Avaliação de Controladores *Software-Defined Networking* Utilizando Metodologia Padrão para *Benchmark*

Émerson R. Silva[1], Kayo H. C. Monteiro[2], Ramide Augusto Sales Dantas[3], Edison Q. Albuquerque[1], Patricia T. Endo[1,2]

[1]Universidade de Pernambuco – Escola Politécnica de Pernambuco – Brasil

[2]Universidade de Pernambuco – Campus Caruaru – Brasil

[3]Instituto Federal de Pernambuco – Campus Recife – Brasil

{ers2, edison}@ecomp.poli.br, kayohcm@gmail.com, patricia.endo@upe.br

Resumo. *Software-Defined Networking (SDN) é uma arquitetura emergente de redes de computadores que desacopla o plano de controle e o plano de dados. Diversos trabalhos na literatura tem como objetivo identificar o melhor controlador SDN, porém, comparam o desempenho de controladores com características distintas, por exemplo, suporte a multi-thread e arquitetura. Este trabalho apresenta uma avaliação de controladores com arquitetura centralizada utilizando um teste padronizado da IETF no intuito de reforçar os testes da literatura e tornar a avaliação mais completa. Os controladores foram avaliados sob as mesmas condições e classificados de acordo com suas características, com o objetivo de se obter uma avaliação mais justa. Foram selecionados cinco controladores populares na literatura (Beacon, Floodlight, NOX-MT, POX, e Ryu), para os quais identificou-se os melhores cenários dentre os testes realizados. O controlador Beacon apresentou o melhor desempenho geral, chegando a obter uma taxa de transferência 46.7x maior que o controlador com menor desempenho.*

1. Introdução

Software-Defined Network (SDN) é uma arquitetura emergente para redes de computadores que permite a dissociação entre o plano de controle (funções e ações executadas para funcionamento da rede) e o plano de dados (meios de encaminhamento de pacotes), viabilizando a programabilidade da rede [Haleplidis et al. 2015]. O plano de controle separado dos dispositivos da rede permite a identificação mais eficiente de problemas, os quais podem ser solucionados através de medidas predefinidas ou baseadas no comportamento geral da rede. Os controladores SDN que formam o plano de controle e os dispositivos do plano de dados se comunicam através protocolos específicos, sendo OpenFlow o padrão da indústria atualmente[McKeown et al. 2008].

Como consequência da aceitação do OpenFlow, estudos visando o aprimoramento e a adequação ao cenário atual surgiram em todo o mundo e mantêm-se através da *Open Networking Foundation* (ONF) [Foundation 2017b]. Estes estudos tiveram como resultados o desenvolvimento de mais cinco versões do mesmo, desde o surgimento do protocolo em 2008. Enquanto existem grandes esforços para o desenvolvimento do OpenFlow, que o permitiu tornar-se o padrão de comunicação nos componentes SDN, não há ainda

o mesmo consenso com relação aos controladores, uma vez que podem ser desenvolvidos em diversas linguagens de programação. Ainda que fossem desenvolvidos em uma mesma linguagem, não haveria uma padronização da *northbound interface*, ou seja, cada controlador possuiria uma sintaxe própria e diferente para implementação das aplicações.

O controlador é responsável pelo plano de controle e é o componente principal de uma SDN. Ele se conecta a todos os dispositivos genéricos de comutação de pacotes, neste trabalho chamado de *switch* SDN, e gerencia o encaminhamento de todos os pacotes da rede. Além disto, ele pode refazer rotas devido a enlaces perdidos, balancear carga de trabalho e tratar estatísticas sobre a rede. Com o controlador como centro de uma SDN, diretamente responsável pelo desempenho da rede, as entidades de pesquisa e grandes empresas iniciaram esforços para terem seu próprio controlador [Kreutz et al. 2015].

No meio acadêmico e industrial, surgiram em torno de trinta controladores, que divergem em características importantes, como: versão suportada do protocolo OpenFlow, linguagem de desenvolvimento e arquitetura [Kreutz et al. 2015]. Isto dificulta a escolha de controladores ao se implantar SDN em uma organização. Faz-se necessário, portanto, uma avaliação de desempenho para identificar e selecionar os melhores controladores, possibilitando também a identificação de problemas existentes.

Os *benchmarks* recentes de controladores avaliam desempenho e escalabilidade [Tootoonchian et al. 2012, Shah et al. 2013, Fernandez 2013, Zhao et al. 2015, Salman et al. 2016], ou apenas desempenho [Alencar et al. 2014], e utilizam principalmente o Cbench [Sherwood and KOK-KIONG 2010] na emulação de carga de trabalho. Shalimov *et al.* [Shalimov et al. 2013] realizou um estudo avançado de controladores, avaliando, além de desempenho e escalabilidade, confiabilidade e segurança.

Apesar do número de trabalhos avaliando controladores na literatura, os testes tipicamente aplicados não abrangem todas as características necessárias para a escolha do controlador para uma SDN, por exemplo, a quantidade total de *switches* SDN que o controlador consegue manter. Em alguns trabalhos, a omissão de detalhes importantes sobre os testes influencia diretamente em dois fatores: no resultado em si da avaliação, que pode se tornar tendencioso; e na replicabilidade dos experimentos, dificultando assim uma comparação com outros trabalhos. Partindo dessa observação, viu-se a necessário da aplicação de testes seguindo uma metodologia padronizada para *benchmark* de controladores, com descrições detalhadas de todo o ambiente de teste, tornando a avaliação mais completa, justa e replicável. Então, para tanto, utiliza-se uma metodologia padronizada proposta pela IETF para avaliação de controladores SDN [Vengainathan et al. 2018a].

Neste trabalho, portanto, é realizada uma avaliação de controladores SDN de arquitetura centralizada, respeitando suas características, como a versão do protocolo OpenFlow e mecanismos de paralelismo. São executados testes para verificar o desempenho do controlador sob diferentes cargas de trabalho, possibilitando a identificação do melhor cenário de cada controlador. Além disto, um teste padronizado é aplicado para verificar uma característica (quantidade máxima de conexões estabelecidas e mantidas entre os *switches* SDN e o controlador) que nenhum teste na literatura verificou.

2. Padronização da SDN

Depois que a tecnologia SDN atraiu a atenção da indústria e academia, foi a vez dos órgãos de padronização iniciarem os trabalhos nesta área [Halpern 2014]. A ONF

[Foundation 2017b], por exemplo, é uma organização criada com o propósito de promover a adoção de SDN através do desenvolvimento de padrões abertos, tendo crescido devido à adoção generalizada do protocolo OpenFlow [McKeown et al. 2008]. As discussões na ONF abrangem diversos aspectos de SDN divididos em nove grupos de trabalho (WGs). Atualmente, ela tem enfocado a combinação de padrões e *softwares* de código aberto, visando transformar componentes desagregados em sistemas completos e integrados. Na ONF, os padrões SDN são aprovados e publicados sob uma das seguintes classificações:

- Especificações técnicas: padrões relacionados ao OpenFlow são publicados como especificações técnicas, podendo incluir definições de protocolo, informações, funcionalidade de componentes e documentos de estrutura relacionados;
- Recomendações técnicas: padrões que definem APIs, protocolos, modelos de dados e similares. As recomendações técnicas são protegidas pela licença Apache 2.0 [Foundation 2017a] ou direitos autorais, podendo ser utilizadas desde que não sejam alteradas e/ou vendidas, e sejam devidamente citadas;
- Informativo: documentos para ajudar a promover a missão da ONF no desenvolvimento de soluções de código aberto para redes. Estes documentos não são normativos e apenas protegidos por direitos autorais, podendo ser utilizados desde que não sejam alterados e/ou vendidos, e sejam devidamente citados.

A *Internet Engineering Task Force* (IETF) é um órgão de padronização cujo objetivo é melhorar o funcionamento da Internet através da produção de documentos técnicos de alta qualidade, que influenciam a forma como as pessoas projetam, usam e gerenciam a Internet [Society 2017]. Atualmente a IETF mantém mais de 100 WGs *(Working Groups)*, sendo quatro relacionados a SDN.

O processo de padronização na IETF é simples. O autor verifica se seu documento se encaixa em algum WG e o publica como *Internet-Draft*. Caso não se encaixe em nenhum WG, ele é iniciado como submissão individual e recebe sugestões de melhorarias de especialistas da área. Este processo se repete até o documento ficar pronto e caso se passem 6 meses da data de publicação é necessário renovar a validade do *Internet-Draft* por mais 6 meses. Quando o *Internet-Draft* estiver suficientemente discutido e houver consenso no WG, o *draft* poderá então se tornar um padrão. O documento será encaminhado para avaliação pelo presidente do WG, e após verificada a coerência do documento e possíveis correções, o *draft* passa, caso aprovado, a ser uma *Request for Coments* (RFC), documento de maior peso da IETF, que é imutável, embora o nome sugira a possibilidade de mudanças.

Em [Halpern 2014] é feita uma análise da situação de padronização de SDN, e nele os autores concluem que "*o panorama de padronização em SDN é bastante amplo*". Entretanto, ainda há pouco esforço para padronizar a avaliação de controladores. Não há um grupo específico que trate disto, apenas um *draft* proposto no *benchmark methodology work group* (BMWG) da IETF, descrito na seção 3. Neste trabalho, ressalta-se a importância de utilizar testes padronizados para comparações justas entre controladores.

3. Metodologia padronizada para *Benchmark* de Controladores SDN

Vengainathan *et al*. propôs um *draft* para a IETF com uma metodologia para *benchmark* de controladores em SDN [Vengainathan et al. 2018a], neste artigo referenciado como "proposta da IETF". O *draft* teve início como uma submissão individual, pois não havia

inicialmente um WG específico para esse tópico. Logo após a segunda versão, o *draft* foi migrado para o BMWG da IETF. Atualmente está em sua nona versão, tendo sido requisitada a avaliação para sua publicação.

Na proposta da IETF o controlador é tratado como uma caixa preta, independente de implementação, e pode ser usado em dois modos diferentes: autônomo ou *cluster*. Além disto, os controladores são avaliados em relação a quatro métricas: desempenho, escalabilidade, segurança e confiabilidade. A Tabela 1 apresenta todos os testes disponíveis na proposta da IETF por modo do controlador, autônomo (A) e *cluster* (C), e por métrica de desempenho (Des.), escalabilidade (Esc.), segurança (Seg.) e confiabilidade (Con.).

Tabela 1. Testes da Proposta da IETF

Métrica	Teste	Controlador		Descrição
		A	C	
(Des.)	*Network Topology Discovery Time*	✓	✓	tempo para determinar a topologia completa da rede
	Asynchronous Message Processing Time	✓	✓	tempo para processar uma mensagem assíncrona
	Asynchronous Message Processing Rate	✓	✓	número máximo de processamento de mensagens assíncronas
	Reactive Path Provisioning Time	✓	✓	tempo para configurar um caminho reativamente entre nós
	Proactive Path Provisioning Time	✓	✓	tempo para configurar um caminho proativamente entre nós
	Reactive Path Provisioning Rate	✓	✓	número máximo de caminhos estabelecidos reativamente
	Proactive Path Provisioning Rate	✓	✓	número máximo de caminhos estabelecidos proativamente
	Network Topology Change Detection Time	✓	✓	tempo para detectar qualquer mudança na topologia da rede
(Esc.)	*Control Session Capacity*	✓	✓	número máximo de sessões que consegue manter
	Network Discovery Size	✓	✓	mede o tamanho de uma rede que consegue descobrir
	Forwarding Table Capacity	✓	✓	número máximo de fluxos que mantêm em sua tabela de fluxos
(Seg.)	*Exception Handling*	✓	✓	efeito de gerenciar pacotes mal formados nos testes em (1)
	Denial of Service Handling	✓	✓	efeito de tratar ataques DoS nos testes em (1) e (2)
(Con.)	*Controller Failover Time*	-	✓	tempo que um controlador leva para substituir outro que falhou
	Network Re-Provisioning Time	✓	✓	tempo para reencaminhar o tráfego quando há uma falha nos enlaces

No modo autônomo, há apenas um controlador que é responsável por gerenciar um ou mais *switches* SDN; no modo *cluster* há dois ou mais controladores e todos devem ser idênticos. Controladores diferentes na mesma rede estão fora do escopo da proposta

da IETF. O teste *controller failover time* não está disponível no modo autônomo porque mede a quantidade de tempo que um controlador precisa para assumir o lugar de outro em caso de falha, o que não se aplica no modo autônomo pois há apenas um controlador.

Todos os testes são descritos no documento de terminologia [Vengainathan et al. 2018b], apresentando suas definições e unidades de medida. Por outro lado, a metodologia [Vengainathan et al. 2018a] descreve o objetivo, configuração referência do teste, pré-requisitos, procedimento, medidas e formato de relatório. Os testes usam os três tipos de mensagem do protocolo OpenFlow [ONF 2009]: controlador para *switch*, usadas para gerenciar ou verificar diretamente o estado de um *switch* SDN pelo controlador; assíncrona, iniciadas pelo *switch* SDN sem o controlador requisitar; e simétrica, enviada do controlador ou *switch* SDN, pode ser em qualquer direção. Cada teste deve ser repetido no mínimo dez vezes. Também há uma recomendação que o controlador deva estar diretamente conectado ao plano de dados (*switches* SDN) e ao plano de gerenciamento (aplicações), para evitar atrasos e falhas durante os testes.

A proposta da IETF é o único método padronizado para avaliação de controladores da literatura, no entanto, ainda pode ser melhorada. Uma abordagem interessante é o teste de confiabilidade de Shalimov *et al*. [Shalimov et al. 2013] porque examina o comportamento do controlador durante uma alta carga de trabalho. Isto permite aos pesquisadores checarem falhas que podem ocorrer durante o tempo em que o controlador está funcionando, como erros de alocação de memória, fechamento errado de conexões e pacotes descartados. Este teste poderia complementar os testes de confiabilidade da proposta da IETF.

4. Experimentos e Resultados

A descrição detalhada dos procedimentos de testes é importante para garantir a replicabilidade do trabalho e esclarecer as condições em que os resultados foram obtidos. Esta seção inicia descrevendo a metodologia dos procedimentos de teste, incluindo os controladores avaliados (com respectivas versões), e os recursos de hardware e software utilizados. Em seguida são apresentados e discutidos os resultados.

4.1. Metodologia

Os testes executados neste trabalho foram: o *Control Session Capacity* da proposta da IETF [Vengainathan et al. 2018a] e os testes de taxa de transferência e latência (usando o Cbench [Sherwood and KOK-KIONG 2010] para emular o tráfego) para verificar o comportamento do controlador sob cargas de trabalho variadas. Os controladores utilizados nos testes são listados na Tabela 2; a versão utilizada foi a mais recente possível e sem alterações no código-fonte. A aplicação SDN adotada para os todos controladores foi um *learning switch*, visto que é uma aplicação genérica presente em todos controladores.

Foram escolhidos os controladores mais utilizados na literatura (baseado em [Tootoonchian et al. 2012, Shah et al. 2013, Fernandez 2013, Shalimov et al. 2013, Alencar et al. 2014, Zhao et al. 2015, Salman et al. 2016]), com código aberto e arquitetura centralizada. No intuito de avaliá-los sob as mesmas condições, o protocolo utilizado na *southbound interface* foi o Openflow na versão 1.0.

Os *softwares* do ambiente de execução dos controladores mudam devido de acordo com a linguagens em que foram desenvolvidos. O NOX-MT é desenvolvido em

C++ e foi compilado no GCC(versão 4.8.4). O Beacon [Erickson 2017] e Floodlight [Floodlight 2017], desenvolvidos em Java, utilizam o Java 8 com o *update* 144(Java *Runtime Environment* 1.8.0_144). Os controladores desenvolvidos em Python (POX e Ryu) utilizam o interpretador PyPy (em sua versão 2.2.1) que garante um ganho de desempenho [Zhao et al. 2015].

Tabela 2. Versão e aplicação dos controladores

Controladores	Versão	Aplicação	Inicializar a aplicação
POX	0.5.0	l2_learning.py	./pox.py pox.forwarding.l2_learning
NOX-MT	0.9.2	switch	./nox_core -i ptcp:6633 switch
Beacon	1.0.4	configurationSwitch	./beacon -configuration configurationSwitch
Ryu	4.3	simple_switch.py	ryu-manager simple_switch.py
Floodlight	Master	learningswitch. LearningSwitch	java -jar target/floodlight.jar

Os testes foram realizados em um único servidor, portanto a comunicação entre os *switches* SDN e controlador realiza-se via *localhost*. Este servidor executa Ubuntu Linux (14.04.5 com kernel 3.19.0-80) sobre processador Intel Core i5-4210U (2.7GHz e 4 núcleos) e 12GB de memória RAM (DDR3 de 1600MHz). O isolamento de núcleos não foi utilizado para geração de carga ou execução do controlador porque se constatou que houve perda no desempenho ao isolar um núcleo para o Cbench. Uma vez que a geração de carga de trabalho consome pouco processamento de um núcleo, o melhor desempenho neste trabalho foi obtido permitindo a concorrência direta entre os processos.

Os gráficos apresentados nas seções a seguir estão divididos por escala para melhor visualização da dispersão dos resultados nos experimentos. Neste trabalho, usou-se 30 repetições por experimento para obter um nível de confiança de 95% [Montgomery et al. 2003]. A hipótese nula, dos *p-values*, sempre é que: ”os valores das distribuições no mesmo controlador apresentadas nos *boxplots* são iguais”, e a hipótese alternativa ”os valores das distribuições no mesmo controlador apresentadas nos *boxplots* são diferentes”. Assim, quando o *p-value* for menor que 0.05 a hipótese alternativa é aceita e os valores são diferentes, com *p-value* maior que 0.05, a hipótese nula é aceita.

Em caso de aceitar a hipótese nula (as distribuições forem iguais), o teste de hipótese é executado novamente para verificar se uma distribuição é melhor (maior em caso de taxa de transferência ou menor em caso de latência) do que a outra. Então a hipótese nula passa a ser: ”a primeira distribuição não é melhor em relação a segunda distribuição”, e a hipótese alternativa ”a primeira distribuição é melhor em relação a segunda distribuição”. Então, quando o *p-value* for menor que 0.05 a hipótese alternativa é aceita e a primeira distribuição é melhor, com *p-value* maior que 0.05, a hipótese nula é aceita. Por fim, caso não exista distribuição melhor do que a outra, a mediana é utilizada para identificar o melhor cenário do controlador em questão naquele teste.

4.2. *Control Session Capacity*

O teste de capacidade de controle de sessão (*Control Session Capacity*) mede o número máximo de conexões OpenFlow ativas de *switches* que o controlador consegue estabelecer e manter. A medida do teste é dada pelo número de conexões estabelecidas com

sucesso. Assim, quando a conexão falhar o resultado será o número imediatamente anterior, que obteve sucesso. Este é um teste de escalabilidade e faz parte da proposta da IETF. A Figura 1, baseada no procedimento descrito em [Vengainathan et al. 2018a], apresenta os resultados do teste com a capacidade máxima de cada controlador. Estes resultados foram representados por *boxplot* pois o Floodlight não chegou a um resultado exato, ao contrário dos demais controladores. Os resultados com dados estatísticos são apresentados na Tabela 3.

Figura 1. Teste *Control Session Capacity*

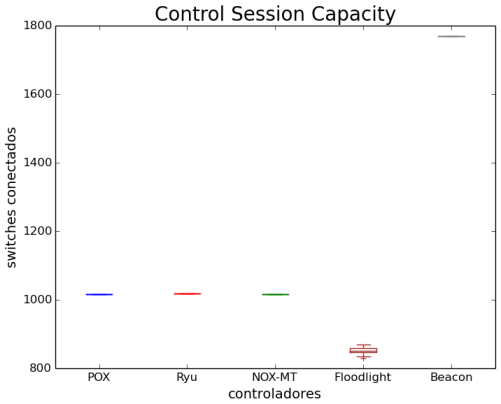


Tabela 3. Dados estatísticos do teste *Control Session Capacity*

Controladores	**POX**	**Ryu**	**NOX-MT**	**Floodlight**	**Beacon**
Mínimo	1016	1017	1015	829	1769
Máximo	**1016**	**1017**	**1015**	**869**	**1769**
Mediana	1016	1017	1015	850,5	1769
Desvio Padrão	0	0	0	9,244	0

O POX conseguiu estabelecer conexões com 1016 *switches* SDN, como apresentado na Tabela 3, sem falhas ou fechamentos inesperados de conexão. Ao ultrapassar o limite de conexões o seguinte aviso é exibido: ”ERROR:openflow.of_01:*Couldn’t accept connection: out of file descriptors*”. Os *switches* SDN que excederam o limite de conexões aguardam a resposta `hello` do controlador, a qual só será encaminhada pelo controlador se algum *switch* SDN sair da rede.

O Ryu atingiu um total de 1017 *switches* SDN. Analisando o resultado do Ryu percebe-se porque o POX só vai até 1016 *switches* SDN. O limite de *sockets* possíveis no Python 2.7 é de 1017, a partir disto ocorre uma exceção. O POX limita o controlador a conectar apenas um *switch* SDN a menos que o limite possível, enquanto o Ryu não trata a exceção e por isto atinge o limite ao tentar abrir *sockets* demais. O Beacon segue o mesmo caminho que o Ryu, não tratando a exceção e para de aceitar conexões apenas com 1769 *switches* SDN, mas mantém normal o funcionamento da aplicação.

No NOX-MT não há mensagem indicando que não são aceitos mais *switches* SDN. O controlador aceita as primeiras 1015 conexões, enquanto outras conexões só ocorrem se algum *switch* SDN sair da rede ou fechar a conexão. O Floodlight foi o único

controlador que não apresentou um número exato na quantidade máxima de conexões estabelecidas (869 máximo e 829 mínimo, como apresentado na Tabela 3). Isto ocorreu por conta de uma falha inesperada no *handshake* do OpenFlow, que acarreta o fechamento de todas as conexões. Ao contrário dos casos anteriores, este fechamento não seguiu um padrão imediatamente claro, como o lançamento de uma exceção.

O Beacon proporcionou o melhor desempenho neste teste, atingindo a marca de 1769 conexões estabelecidas e mantidas com os *switches* SDN, pouco mais que 2x o desempenho do Floodlight (pior desempenho no teste). Entretanto, o Beacon não tratou a exceção que estabelece um limite máximo de *switches* SDN conectados, para evitar que a mesma ocorresse, como fazem o POX e NOX-MT.

4.3. Taxa de transferência com diferentes números de *switches* SDN

Este teste consiste em aumentar gradualmente o número de *switches* SDN executando o modo *throughput* do Cbench. Com isso é verificado o desempenho dos controladores ao aumentar o número de sessões realizando requisições contínuas. O número de *switches* SDN variou de 2^0 até 2^8, obtendo-se assim nove combinações para cada controlador. A Figura 2 apresenta as distribuições dos resultados para o POX, Ryu e Floodlight, este último sem resultados para 64, 128 e 256 devido ao fechamento súbito da conexão com os *switches* durante o teste por não suportar a quantidade de requisições.

Ao aumentar o número de *switches* SDN, o POX tem desempenho crescente até 128 *switches* SDN. Esse desempenho pode ser causado pelo uso do compilador JIT, o qual aumenta a velocidade de execução quando há código Python que é executado repetidamente. A distribuição com 128 *switches* SDN não possui diferença estatística da distribuição de 256 *switches* SDN. Entretanto, ao executar o teste de Wilcoxon [Montgomery et al. 2003], com a finalidade de verificar se a distribuição dos resultados de 128 *switches* SDN é melhor que a de 256, obtém-se um *p-value* de 0.04203. Portanto, o cenário com melhor desempenho para o POX é com 128 *switches* SDN, o qual obteve um máximo de 103819 fluxos/segundo.

O Ryu tem perda gradual de desempenho durante as variações de 1 a 256 *switches* SDN. Esta perda é ocasionada pelo processamento de mensagens de forma sequencial, comportamento comum ao aumentar a quantidade de *switches* conectados (também há aumento de 1000 *hosts* por *switch* SDN). O melhor desempenho do Ryu aconteceu com 1 *switch* (47631 fluxos/segundo), e durante todo o teste manteve a consistência de seus resultados não gerando nenhum *outlier*.

O Floodlight conseguiu se manter funcionando apenas até 32 *switches* SDN, entretanto, já apresentando mau funcionamento e com resultados dispersos e alguns *outliers*. O melhor desempenho aconteceu com 8 e 16 *switches* SDN (não houve diferença estatística entre as distribuições). No intuito de verificar qual é o melhor cenário entre 8 e 16 *switches* SDN, o teste T de Student foi realizado. O *p-value* foi 0.6126 para o 16 melhor que o 8, e 0.3874 para o 8 melhor que 16. Dessa forma não há uma distribuição melhor, segundo o teste T de Student [Montgomery et al. 2003]. No cenário com 8 *switches* SDN o desempenho do Floodlight chegou a 145155 fluxos/segundo; com 16 *switches* foram 144918 fluxos/segundo.

Os resultados do NOX-MT e do Beacon são apresentados na Figura 3. O NOX-MT finalizou abruptamente durante a execução do teste com 256 *switches* SDN, por isto

Figura 2. Taxa de transferência (POX, Ryu e Floodlight com 1 *thread* e 10^3 *hosts*)

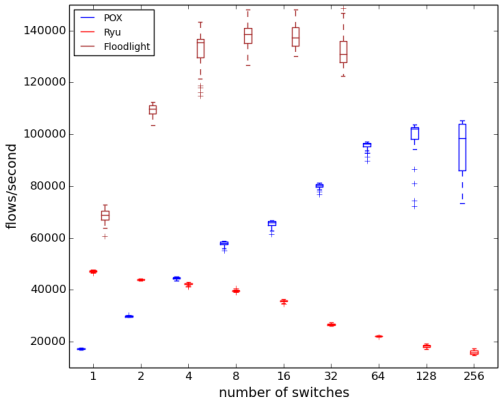
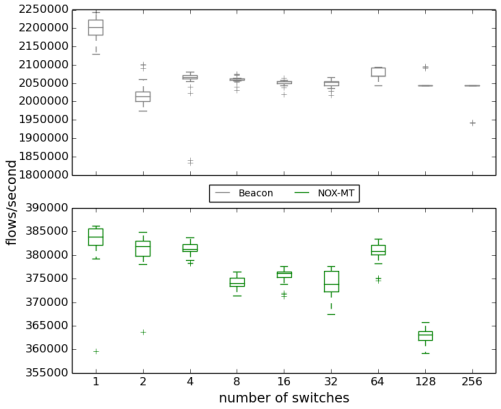


Figura 3. Taxa de transferência (NOX-MT e Beacon com 1 *thread* e 10^3 *hosts*)



não há resultados para esta combinação. Também é perceptível que seu desempenho degradou já em 128 *switches* SDN, o pior resultado do controlador neste teste. No cenário com 1 *switch* SDN foi observado o melhor desempenho, com até 386129 fluxos/segundo, com diferença estatística entre as distribuições comprovadas pelos testes de hipótese.

O Beacon atingiu a marca dos 2 milhões de fluxos/segundo em todas as variações de números de *switches* SDN, sendo o pico de seu desempenho com 1 *switch* (tal qual o Ryu e o NOX-MT) com 2223940 fluxos/segundo. A partir de 4 *switches* SDN o Beacon chegou ao ponto de saturação (resultados com distribuições estatisticamente iguais). Neste ponto em diante o desempenho do controlador se mantém em uma faixa na qual a quantidade de mensagens trocadas é aproximadamente a mesma.

Com base neste teste, e tendo os testes de hipótese como suporte, foi possível verificar os pontos de saturação dos controladores. Neste caso, apenas o Beacon mantém uma quantidade aproximadamente estável de mensagens trocadas após saturar, a partir de 4 *switches* SDN. Para alguns controladores, ao chegarem no ponto de saturação, o desempenho começa a cair, como no POX (satura com 128 *switches*) e o Floodlight (com 8). Aparentemente isto ocorre porque o controlador não consegue processar mais mensagens, e ao tentar processar todas as mensagens, degrada seu desempenho. O desempenho do Ryu tem decaimento mais gradual, o que decorre ao tentar processar todas as mensagens, o que degrada sua capacidade progressivamente ao ampliar o número de *switches* SDN realizando solicitações. O NOX-MT está em situação semelhante ao Ryu, no entanto, não conseguindo responder a mais requisições após atingir seu limite, acarretando assim no encerramento inesperado do controlador com 256 *switches* SDN.

4.4. Latência

A latência dos controladores é apresentada na Figura 4 com exceção do POX, que devido a uma escala 17 vezes maior dificultaria a visualização dos resultados do teste. Como pode ser visto, o Ryu obteve latência crescente, ao contrário do que ocorreu no teste de taxa de transferência, mostrando coerência no experimento uma vez que a taxa de transferência diminui quando a latência aumenta. Um *switch* SDN fazendo requisições é o melhor cenário do Ryu neste teste, com $21.0463\mu s$/fluxo.

O NOX-MT não parou de funcionar com 256 *switches* SDN como no teste da Subseção 4.3 (Taxa de Transferência) devido a diferença na carga de trabalho gerada. O teste de latência gera apenas uma requisição por *switch* SDN e aguarda a resposta do controlador para poder enviar outra. Assim, o controlador trabalha menos pressionado do que no teste de taxa de transferência, o qual mantém o *buffer* de entrada do controlador e de saída dos *switches* SDN cheios. Percebe-se que o comportamento irregular do NOX-MT no teste de Taxa de Transferência ocorreu devido à alta carga de trabalho gerada pelo teste.

Figura 4. Latência (Ryu, NOX-MT, Floodlight e Beacon com 1 *thread* e 10^3 *hosts*)

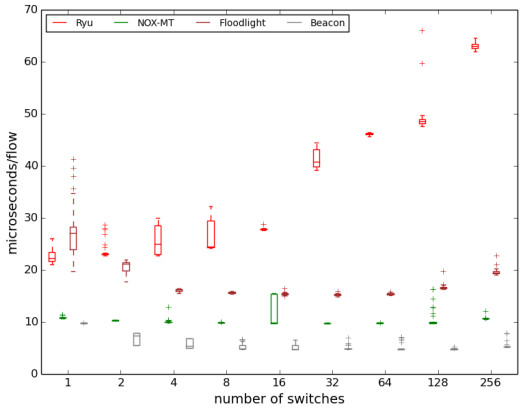
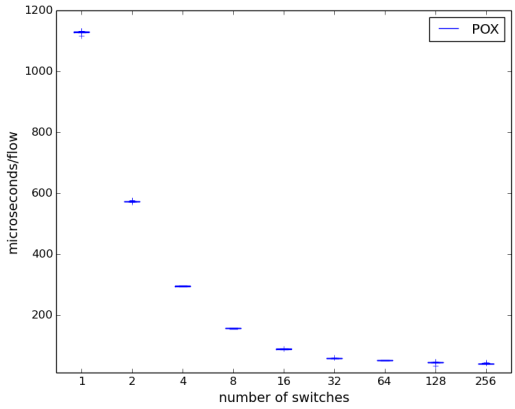


Figura 5. Latência (Controlador POX utilizando 1 *thread* e 10^3 *hosts*)



Não foi possível identificar o cenário com melhor desempenho do NOX-MT. Então, no intuito de identificar a menor latência para o NOX-MT, os testes de Wilcoxon ou T de Student são aplicados com os parâmetros de 8, 16, 32, 64 e 128 *switches* SDN. Os testes de hipótese indicam que com 32 *switches* SDN os *p-values* foram: 2.2e-16, 7.464e-06, 2.69e-05 e 2.987e-06. Portanto, a menor latência do NOX-MT está no cenário de 32 *switches* SDN com $9.6862\mu s$/fluxo.

O Floodlight apresentou comportamento semelhante ao NOX-MT. Neste teste, a baixa carga de trabalho permitiu ao Floodlight um funcionamento normal. No entanto, como apenas fechou as conexões no teste da Subseção 4.3, a carga não o afetou tanto quanto ao NOX-MT, que parou de funcionar por completo. Baseado nos resultados do Floodlight, na Figura 4, não é possível dizer qual a menor latência, então foram aplicados testes de hipótese para identificar o melhor cenário. Os *p-values* são 0.004084 e 0.002332, ao testar para 32 *switches* SDN contra 16 e 64 *switches* SDN, logo o melhor cenário foi com 32 *switches*, atingindo uma latência de $14.8589\mu s$/fluxo.

O Beacon teve redução gradual na latência no teste até 16 *switches* SDN. A partir daí é impossível distinguir resultados visualmente. Realizaram-se testes de hipótese para identificar a menor latência, porém, não houve resultado estatisticamente menor que o outro entre 16, 32, 64 e 128 *switches*, onde obtive-se uma latência miníma de, respectivamente, 4.6615, 4.7622, 4.7198 e $4.7466\mu s$/fluxo.

A latência do POX é apresentada na Figura 5 para melhor visualização dos resultados. Os primeiro 4 cenários tiveram resultados elevados se comparados aos demais

controladores. O POX apresenta uma queda da latência contínua até 256 *switches* SDN, porque todos valores são diferentes baseados nos testes estatísticos. A menor latência ocorreu com 256 *switches* SDN, atingindo $39.1023\mu s$/fluxo.

Percebe-se que a latência aparenta comportamento inverso ao teste da taxa de transferência ao alterar o número de *switches* SDN (comparando os gráficos nas Figuras 2 e 3 com as Figuras 4 e 5), reforçando o resultado de ambos os testes. Neste teste, o controlador Beacon alcançou a menor latência, com $4.6615\mu s$/fluxo, 8.38x menor que o POX, controlador com maior latência no teste em seu melhor cenário.

4.5. Discussão

Os testes realizados neste trabalho permitiram identificar o melhor cenário de cada controlador para algumas variáveis importantes em uma rede SDN, como número de *switches* SDN e tempo de processamento de mensagens. Estes cenários são sumarizados na Tabela 4. O controlador Beacon alcançou o melhor desempenho nos testes executados neste trabalho, como visto na tabela, tanto nos testes de taxa de transferência e latência, quanto no teste da proposta da IETF (*Control Session Capacity*). O alto desempenho do controlador podem ser atribuídos aos mecanismos de manipulação de eventos e escrita de mensagens OpenFlow, descritos em [Erickson 2013]. No Beacon, uma série de *threads* de entrada e saída leem as mensagens OpenFlow e as colocam em uma fila compartilhada, enquanto outra série de *threads* retiram as mensagens da fila e as designam ao local de processamento correspondente por tipo. Assim, as mensagens são processadas por um mecanismo específico para cada tipo.

Tabela 4. Cenários com melhores resultados dos controladores

Controladores/ Testes	**POX**	**Ryu**	**NOX-MT**	**Floodlight**	**Beacon**
Taxa de transferência	*switches*=128 103819 F/s	*switch*=1 47631 F/s	*switch*=1 386129 F/s	*switches*=8 145155 F/s	*switch*=1 2223940 F/s
Latência	*switches*=256 $39.1023\mu s$/F	*switch*=1 $21.0463\mu s$/F	*switches*=32 $9.6862\mu s$/F	*switches*=32 $14.8589\mu s$/F	*switches*=16 $4.6615\mu s$/F
Control Session Capacity	1016 *switches*	1017 *switches*	1015 *switches*	869 *switches*	1769 *switches*

Entretanto, o Beacon tem problemas com segurança relatados por Shalimov *et al*. [Shalimov et al. 2013] em seus testes. Este controlador processa mensagens OpenFlow malformadas sem parar a execução ou fechar a conexão, passando o erro despercebido pelo controlador, o que caracteriza uma possível vulnerabilidade. Essas falhas acontecem ao receber mensagens `Hello` com versão do OpenFlow inválida, cabeçalho com tipo da mensagem não correspondente ao conteúdo e ao identificar o pacote como protocolo ARP no cabeçalho e no conteúdo haver um pacote IP. Dessa forma, parte do desempenho do Beacon deriva da falta de checagens nas mensagens OpenFlow.

Por outro lado, o teste que expôs as falhas no processamento de mensagens do Beacon também permitiu uma análise mais informada dos resultados dos controladores Ryu e POX, uma vez que o Ryu foi o controlador mais seguro nos testes de Shalimov *et al*., passando em 80% deles, seguido do POX que passou com sucesso em 40% [Shalimov et al. 2013]. Estes controladores realizam mais checagens nas mensagens OpenFlow, o que demanda mais tempo de processamento por mensagem. Este é um

dos fatores que fazem estes controladores processarem uma quantidade menor de mensagens. O POX mostrou-se capaz de lidar com o aumento de carga (mais *switches* SDN realizando requisições), no entanto, obteve a maior latência dentre os controladores.

O Ryu obteve o pior desempenho no teste de taxa de transferência dentre os controladores testados. Um fator que contribuiu para isto foi o mecanismo de otimização do interpretador PyPy, que somente otimiza código Python, enquanto que o Ryu possui 12.5% do código escrito em Erlang. Contudo, com PyPy o controlador permanece com desempenho melhor que utilizando o interpretador nativo (CPython). Também ao tentar gerenciar toda a crescente demanda mensagens na entrada de seu *buffer*, o desempenho do Ryu degrada e acaba não mantendo a capacidade de gerenciamento inicial. No entanto, suportou a carga de trabalho de todos os testes, ao contrário do NOX-MT que se comportou de forma irregular por conta do aumento da carga de trabalho e conexões simultâneas no teste de taxa de transferência, que acarretou o fim abrupto de sua execução.

O Floodlight, ao ser submetido a uma alta carga de trabalho, fechou todas as conexões. Isto ocorreu no teste de taxa de transferência com diferentes números de *switches* SDN (acima de 32 *switches* conectados), o que pode acontecer devido à sobrecarga no canal de comunicação com o *switch* SDN. Constatou-se que há troca excessiva de mensagens OpenFlow ao executar qualquer operação no Floodlight. Neste caso foi observada a quantidade de dados (em mensagens OpenFlow) durante o teste *Control Session Capacity* da Subseção 4.2. O Floodlight, com 85.6% do desempenho do NOX-MT(controlador com segundo menor desempenho), transferiu 3x mais dados que o NOX-MT para estabelecer e manter a conexão com os *switches* SDN. Além disto, Fernandez [Fernandez 2013] relatou em seu trabalho que o Floodlight tem problemas com o gerenciamento de memória, consumindo uma quantidade atípica em relação ao total de mensagens trocadas.

5. Conclusão

A proposta da IETF é o único processo padronizado para avaliação de controladores SDN. Esta aborda maior número de características que o *benchmark* da literatura mais completo, contudo, não há trabalhos que executem seus testes na literatura ou ferramentas auxiliares para isto, porque ainda é complexo implementá-los. Um dos motivos é que não há padrão na API dos controladores, assim, mesmo os desenvolvidos com a mesma linguagem a programação é diferente para executar a mesma ação.

Este trabalho avaliou contém cinco controladores de arquitetura centralizada e código aberto: POX, Ryu, NOX-MT, Floodlight e Beacon. O teste *Control Session Capacity* da proposta da IETF foi executado para verificar uma característica não ainda abordada na literatura: o número máximo de conexões estabelecidas e mantidas entre controlador e *switches* SDN. Foi observado que os controladores POX e NOX-MT não permitem o estabelecimento de novas conexões ao atingirem seus respectivos limites. O Ryu e o Beacon, por outro lado, não tratam esta situação e só rejeitam conexões quando entram em uma exceção. Agora é possível evitar isto, pois sabe-se o limite de ambos. O Floodlight teve o pior desempenho no teste devido a instabilidade de conexões.

O NOX-MT e Floodlight não suportaram, em alguns casos, o estresse de uma alta carga de trabalho. No caso do NOX-MT o problema ocorreu apenas no teste de taxa de transferência com o maior número de *switches* SDN conectados, 256, cuja carga de trabalho fez este parar de funcionar. O Floodlight, por sua vez, fechou inesperadamente

todas as conexões com os *switches* SDN na maioria dos testes, por não suportar a carga de trabalho imposta. Além disto, este controlador demanda uma quantidade maior troca de mensagens OpenFlow para realizar suas ações, com fluxo de mensagens 3x maior que o NOX-MT no teste da proposta da IETF. Com isso o canal TLS para entre os *switches* SDN e o próprio controlador ficam sobrecarregados com mensagens de controle.

Ao observar o melhor cenário dos controladores percebe-se a discrepância entre o melhor e pior desempenho nos testes, chegando a ser de até 46.7x no teste de taxa de transferência. O melhor desempenho nesta avaliação deu-se com o controlador Beacon, o qual também obteve os melhores resultados nos demais testes. Além disto, ele apresentou um comportamento consistente, com pouca dispersão nos resultados. Entretanto, ainda há problemas na checagem de mensagens OpenFlow malformadas e não se sabe, quando resolvidos, o quão afetarão o desempenho final deste controlador.

Os resultados encontrados tornam evidente a necessidade de um processo sistemático de avaliação de controladores, no intuito de verificar seu comportamento em situações críticas antes de implementá-los em redes SDN reais. Esse processo também possibilita identificar pontos de melhorias e correções dos controladores por suas equipes de desenvolvimento. A proposta da IETF é um ponto de partida importante na avaliação de controladores, pois complementa e detalha os testes da literatura.

Referências

Alencar, F., Santos, M., Santana, M., and Fernandes, S. (2014). How software aging affects sdn: A view on the controllers. In *Global Information Infrastructure and Networking Symposium (GIIS), 2014*, pages 1–6. IEEE.

Erickson, D. (2013). The beacon openflow controller. In *Proceedings of the second ACM SIGCOMM workshop on Hot topics in software defined networking*, pages 13–18. ACM.

Erickson, D. (Acessado em: Setembro, 2017). What is beacon? Disponível em: `https://openflow.stanford.edu/display/Beacon/Home`.

Fernandez, M. P. (2013). Comparing openflow controller paradigms scalability: Reactive and proactive. In *Advanced Information Networking and Applications (AINA), 2013 IEEE 27th International Conference on*, pages 1009–1016. IEEE.

Floodlight, P. (Acessado em: Maio, 2017). Floodlight. Disponível em: `http://www.projectfloodlight.org/floodlight/`.

Foundation, A. S. (Acessado em: Agosto, 2017a). Licença apache 2.0. Disponível em: `http://www.apache.org/licenses/LICENSE-2.0.txt`.

Foundation, O. N. (Acessado em: Junho, 2017b). Openflow. Disponível em: `https://www.opennetworking.org/sdn-resources/openflow`.

Haleplidis, E., Pentikousis, K., Denazis, S., Salim, J. H., Meyer, D., and Koufopavlou, O. (2015). Rfc 7426 - software-defined networking (sdn): Layers and architecture terminology. Technical report, Internet Engineering Task Force.

Halpern, J. M. (2014). Standards collisions around sdn. *IEEE Communications Magazine*, 52(12):10–15.

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

McKeown, N., Anderson, T., Balakrishnan, H., Parulkar, G., Peterson, L., Rexford, J., Shenker, S., and Turner, J. (2008). Openflow: enabling innovation in campus networks. *ACM SIGCOMM Computer Communication Review*, 38(2):69–74.

Montgomery, D. C., Runger, G. C., and Calado, V. (2003). *Estatística aplicada e probabilidade para engenheiros*. Grupo Gen-LTC.

ONF (2009). Openflow switch specification version 1.0.0 (wire protocol 0x01). *Open Networking Foundation*.

Salman, O., Elhajj, I. H., Kayssi, A., and Chehab, A. (2016). Sdn controllers: A comparative study. In *2016 18th Mediterranean Electrotechnical Conference (MELECON)*, pages 1–6. IEEE.

Shah, S. A., Faiz, J., Farooq, M., Shafi, A., and Mehdi, S. A. (2013). An architectural evaluation of sdn controllers. In *Communications (ICC), 2013 IEEE International Conference on*, pages 3504–3508. IEEE.

Shalimov, A., Zuikov, D., Zimarina, D., Pashkov, V., and Smeliansky, R. (2013). Advanced study of sdn/openflow controllers. In *Proceedings of the 9th central & eastern european software engineering conference in russia*, page 1. ACM.

Sherwood, R. and KOK-KIONG, Y. (2010). Cbench: an open-flow controller benchmarker.

Society, I. (Acessado em: Junho, 2017). The internet engineering task force. Disponível em: `https://www.ietf.org/`.

Tootoonchian, A., Gorbunov, S., Ganjali, Y., Casado, M., and Sherwood, R. (2012). On controller performance in software-defined networks. In *Presented as part of the 2nd USENIX Workshop on Hot Topics in Management of Internet, Cloud, and Enterprise Networks and Services*.

Vengainathan, B., Basil, A., Tassinari, M., Manral, V., and Banks, S. (2018a). Benchmarking Methodology for SDN Controller Performance. Internet-Draft draft-ietf-bmwg-sdn-controller-benchmark-meth-08, Internet Engineering Task Force. Work in Progress.

Vengainathan, B., Basil, A., Tassinari, M., Manral, V., and Banks, S. (2018b). Terminology for Benchmarking SDN Controller Performance. Internet-Draft draft-ietf-bmwg-sdn-controller-benchmark-term-09, Internet Engineering Task Force. Work in Progress.

Zhao, Y., Iannone, L., and Riguidel, M. (2015). On the performance of sdn controllers: A reality check. In *Network Function Virtualization and Software Defined Network (NFV-SDN), 2015 IEEE Conference on*, pages 79–85. IEEE.