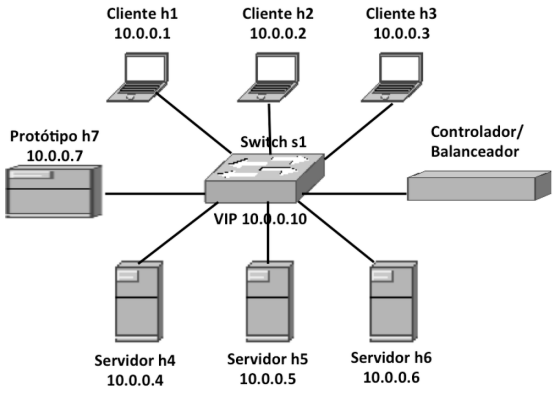


Fig. 2. Topologia utilizada nos testes com o balanceador de carga.

As informações de disponibilidade do protótipo corresponderam exatamente às dos dados de validação. Essa disponibilidade era verificada a partir da API do balanceador de carga, que checava antes de encaminhar cada requisição HTTP se havia servidores disponíveis para atendê-la, e informava esse *status* ao *host* centralizador (h7). Ela foi calculada como a porcentagem de requisições que foram atendidas para cada *host*. Apenas no teste 3, em que houve uma interrupção sequencial dos links de comunicação dos *hosts* h4, h5 e h6, foi notada alguma indisponibilidade. O período em que o serviço ficou totalmente indisponível fez com que 4 das 300 requisições de cada *host* cliente não fossem atendidas, levando a um percentual de disponibilidade de aproximadamente 98,67%.

Já na comparação entre os tempos de resposta obtidos pelo protótipo e pelo HTTPing, houve diferenças sutis, descritas na linha "Diferença T. Resp." das tabelas de resultados. Uma

[2]No teste 3 foram desconsideradas as requisições HTTP que não foram respondidas para o cômputo do tempo de resposta. Portanto, o cálculo do tempo de resposta conta com um número menor de conexões de cada *host*.

TABELA III

PRINCIPAIS RESULTADOS DO TESTE 1 COM O BALANCEADOR.

| | | *Host* 1 | *Host* 2 | *Host* 3 | Total |
|---|---|---|---|---|---|
| Protótipo | Tempo Resp. (ms) | 33,85 | 30,30 | 34,59 | 32,91 |
| | Disponibilidade (%) | 100 | 100 | 100 | 100 |
| Dados de Validação | Tempo Resp. (ms) | 33,61 | 30,22 | 34,43 | 32,76 |
| | Disponibilidade (%) | 100 | 100 | 100 | 100 |
| Diferença T. Resp. | Média Absoluta (ms) | 0,24 | 0,16 | 0,20 | 0,20 |
| | Desvio Padrão (ms) | 0,16 | 0,12 | 0,20 | 0,20 |

TABELA IV

PRINCIPAIS RESULTADOS DO TESTE 2 COM O BALANCEADOR.

| | | *Host* 1 | *Host* 2 | *Host* 3 | Total |
|---|---|---|---|---|---|
| Protótipo | Tempo Resp. (ms) | 62,23 | 58,74 | 60,03 | 60,33 |
| | Disponibilidade (%) | 100 | 100 | 100 | 100 |
| Dados de Validação | Tempo Resp. (ms) | 62,03 | 58,63 | 59,86 | 60,17 |
| | Disponibilidade (%) | 100 | 100 | 100 | 100 |
| Diferença T. Resp. | Média Absoluta (ms) | 0,25 | 0,23 | 0,24 | 0,24 |
| | Desvio Padrão (ms) | 0,44 | 0,37 | 0,32 | 0,38 |

TABELA V

PRINCIPAIS RESULTADOS DO TESTE 3 COM O BALANCEADOR.

| | | *Host* 1 | *Host* 2 | *Host* 3 | Total |
|---|---|---|---|---|---|
| Protótipo | Tempo Resp. (ms) | 62,00 | 60,34 | 61,31 | 61,22 |
| | Disponibilidade (%) | 98,67 | 98,67 | 98,67 | 98,67 |
| Dados de Validação | Tempo Resp. (ms) | 61,71 | 60,11 | 61,11 | 60,98 |
| | Disponibilidade (%) | 98,67 | 98,67 | 98,67 | 98,67 |
| Diferença T. Resp. | Média Absoluta (ms) | 0,30 | 0,23 | 0,22 | 0,25 |
| | Desvio Padrão (ms) | 0,76 | 0,16 | 0,22 | 0,47 |

possível explicação para essas diferenças é que o HTTPing trunca os dados de cada conexão em duas casas decimais, enquanto que o protótipo mantém quatro casas. Nota-se, porém, que as diferenças foram mínimas: no teste 1, representaram um valor médio absoluto de 0,20 ms; no teste 2, de 0,24 ms; e no teste 3, de 0,25 ms. Esses valores, bem como os de desvio padrão das diferenças, representam menos de 0,8% dos valores médios de tempo de resposta. Assim, as diferenças podem ser consideradas desprezíveis.

Os demais dados relativos a cada conexão (quantidade de *bytes* trocados, IP do servidor que atendeu à requisição, *status* do *pool* de servidores e política de balanceamento) foram sempre idênticos aos dados de validação. Logo, considera-se que os testes com o balanceador de carga foram bem-sucedidos, pois mostraram que os dados de monitoramento trazidos pelo protótipo são consistentes.

### *B. Testes com o IPS*

Para os três testes com o IPS, foi utilizada a topologia da Figura 3, com cinco *hosts* clientes (h1, h2, h3, h4 e h5). As Tabelas VI, VII e VIII exibem os resultados de tempo de resposta médio e de disponibilidade média para o conjunto de conexões dos testes 1, 2 e 3, respectivamente.

Os resultados dos testes com o IPS foram bastante semelhantes aos do balanceador. As informações de disponibilidade do protótipo foram idênticas às dos dados de validação: de 100% nos testes 1 e 2, e de 60% no teste 3, pois dois dos cinco *hosts* clientes tinham suas requisições sempre bloqueadas pelo IPS. Também houve diferenças sutis nos tempos de resposta

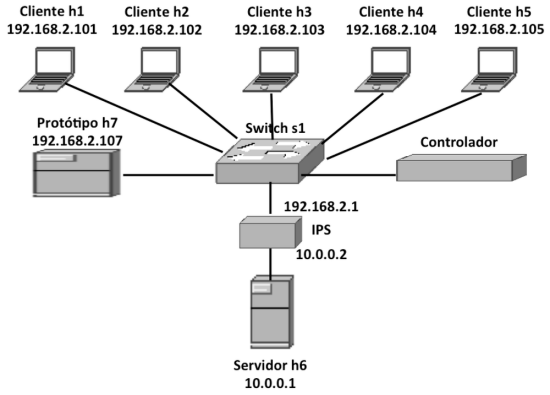


Fig. 3. Topologia utilizada nos testes com o IPS.

TABELA VI

PRINCIPAIS RESULTADOS DO TESTE 1 COM O IPS.

| | | *Host* 1 | *Host* 2 | *Host* 3 | *Host* 4 | *Host* 5 | Total |
|---|---|---|---|---|---|---|---|
| Protótipo | Tempo Resp. (ms) | 29,84 | 30,60 | 30,50 | 33,39 | 32,55 | 31,38 |
| | Disponib. (%) | 100 | 100 | 100 | 100 | 100 | 100 |
| Validação | Tempo Resp. (ms) | 29,90 | 30,68 | 30,58 | 33,46 | 32,61 | 31,44 |
| | Disponib. (%) | 100 | 100 | 100 | 100 | 100 | 100 |
| Diferença | Média Abs. (ms) | 0,08 | 0,09 | 0,10 | 0,08 | 0,09 | 0,09 |
| T. Resp. | Desv. Padrão (ms) | 0,06 | 0,07 | 0,18 | 0,10 | 0,11 | 0,11 |

TABELA VII

PRINCIPAIS RESULTADOS DO TESTE 2 COM O IPS.

| | | *Host* 1 | *Host* 2 | *Host* 3 | *Host* 4 | *Host* 5 | Total |
|---|---|---|---|---|---|---|---|
| Protótipo | Tempo Resp. (ms) | 33,70 | 32,20 | 32,03 | 34,20 | 28,96 | 32,22 |
| | Disponib. (%) | 100 | 100 | 100 | 100 | 100 | 100 |
| Validação | Tempo Resp. (ms) | 33,47 | 31,92 | 31,83 | 34,08 | 28,76 | 32,01 |
| | Disponib. (%) | 100 | 100 | 100 | 100 | 100 | 100 |
| Diferença | Média Abs. (ms) | 0,28 | 0,31 | 0,24 | 0,18 | 0,23 | 0,25 |
| T. Resp. | Desv. Padrão (ms) | 0,22 | 0,22 | 0,18 | 0,20 | 0,19 | 0,21 |

TABELA VIII

PRINCIPAIS RESULTADOS DO TESTE 3 COM O IPS.

| | | *Host* 1 | *Host* 2 | *Host* 3 | *Host* 4 | *Host* 5 | Total |
|---|---|---|---|---|---|---|---|
| Protótipo | Tempo Resp. (ms) | – | – | 29,44 | 29,78 | 29,34 | 29,52 |
| | Disponib. (%) | 0 | 0 | 100 | 100 | 100 | 60 |
| Validação | Tempo Resp. (ms) | – | – | 29,47 | 29,81 | 29,39 | 29,55 |
| | Disponib. (%) | 0 | 0 | 100 | 100 | 100 | 60 |
| Diferença | Média Abs. (ms) | – | – | 0,06 | 0,06 | 0,06 | 0,06 |
| T. Resp. | Desv. Padrão (ms) | – | – | 0,04 | 0,04 | 0,4 | 0,04 |

do protótipo e os obtidos pelo HTTPing: no teste 1, a diferença média absoluta foi de 0,09; no teste 2, de 0,25; e no teste 3, de 0,06. Esses valores, bem como os de desvio padrão das diferenças, representam menos de 0,8% dos valores médios de tempo de resposta, podendo, portanto, ser considerados desprezíveis. Os demais dados obtidos pelo protótipo (quantidade de *bytes* trocados, *status* do serviço, *status* de bloqueio do IPS) foram sempre idênticos aos de validação. Assim, novamente o protótipo trouxe informações consistentes.

## V. CONCLUSÃO E TRABALHOS FUTUROS

Este trabalho propôs uma nova arquitetura de monitoramento para SDN, com foco em atacar os desafios trazidos pela presença ampla e diversa de *middleboxes* nas redes corporativas. Sugeriu-se que os *middleboxes*, por atuarem sobre o

tráfego de forma variada e muitas vezes imprevisível, deveriam também ser monitorados por ferramentas de APM. A visibilidade do funcionamento dos *middleboxes* se daria a partir de um conjunto de métricas padronizado por tipo/funcionalidade, que seria obtido por meio de APIs, em consonância com a tendência de programabilidade de dispositivos em SDN. Em adição às métricas de *middleboxes*, sugeriu-se a coleta de fluxos de rede a partir dos controladores SDN. Como os controladores têm domínio total sobre os fluxos da rede, o trabalho de coleta de fluxos a partir desses pontos se torna extremamente simplificado em relação ao monitoramento tradicional, em que normalmente é necessário "instrumentar" diversos segmentos e dispositivos da rede.

Por fim, foi proposto um protótipo de monitoramento que une as informações dos *middleboxes* aos dados de fluxos obtidos a partir dos controladores. Esse protótipo é capaz de avaliar a disponibilidade e o tempo de resposta de uma aplicação, e de trazer também para cada conexão as informações de estado dos *middleboxes* pelas quais ela trafega. Ele foi validado em seis testes com duas topologias e dois tipos de *middleboxes* diferentes, um balanceador de carga e um IPS. Em todos os testes, o protótipo trouxe resultados bastante consistentes.

Ainda que o monitoramento implementado no protótipo seja bastante simplificado, o conceito da arquitetura proposta pode ser estendido a sistemas mais complexos, que incluam informações de mais fontes, como dados específicos aos servidores de aplicação, ou informações obtidas a partir de *traps* SNMP, por exemplo. O protótipo demonstrado não se propõe a substituir ferramentas de monitoramento tradicionais, mas sim a simplificar a forma como algumas das informações necessárias a essas ferramentas são obtidas, e a trazer maior visibilidade dos *middleboxes* de uma rede, que tradicionalmente não são foco de ferramentas de gerenciamento de desempenho, apesar de sua grande relevância.

Assim, a arquitetura proposta mostrou ser uma forma de monitoramento inovadora e viável para SDN, especialmente no que diz respeito a contornar as limitações trazidas pela presença de *middleboxes*. Em trabalhos futuros, espera-se validar o protótipo desenvolvido para abranger mais tipos de *middleboxes*, como *firewalls*, *proxies* e NATs. Espera-se, ainda, testá-lo em topologias mais complexas, com diversos serviços e *middleboxes* funcionando simultaneamente.

## REFERÊNCIAS


[1] B. Carpenter and S. Brim. Middleboxes: Taxonomy and issues. *RFC 3234*, 2002.

[2] C. Paasch and O. Bonaventure. Multipath TCP. *Commun. ACM*, 57(4):51–57, 2014.

[3] Aaron Gember, Prathmesh Prabhu, Zainab Ghadiyali, and Aditya Akella. Toward software-defined middlebox networking. In *Proceedings of the 11th ACM Workshop on Hot Topics in Networks*, pages 7–12. ACM, 2012.

[4] C. Rothenberg, M. Nascimento, M. Salvador, and M. Magalhães. OpenFlow e redes definidas por software: um novo paradigma de controle e inovação em redes de pacotes. *Cad. CPqD Tecnologia*, 7(1):65–76, 2011.

[5] OpenFlow Archive. `http://archive.openflow.org/wp/learnmore/`, 2014. Acessado em 04/05/2014.

[6] Mininet. `http://mininet.org`, 2015. Acessado em 01/02/2015.

[7] Richard Wang, Dana Butnariu, and Jennifer Rexford. OpenFlow-based server load balancing gone wild. *Workshop on Hot Topics in Management of Internet, Cloud, and Enterprise Networks and Services (Hot-ICE)*, 2011.

[8] Z. Shang, W. Chen, Q. Ma, and B. Wu. Design and implementation of server cluster dynamic load balancing based on openflow. In *International Joint Conference on Awareness Science and Technology and Ubi-Media Computing (iCAST-UMEDIA)*. IEEE, 2013.

[9] Hardeep Uppal and Dane Brandon. Openflow based load balancing. *Proceedings of CSE561: networking. project report. University of Washington, Spring*, 2010.

[10] F5. BIG-IP. `https://f5.com/products/big-ip`, 2014. Acessado em 06/05/2014.

[11] Citrix. NetScaler. `http://www.citrix.com.br/products/netscaler-application-delivery-controller/`, 2014. Acessado em 06/05/2014.

[12] Radware. FastView. `http://www.radware.com/Products/FastView/`, 2014. Acessado em 06/05/2014.

[13] RFC4540. `http://tools.ietf.org/html/rfc4540`, 2006. Acessado em 10/08/2014.

[14] Colin Dixon, Hardeep Uppal, Vjekoslav Brajkovic, Dane Brandon, Thomas Anderson, and Arvind Krishnamurthy. ETTM: a scalable fault tolerant network manager. In *Proceedings of the 8th USENIX conference on Networked systems design and implementation*, pages 7–7. USENIX Association, 2011.

[15] V. Sekar, N. Egi, S. Ratnasamy, M. Reiter, and G. Shi. Design and implementation of a consolidated middlebox architecture. In *Proceedings of the 9th USENIX conference on Networked Systems Design and Implementation*, pages 24–24, 2012.

[16] James W Anderson, Ryan Braud, Rishi Kapoor, George Porter, and Amin Vahdat. xOMB: extensible open middleboxes with commodity servers. In *Proceedings of the eighth ACM/IEEE symposium on Architectures for networking and communications systems*, pages 49–60. ACM, 2012.

[17] Zafar Qazi, Cheng-Chun Tu, Rui Miao, Luis Chiang, Vyas Sekar, and Minlan Yu. Practical and incremental convergence between sdn and middleboxes. *Open Network Summit, Santa Clara, CA*, 2013.

[18] Riverbed. SteelCentral. `http://www.riverbed.com/products/performance-management-control/`, 2015. Acessado em 20/03/2015.

[19] Wireshark. `https://www.wireshark.org`, 2015. Acessado em 20/03/2015.

[20] JSON. Introducing JSON. `http://json.org`, 2015. Acessado em 20/03/2015.

[21] HTTPing. Linux man page. `http://linux.die.net/man/1/httping`, 2015. Acessado em 21/02/2015.

[22] CGIHTTPServer. CGI-capable HTTP request handler. `https://docs.python.org/2/library/cgihttpserver.html`, 2015. Acessado em 20/03/2015.

[23] NOX/POX. NOX/POX website. `http://www.noxrepo.org/`, 2015. Acessado em 27/02/2015.

[24] Snort. `https://www.snort.org`, 2015. Acessado em 10/03/2015.

[25] SDNCentral. SDN API List. `https://www.sdncentral.com/comprehensive-list-of-sdn-apis/`, 2015. Acessado em 20/01/2015.