ENGENHARIA DE COMPUTAÇÃO

Gustavo Hermínio de Araújo

METODOLOGIA PARA AVALIAÇÃO DE DESEMPENHO EM *SWITCHES* OPENFLOW

Santa Cruz do Sul
2015

ENGENHARIA DE COMPUTAÇÃO

Gustavo Hermínio de Araújo

METODOLOGIA PARA AVALIAÇÃO DE DESEMPENHO EM *SWITCHES* OPENFLOW

Prof. Dr. Leonel Pablo Tedesco
Orientador

Santa Cruz do Sul
2015

ENGENHARIA DE COMPUTAÇÃO

Gustavo Hermínio de Araújo

METODOLOGIA PARA AVALIAÇÃO DE DESEMPENHO EM *SWITCHES* OPENFLOW

Prof. Dr. Juliano Araújo Wickboldt
Avaliador

Santa Cruz do Sul
2015

ENGENHARIA DE COMPUTAÇÃO

Gustavo Hermínio de Araújo

METODOLOGIA PARA AVALIAÇÃO DE DESEMPENHO EM *SWITCHES* OPENFLOW

Prof. M.e Lucas Fernando Muller
Avaliador

Santa Cruz do Sul
2015

Gustavo Hermínio de Araújo

METODOLOGIA PARA AVALIAÇÃO DE DESEMPENHO EM *SWITCHES* OPENFLOW

Trabalho de Conclusão II apresentado ao Curso de Engenharia de Computação da Universidade de Santa Cruz do Sul, como requisito parcial para a obtenção do título de Bacharel em Engenharia de Computação.

Orientador:Prof. Dr. Leonel Pablo Tedesco

Santa Cruz do Sul
2015

RESUMO

Software-Defined Networking (SDN) é um novo paradigma de redes de computadores que propõe separar a lógica de controle do encaminhamento de dados. Nesse paradigma, o controle da rede normalmente é logicamente centralizado permitindo que os dispositivos de encaminhamento sejam responsáveis apenas por encaminhar os pacotes de dados, de acordo com o que é definido por um entidade externa chamada de controlador. Ao projetor de uma rede SDN deve-se levar em consideração algumas particularidades de desempenho particulares deste paradigma, como por exemplo: tabela de fluxos, comunicação com controlador. Estas características tornam as métricas tradicionais de desempenho como vazão, latência, *jitter* e perda de pacotes insuficientes para avaliar o desempenho de um *switch* SDN. Dessa forma, é preciso o desenvolvimento de uma metodologia específica para testes de desempenho em *switches* e controladores SDN. Testes de conformidade para o protocolo Openflow já estão sendo definidos pela *Open Networking Foundation*, entretanto, testes de desempenho para *switches* baseados em Openflow ainda estão no início de sua especificação, e além disso, existem poucas ferramentas para testes de desempenho para *switches* SDN. Entretanto, essas ferramentas focam seus testes apenas na parte da lógica de controle da rede ou na parte de encaminhamento de dados, e não realizam uma avaliação analisando características do *hardware* do *switch*. Com base nesse contexto atual, esse trabalho de conclusão de curso propõe o desenvolvimento de uma metodologia que consiga avaliar o desempenho do *hardware* de *switches* SDN.

Palavras-chave: SDN, OpenFlow, Avaliação Desempenho, Redes de Computadores.

ABSTRACT

Software-Defined Networking (SDN) is a new computer network paradigm which proposes separating the control logic of the data traffic. In this paradigm, the network control is normally centralized allowing switches to be solely responsible for forward the data packet according to the one defined by the controller. In the construction of an SDN network we should take into account some peculiar performance aspects in the paradigm, such as: flow table, communicating with the controller, etc. These peculiarities make the traditional performance metrics, such as throughput, delay, jitter and packet loss, insufficient to assess the performance of an SDN switch. Thus, the development of a specific methodology for SDN switches and controllers performance tests is necessary. Conformance tests for the OpenFlow protocol are being defined by the Open Networking Foundation, however, OpenFlow-based switches performance tests are still in the beginning of their specification. Currently, there are few tools for SDN switches performance tests. Nevertheless, these tools focus their tests only on the control logic of the network or on the data traffic, and do not perform an assessment analyzing the switch hardware features. Based on this current context, this final paper proposes the development of a methodology which is able to assess the performance of a SDN switch hardware.

Keywords: SDN, OpenFlow, Performance Evaluation, Computer Network.

LISTA DE ILUSTRAÇÕES

LISTA DE TABELAS

LISTA DE ABREVIATURAS

SDN	*Software-Defined Networking*
SNMP	*Simple Network Management Protocol*
IP	*Internet Protocol*
MAC	*Media Access Control*
LAN	*Local Area Network*
VLAN	*Virtual LAN*
QoS	*Quality of Service*
MEF	*Metro Ethernet Forum*
ONF	*OpenFlow Networkin Foundation*
RFC	*Request For Commentes*
DUT	*Device Under Test*
RTT	*Round Trip Time*
TCP	*Transmission Control Protocol*
CPU	Central Processing Unit
ASIC	*Application Specific Integrated Circuits*

SUMÁRIO

1 INTRODUÇÃO

Software-Defined Networking é um paradigma emergente em redes de computadores que propõe uma plataforma flexível para o desenvolvimento de novas soluções para infraestruturas de redes. Uma de suas principais características está na separação da lógica de controle da rede (controlador) dos equipamentos que encaminham os pacotes através de um plano de dados (*switches* e roteadores). Com essa separação, os *switches* se tornam apenas equipamentos de encaminhamento de pacotes (KREUTZ et al., 2014). Para que essa separação ocorra, é necessário um protocolo de comunicação entre o plano de controle e o plano de dados. Com este propósito, o protocolo OpenFlow foi desenvolvido, sendo atualmente, a implementação mais relevante do paradigma SDN (MCKEOWN et al., 2008a). Um *switch* Openflow possui uma ou mais tabelas com regras para manipulação dos pacotes de dados (tabela de fluxos). Cada regra corresponde a um subconjunto do tráfego e uma sequência de ações de como lidar com os pacotes, como por exemplo, descartar, encaminhar ou modificar cabeçalho (ONF, 2013).

Com esse novo paradigma, as métricas tradicionais de avaliação de desempenho como latência, *jitter*, perda de pacotes e largura de banda se tornam insuficientes para analisar o desempenho de equipamentos de redes baseadas em SDN. Entre outras palavras, o desempenho de um *switch* OpenFlow não está ligado somente à vazão do tráfego de dados, mas também em como é feito o processamento do tráfego de controle. Por exemplo, é importante avaliar a quantidade de que o *switch* envia para o controlador por segundo que podem ser enviadas ao *switch* e adicionadas a tabela de fluxos. O envio de mensagens acima da quantidade suportada adicionara um atraso na transmissão dos pacotes do plano de dados, devido ao fato, do *switch* não tomar decisões sobre o que fazer com os pacotes. O *switch* deve enviar ao controlador e então o controlador toma uma decisão sobre o que fazer com este pacote. Deste modo, uma metodologia de avaliação de desempenho deve ser elaborada para esse novo paradigma de rede.

Atualmente, a literatura sobre SDN apresenta poucas soluções para a avaliação de desempenho. Tootoonchian *et al.*,2010, propõem um *framework* que avalia a atualização da tabela de fluxos e o atraso de pacotes que necessitam de modificação do cabeçalho para serem encaminhados. Jarschel *et al.*,2012, apresenta uma ferramenta para análise de controladores. Essa análise, verifica métricas como os pacotes por segundos enviados do controlador para o *switch*. Rotsos *et al.*,2011, desenvolveram uma ferramenta que permite a

avaliação do custo/benefício de diferentes arquiteturas de rede utilizando *Openflow*, como por exemplo, a comparação entre a utilização de um controlador centralizado e vários controladores distribuídos na mesma rede. Entretanto, esses estudos apresentam poucos experimentos realizados em *hardware*, ficando muitas vezes apenas em nível de simulação ou análises matemáticas.

Para este trabalho de conclusão de curso, desenvolveu-se uma metodologia para avaliação de desempenho em *switches Openflow* focada em avaliar elementos do *hardware* do *switch*. Para esta avaliação, foram desenvolvidos três tipos de teste: (*i*) análise da capacidade de armazenamento da tabela de fluxos, (*ii*) análise da quantidade de memória RAM utilizada para a instalação de fluxos e (*iii*) análise da carga da CPU. Ao final deste trabalho, são discutidos os resultados obtidos comparando a avaliação realizada em diferentes switches. Para a avaliação, foram utilizados tanto *switches* virtuais quanto *switches* reais.

O restante deste trabalho esta organizado da seguinte forma: no Capítulo 2 são apresentada as bases para realização deste trabalho. No Capítulo 3 é apresentada a proposta desse trabalho de conclusão de curso. No Capítulo 4 é apresentado a metodologia para a validação de desempenho. No Capítulo 5, são exibidos os resultados obtidos com a implementação da metodologia proposta. E finalmente, no capítulo 6 são exibidas as considerações finais e trabalhos futuros.

2 FUNDAMENTAÇÃO TEÓRICA

Neste capítulo são apresentados os conceitos básicos para a compreensão deste trabalho de conclusão de curso. O capítulo está organizado da seguinte forma: na Seção 2.1 são abordados os conceitos do paradigma SDN e o protocolo OpenFlow, na Seção 2.2 são analisados aspectos da avaliação de desempenho do protocolo OpenFlow e na Seção 2.3 são apresentados alguns trabalhos relacionados a este trabalho de conclusão.

2.1 *SDN/OpenFlow*

Como apresentado por Kreutz *et al.* (KREUTZ et al., 2014), as redes de computadores podem ser organizadas em três planos de funcionalidades: dados, controle e gerencia. O plano de dados representa os equipamentos usados para encaminhar os pacotes de dados, por exemplo, *switches* e roteadores. O plano de controle é representado pelos protocolos usados para que os equipamentos saibam como encaminhar o tráfego de dados, como por exemplo, BGP (*Border Gateway Protocol*), OSPF (*Open Shortest Path First*), etc. O plano de gerência que inclui serviços utilizados para monitorar e configurar remotamente a camada de controle, como por exemplo, SNMP (*Simple Network Management Protocol*). Em síntese, a plano de gerência define regras para a rede, o plano de controle implementa essas regras e a plano de dados encaminha o tráfego de acordo com as regras que foram definidas.

Em uma rede IP tradicional, os planos de dados e controle são acoplados e embarcados no mesmo equipamento de rede, com controle sendo descentralizado, fazendo com que os equipamentos tomem decisões sobre o encaminhamento de pacotes de maneira autônoma. Este tipo de abordagem torna a arquitetura de uma rede muito complexa. Como consequência, tem-se uma rede muito estática que dificulta o desenvolvimento de novos serviços. Visando permitir que soluções de rede sejam facilmente projetadas e desenvolvidas o paradigma SDN foi baseado nos seguintes pilares:

- Desacoplar plano de dados do plano de controle. Desta maneira, os equipamentos de rede se tornam apenas encaminhadores de pacotes, não tomando decisões autônomas.

- Lógica de controle é movida para uma entidade externa, centralizada, implementada em *software*. Esta entidade chama-se controlador e tem a funcionalidade de implementar

toda a lógica de controle da rede, isto faz com que, o desenvolvimento de novos serviços de rede sejam facilmente implementáveis.

- O encaminhamento de dados torna-se baseado em fluxos ao invés de ser baseado em destino. Um fluxo pode ser definido como um conjunto de campos de cabeçalhos utilizados para identificação (*match*) e um conjunto de ações (*action*) (KREUTZ et al., 2014).

Na arquitetura SDN, o plano de dados pode ser representado pelos equipamentos de rede como *switches* e roteadores, porém, nenhuma decisão de controle é tomada de maneira embarcada nestes equipamentos. Sendo assim, apenas implementa as regras definidas pelo plano de controle. O plano de dados torna-se responsável apenas por lidar com os pacotes de dados dos usuários e de garantir o suporte necessário para virtualização, conectividade e segurança (ONF, 2013).

O plano de controle, no paradigma SDN, é representado pelo controlador que é responsável pela inteligência da rede. Através de uma interface de comunicação, o plano de controle "programa" os equipamentos do plano de dados, de acordo com a necessidade da rede. Desta maneira os controladores são responsáveis por implementar os protocolos de controle da rede.

O paradigma SDN foi amplamente difundido, após a especificação do protocolo OpenFlow (MCKEOWN et al., 2008b). Este protocolo define uma interface entre o plano de dados e o plano de controle. O protocolo OpenFlow é implementado em ambos os planos, de forma que, o equipamento do plano de dados possui uma tabela que armazena regras de encaminhamento de dados, chamada tabela de fluxos. Este protocolo também prevê um canal seguro para comunicação entre *switch* e controlador, como por exemplo *Security Socket Layer*, assim como um conjunto de mensagens para manipulação da tabela de fluxos (MCKEOWN et al., 2008b).

A tabela de fluxos é composta por uma sequência de campos de cabeçalhos, como por exemplo: IP origem, IP destino, VLAN, etc. Cada entrada nesta tabela pode possuir um determinado valor para o cabeçalho ou o ignorar. Quando um pacote chega a um *switch* OpenFlow, primeiramente, verifica-se seus cabeçalhos para identificar se já existe uma entrada na tabela de fluxos (*match*). Caso exista, é aplicada uma ação ao pacote de dados que pode ser, por exemplo, encaminhar para uma determinada porta ou descartá-lo. Caso contrário, o pacote é encapsulado e enviado ao controlador (*table miss*), que define uma regra e a adiciona na tabela

de fluxos.

O protocolo *OpenFlow* possui três tipos de mensagens: controlador para o *switch*, assíncrona e simétricas (**?**). Mensagens do tipo controlador para *switch* são mensagens que o controlador envia para obter informações sobre o estado do *switch*, por exemplo: verificar estatísticas de um determinado fluxo. Essas mensagens são:

- ***Feature***: ao estabelecer uma conexão, o controlador envia esta mensagem requisitando que o *switch* informe suas capacidades.
- ***Configuration***: o controlador informa parâmetros de configuração para os *switches*.
- ***Modify-State***: utilizado pelo controlador para gerenciar o estado dos *switches*. Comumente usado para adicionar, deletar ou modificar fluxos na tabela de fluxos.
- ***Read-State***: utilizado pelo controlador para coletar estatísticas das tabelas de fluxos, portas, fluxos e filas do *switch*.
- ***Send-Packet***: utilizada pelo controlador para enviar pacotes por uma porta específica.
- ***Barrier***: mensagem utilizada para verificar se as dependências das mensagens foram alcançadas ou receber notificação sobre tarefas concluídas.

Mensagens assíncronas são enviadas do *switch* para o controlador para informar sobre eventos na rede e mudanças no estado do *swtich*, por exemplo: quando um novo fluxo chega ao *switch* e não existe uma entrada corresponde na tabela de fluxos ou para remoção de um fluxo. Essas mensagens são:

- ***Packet-In***: existem duas razões para o *switch* enviar esta mensagem: quando há uma ação explícita na tabela de fluxos para que seja enviado para o controlador ou quando não há um *match* para o pacote.
- ***Flow-Removed***: relativa a remoção de regras no *switch*.
- ***Port Status***: obtem *status* das portas do *switch*.
- ***Error***: o *switch* pode enviar mensagens para notificar problemas ao controlador por mensagens de erro.

Finalmente, mensagens simétricas são iniciadas por qualquer uma das partes sem nenhuma solicitação, por exemplo: início de conexão entre controlador e *switch*. Essas mensagens são:

- ***Hello***: esta mensagem é utilizada no início da conexão entre *switch* e controlador.
- ***Echo***: utilizado para obter informações sobre a conexão entre *switch* e controlador como: latência, largura de banda e conectividade.
- ***Vendor***: na versão 1.0.0 do protocolo *OpenFlow*, esta mensagem é utilizada para que os fabricantes possam adicionar novas funcionalidades utilizando este protocolo.

2.2 Avaliação de Desempenho

A avaliação do desempenho de um equipamento de rede é uma etapa fundamental de seu desenvolvimento e implantação. Na indústria, uma grande parte do desenvolvimento de um novo equipamento é destinado somente a validação e verificação do desempenho, pois, caso algum equipamento apresente um problema em campo, o fornecedor terá um custo extra com sua manutenção. A RFC 2544 (BRADNER; MCQUAID, 1999), propõe, uma metodologia de testes de desempenho com quatro testes: vazão, latência, taxa de perda de quadros e *back-to-back*. O teste de vazão avalia a capacidade máxima de quadros por segundo que podem ser encaminhados sem erros. O teste de latência avalia o tempo em que um quadro leva para percorrer a rede da origem até o destino. O teste de perda de quadros avalia a rede em condições de sobrecarga, verificando a taxa de quadros perdidos. Finalmente, o teste de *back-to-back* avalia a capacidade máxima de quadros que podem ser recebidos sem erros. Com estes testes, somente é avaliado a capacidade de encaminhamento de um *switch*, entretanto, outros aspectos são importantes para o protocolo *OpenFlow* como por exemplo, latência do canal de controle, tempo de instalação de um fluxo na tabela, quantidade máxima de fluxos que podem ser instalados por segundo.

O desempenho de um *switch* OpenFlow não está relacionado somente à vazão do tráfego de dados, mas também em como é feito o processamento do tráfego de controle. Por exemplo, invocar o controlador para instalação de fluxos pode vir a ser um problema de escalabilidade. Cada controlador pode lidar apenas com algumas centenas de fluxos por segundo. Como mostrado por Tootoonchian *et al.* (TOOTOONCHIAN et al., 2012), o controlador NOX consegue lidar com 30K fluxos por segundo com um tempo médio de instalação de fluxos de 10ms. Porém, alguns estudos mostram que em uma rede real o nível de desempenho exigido é bem maior. Kadula *et al.* (KANDULA et al., 2009) apresenta um cenário em que há um *cluster* com 1500 servidores possuindo uma média de 100K fluxos por segundo. Benson *et al.* (BENSON; AKELLA; MALTZ, 2010) mostra que uma rede com 100 *switches* pode possuir

picos com 10M fluxos por segundo. Tendo em vista os dados apresentados do controlador NOX, a rede possui uma tráfego de dados maior do que o controlador consegue gerenciar (TOOTOONCHIAN et al., 2012). Deste modo, uma metodologia de avaliação de desempenho deve ser elaborada para esse novo paradigma de rede. Esta nova metodologia deve abranger não somente a capacidade de encaminhar pacotes do *switch*, mas também, o impacto que o tráfego de controle tem sobre o tráfego de dados.

2.3 Trabalhos Relacionados

Nesta seção são descritos os principais trabalhos relacionados a este trabalho de conclusão de curso. Os trabalhos são: (*i*) Cbench, para a avaliação de desempenho de controladores, (*ii*) OFCBenchmark, também para avaliação de controladores e (*iii*) Oflops, uma plataforma para desenvolvimento de testes em *switches* OpenFlow.

2.3.1 Cbench

Cbench é uma ferramenta para a avaliação de desempenho de controladores *Openflow*. Através desta ferramenta é possível avaliar a latência de processamento do controlador para requisições do tipo *Packet-in* enviadas do *switch* para o controlador, bem como quantidade máxima de mensagens *Packet-in* que o controlador consegue suportar por segundo. O teste de latência é realizado enviando mensagens *Packet-in* para o controlador, uma nova mensagem só é enviada quando a anterior for respondida. Para avaliar a vazão, cada *switch* envia várias requisições paralelamente ao controlador. Além disso, através desse *framework* é possível utilizar um modo híbrido em que são enviados uma quantidade configurável de fluxos, de maneira paralela, para o controlador.

O Cbench possui a desvantagem de simular apenas um *switch* conectado a um controlador. Isso não permite que o controlador seja avaliada conectado com vários *switches* em uma rede. Outra desvantagem é que o Cbench não fornece as estatísticas de cada *switch*, apenas estatísticas gerais. Sendo assim, na próxima subseção é discutido o OFCBenchmark, uma ferramenta que permite a um cenário de testes com vários *switches* conectados em um controlador centralizado (JARSCHEL et al., 2012).

2.3.2 OFCBenchmark

Assim como o Cbench, o OFCBenchmark é uma ferramenta para avaliação de desempenho de controladores. Esta ferramenta consegue emular um conjunto de *switches* independentes, possibilitando configurar um cenário específico, de modo que, cada *switch* mantenha suas próprias estatísticas (JARSCHEL et al., 2012). OFCBenchmark foi desenvolvido para fazer medições de RTT e pacotes enviados/recebido por segundo para cada *switch* na rede. Isso permite verificar como o controlador gerencia cada *switch*.

O OFCBenchmark consiste em três componentes: *OFCBenchmark Control*, *OFCBenchmark Client* e *Virtual Switch*. *OFCBenchmark Control* consiste em uma interface gráfica em que o desenvolvedor pode realizar as configurações dos testes. Dentre as configurações possíveis estão: número de switches virtuais, tempo de intervalo de transmissão dos pacotes do *switch* para o controlador e tamanho dos pacotes. *OFCBenchmark Client* é responsável por criar e configurar a topologia dos *switches* virtuais. Por último, os *Virtuals Switches* são responsáveis por realizar as conexões com o controlador OpenFlow, sendo que, cada *switch* possui uma conexão individual com o controlador. Além disso, cada *switch* pode gerar tráfego de controle independentemente. Desta maneira, é possível configurar uma rede virtual com uma grande quantidade de *switches*, limitado apenas, pela quantidade de memória que a máquina utilizada possui (JARSCHEL et al., 2012).

Comparando com o CBench, esta ferramenta pode simular mais facilmente uma rede real, uma vez que, o CBench utiliza apenas uma conexão com o controlador. Outro ponto a favor, é que o OFCBenchmark fornece as estatísticas de cada *switch* na rede, assim sendo, é possível ter mais detalhes de como o controlador se comporta, dependendo do tráfego de dados. Porém, ambas as ferramentas realizam testes utilizando apenas *switches* virtualizados, desta maneira, não é possível obter dados de como o hardware se comporta de acordo com o controlador.

2.3.3 Oflops

O *framework* Oflops possui uma grande quantidade de ferramentas para avaliação de desempenho em *switches* OpenFlow (ROTSOS et al., 2012). Com esta ferramenta é possível verificar a capacidade dos *switches* de encaminhar os pacotes de dados, o tráfego do canal de controle, latência para instalação de um novo fluxo, além de monitorar o tráfego. Essa ferramenta também possibilita a geração de tráfego de dados e tráfego de controle definidos pelo usuário. Oflops possui uma arquitetura modular, permitindo que novos testes sejam adicionados

ao *framework* de maneira rápida (ROTSOS et al., 2012).

A arquitetura do *framework* Oflops é ilustrada na Figura 1. Esta arquitetura é organizadas em três níveis, sendo que, o nível 1 é responsável somente por gerenciar o teste que está sendo feito. A plataforma também fornece suporte ao protocolo SNMP, disponibilizando uma série de MIBs (*Management Information Base*) que permite a leitura de contadores como por exemplo, utilização da CPU, contadores de pacotes, etc. O nível 2 fornece para o usuário ferramentas que possibilitam a configuração de tráfego de dados, tráfego de controle, além da abstração da implementação de *drivers* para a transmissão de tráfego. E finalmente, o nível 3 da arquitetura é responsável por transmitir o tráfego de dados e de controle. O *framework* possibilita ainda, a configuração de vários canais para transmissão de dados e um canal para o controle.

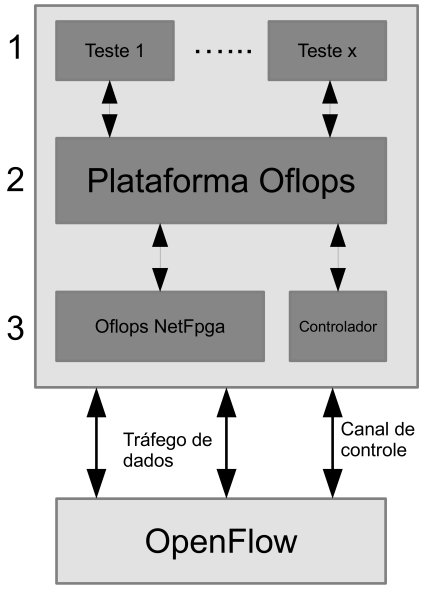

Figura 1: Arquitetura do Oflops

2.3.4 Comparação dos Trabalhos Relacionados

Na Tabela 1 é possível visualizar uma comparação entre os trabalhos relacionados a este trabalho de conclusão de curso. É possível perceber que apenas o framework Oflops fornece testes para a avaliação de desempenho de *switches* OpenFlow, enquanto, Cbench e OFBenchmark focam-se em testes para controladores. Atualmente, existem poucas ferramentas que avaliam o comportamento de *switch* OpenFlow, mesmo o framework Oflops oferece pouco monitoramento quanto a utilização do hardware dos switches. Outro ponto interessante é integrar o monitoramento de hardware ao controlador, fornecendo mais uma camada de

informação ao gerente da rede.

Ferramenta	Principal Funcionalidade	Descrição
Cbench	Avaliação de Controladores	Pode simular switches que enviam mensagens para sobrecaregar o controlador.
OFBenchmark	Avaliação de Controladores	Simula diferentes *switches* com suas próprias configurações.
Oflops	Avaliação de *Switches OpenFlow*	Framework que disponibiliza vários testes tanto para controladores quanto para o *switches*.

Tabela 1: Comparativo das características dos trabalhos relacionados

O Oflops também não realiza testes que possibilitam descobrir qual a quantidade máxima de fluxos que podem ser instalados na tabela de fluxos. Através deste teste pode-se saber qual a quantidade de fluxos que um switch consegue gerenciar. Se essa quantidade for muito pequena para a quantidade de fluxos que a rede exige, então haverá uma sobrecarga no canal de controle. Para cada pacote que não esteja na tabela de fluxos, será gerado uma mensagem de *Packet-in* e uma de *Packet-out*. Além disso, novas políticas para remoção de fluxos da tabela devem ser utilizadas, como por exemplo, diminuir o tempo que um fluxo fica ativo na tabela de fluxos. Essas limitações fazem com que o Oflops não consiga abranger todos os aspectos necessários para uma análise de desempenho completa. Dessa forma, o próximo capítulo apresenta as definições de novos testes para complementar a metodologia definida pelo Oflops.

3 PROPOSTA

Tendo em vista o que foi apresentado mo capítulo anterior, nota-se que existem poucas ferramentas para avaliação de desempenho focadas em avaliar o dispositivo que encaminha os pacotes de dados (*switch*). A maioria foca-se em avaliar os controladores e utiliza *switches* virtuais para simular topologias e então verificar a escalabilidade de determinado controlador. Já as metodologias focadas na avaliação dos *switches*, como Oflops, focam-se em em verificar a latência que o canal de controle possui ao instalar uma nova regra. Nenhuma das metodologias citadas apresentada testes de desempenho focados em características do hardware do *switch*, como por exemplo: utilização da memória conforme fluxos são instalados, monitoramento da CPU na coleta de estatísticas, etc. Algumas das metodologias citadas utilizam o protocolo SNMP para coletar informações referentes ao teste que esta sendo realizado, por exemplo quantidade de pacotes enviados em uma determinada interface, quantidade de memória utilizada, etc. Entretanto, muitos *switches* OpenFlow não possuem suporte a este protocolo, portanto, uma nova forma de coletar essas informações deve ser elaborada.

Este trabalho de conclusão de curso propõem uma alternativa para complementar as metodologias de avaliação de desempenho já existentes. A metodologia proposta foca-se em testes que realizam o monitoramento de recursos do hardware, como por exemplo: memória utilizada quando uma determinada quantidade de fluxos são instalados. Também é proposto uma forma alternativa ao uso do protocolo SNMP para coletar as estatísticas dos testes. Caso o *switch* não suportar este protocolo, então, tais testes não podem ser realizados. Então, para abranger uma maior quantidade de *switches* OpenFlow, a metodologia proposta não utiliza o protocolo SNMP, ao invés disso, é proposto uma forma alternativa para coletar as informações, como por exemplo: a API CLI que o equipamento possui. E finalmente, propõe-se que os testes sejam implementados em um controlador. Desta forma, pode-se realizar o monitoramento do hardware enquanto o *switch* esta em campo.

Este capítulo esta organizado da seguinte forma: na seção 3.1 é apresentado o teste que avalia a quantidade de fluxos que podem ser instalados na tabela de fluxos, na Seção 3.2 é apresentado o teste que monitora e avalia a utilização da memória RAM do *switch* e por fim, na Seção 3.3 é apresentado o teste que monitora e avalia a utilização da CPU.

3.1 Capacidade Máxima de Regras da Tabela de Fluxos

A verificação da quantidade máxima de regras na tabela de fluxos não é abordado nas metodologias apresentadas no capítulo 2. Este teste visa verificar a quantidade de fluxos que podem ser instalados no DUT. Saber a quantidade máxima de fluxos suportados pelo *switch* é necessária, uma vez que, caso a tabela de fluxos não possua uma capacidade suficiente para comportar os diferentes fluxos do plano de dados, resultará em uma grande quantidade de troca de mensagens entre controlador e *switch*, e como consequência teremos o *delay* da instalação das novas regras na tabela de fluxos adicionado ao tráfego de dados. Além disso, novas políticas para remoção de de fluxos da tabela devem ser utilizadas, como por exemplo, diminuir o tempo que um fluxo fica ativo na tabela de fluxos para garantir que novos fluxos possam ser instalados.

A figura 2 ilustra o cenário para a avaliação da tabela de fluxos. Nota-se que este teste possui um cenário simples para a avaliação e pode ser realizado somente com o controlador e o *switch*, sem a necessidade de um equipamento especial para gerar tráfego de controle. Através deste teste pretende-se verificar a capacidade máxima da tabela de fluxos para uma quantidade X de *matchs*, por exemplo: quantas regras podem ser instaladas para o seguinte *match*: *dl_src=00:01:02:03:04:05, dl_vlan=10:11:12:13:14:15*. Desta maneira pretende-se descobrir também a existência mecanismos de otimização para determinadas combinações de *matchs*, como por exemplo: otimizações para camadas L2 e L3.

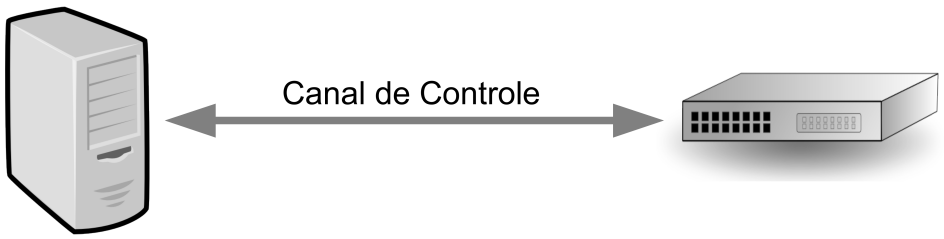

Figura 2: Cenário para o teste da quantidade máxima de fluxos

3.2 Utilização da Memória

Atualmente, a maioria dos *switches OpenFlow* utilizam memórias TCAM para implementação da tabela de fluxos, por serem mais eficientes para as rotinas de match. Porém,

este tipo de memória apresenta um alto custo ao projeto e uma capacidade de armazenamento pequena, geralmente entre 4K até 32K regras (KREUTZ et al., 2014). Para contornar esse problema é utilizado um *design* hibrido, utilizando um tipo de memória diferente, normalmente de baixo custo, para dar suporte a pequena capacidade de armazenamento da memória TCAM (KREUTZ et al., 2014). Uma alternativa é a utilização da memórias RAM, para dar suporte a TCAM. Tento em vista esse contexto, a análise da memória utilizada pelo *switches* pode identificar possíveis picos de utilização da memória.

Esta análise é necessária para identificar como é o aumento do consumo da memória conforme novos fluxos são instalados e estatísticas são coletadas. Caso o *switch* necessite de uma quantidade de memória acima da que esta disponível, isso pode ocasionar em aumento no tempo para instalação de fluxos ou, no pior dos casos, no não funcionamento do *switch*. Estes testes permitem identificar limitações como estas e apontar possíveis pontos de falha na implementação dos agentes OpenFlow.

Inicialmente os testes foram planejados para utilizarem o protocolo SNMP para coletar as informações da utilização de memória: Porém, poucos *switches* OpenFlow disponibilizam este protocolo para gerenciamento. Para contornar este problema foi utilizado no controlador *Floodlight* um mecanismo que coleta essas informações através do protocolo *Telnet* utilizando o CLI (*Command Line Interface*) disponibilizado pelo fabricante do *switch*. Na figura 3 pode-se observar o cenário utilizado para esta categoria de testes. O Canal de controle realiza a instalação de fluxos na tabela do *switch* e através do protocolo SNMP ou Telnet as estatísticas de memória são coletadas.

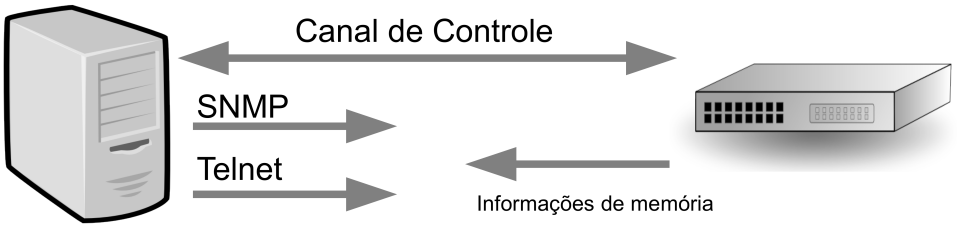

Figura 3: Cenário de testes de utilização da memória.

3.3 Utilização da CPU

A análise da utilização da CPU em *switches Openflow* também é pouco abordada nas metodologias apresentadas no capítulo 2. Em *switches Openflow* os controladores instalam novos fluxos em resposta a mensagens do tipo *Packet-in* enviadas pelo *switch*. Quando um novo fluxo começa a passar pelo plano de dados o tráfego de controle e a utilização da CPU podem ter picos de utilização. Isso faz com que o canal de controle fique instável (**?**). Saber o impacto deste pico de utilização de CPU é imprescindível para detectar problemas que podem ocasionar com que o *switch* fique indisponível, ou o impacto que este comportamento terá no tráfego de dados e na disponibilidade do canal de controle.

Estes testes pode identificar a necessidade de uma otimização nos mecanismos fornecedores de estatísticas de para cada fluxo, uma vez que, uma tabela de fluxos pode possuir uma grande quantidade de fluxos instalados. Caso o controlador requisite as estatísticas de vários fluxos paralelamente, isso poderá afetar o tráfego de dados e a latência de instalação de novas regras. Desta maneira, estes testes podem fornecer informações sobre o comportamento da CPU nos momentos de pico.

Neste capítulo foi apresentada a propostas dos testes para avaliação de desempenho. O próximo capítulo apresenta a metodologia utilizada assim como detalhes da implementação.

4 METODOLOGIA

Este capítulo tem o objetivo de descrever a metodologia utiliza para a avaliação de desempenho proposta no capítulo 3. Para que os objetivos fossem alcançados, três premissas básicas foram seguidas: (*i*) os teste devem ser implementados em um controlador, desta forma, é possível utilizar as funcionalidades de monitoramento enquanto o *switch* está sendo utilizado em uma rede real; (*ii*) os testes devem apresentar cenários de teste simples; (*iii*) utilizar uma forma alternativa ao protocolo SNMP para coleta de estatísticas, isso justifica-se, porque muitos *switches Openflow* não possuem este protocolo em suas implementações.

Para alcançar esses objetivos foi escolhido o controlador Floodlight (NETWORK, 2014). Este controlador esta disponível pela licença Apache (APACHE, 2015), podendo ser utilizado livremente tanto para aplicações acadêmicas quando para aplicações privadas. Além disso, é um controlador muito difundindo dentro da comunidade SDN possuindo uma boa documentação sendo de fácil utilização. E por fim, este controlador possui uma API JSON permitindo uma fácil interação de maneira remota. Neste trabalho de conclusão de curso, esta API foi utilizada para a passagem de parâmetros, inicialização e coleta de resultados. A figura 4 apresenta um fluxograma de como é feito a requisição de um teste e como o retorno dos resultados é realizado.

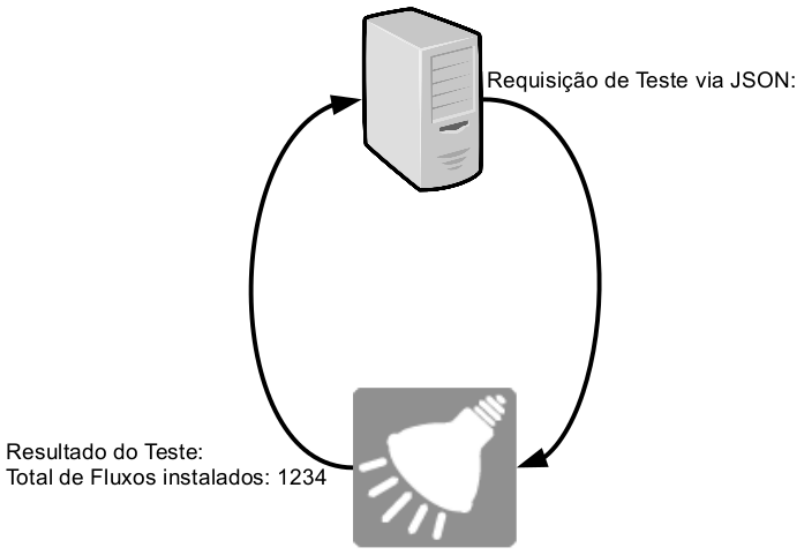

Figura 4: Comunicação entre PC e Floodlight

Como alternativa ao protocolo SNMP, os testes implementados neste trabalho de

conclusão de curso utilizam o protocolo Telnet (POSTEL; REYNOLDS, 1983). Desta maneira, é possível acessar as informações relevantes ao testes, por exemplo quantidade de memória utilizada, através da API CLI disponibilizada pelo fabricante. Porém, os testes ainda possui a limitação do *switch* possuir algum mecanismo que informe as estatísticas relevantes para o teste.

A sequencia deste capítulo são apresentados detalhes das implementações dos testes propostos no capítulo 3. Na seção 4.1 é apresentado o teste relacionados a quantidade de fluxos que podem ser instalados na tabela de fluxos, na seção 4.2 e finalmente, na seção 4.3 é apresentado o monitoramento da CPU.

4.1 Capacidade máxima da tabela de fluxos

Este teste visa verificar a capacidade de armazenamento de regras da tabela de fluxos. Desta forma, pretende-se verificar, além da capacidade máxima de fluxo, se a quantidade de *matches* e/ou uma determinada combinação de *matches* podem influênciar na quantidade de regras que a tabela pode comportar. Como parâmetros o teste possui:

- **DPID**: Datapath Identifier do *switch* que o teste será feito.

- **Quantidade máxima de fluxos**: define quantos fluxos vão ser instalados ou se o teste será feito até o *switch* retornar que a tabela esta lotada.

Para gerar novos fluxos para a instalação é utilizado um algoritimo que incrementa o MAC destino e o MAC origem. Este mecanismo é utilizado, pois, o endereço MAC possui 48 bits de endereçamento, isto permite uma grande combinação de fluxos diferentes sejam instalados como por exemplo, um fluxo com MAC origem: 00:00:11:11:22:22 e vlan: 100. A *action* utilizada é o direcionamento do tráfego para uma determinada porta. Para realizar o teste, o *switch* não deve possuir nenhuma conexão em suas interfaces. Isso previne que o *switch* solicite a instalação de algum novo fluxo ao controlador.

O teste é realizado da seguinte maneira, primeiramente a tabela de fluxos é limpa. Em seguida, envia-se para o *switch* uma mensagem do tipo *flow add*. Após isso, aguarda-se um momento para verificar se o *switch* não retorna informa que a tabela está lotada. Posteriormente, incrementa-se os endereços MAC destino e origem. Este processo é repetido até que mensagem de erro ocorra ou quantidade máxima de fluxos definida no teste seja atingida.

4.2 Monitoramento de Memória

Este teste visa verificar o consumo da memória do *switch*. Este teste segue o mesmo modelo do teste apresentado na sessão 4.1. São utilizados como parâmetros:

- **DPID**: Datapath Identifier do *switch* que o teste será feito.
- **Quantidade máxima de fluxos**: define quantos fluxos vão ser instalados ou se o teste será feito até o *switch* retornar que a tabela esta lotada.
- **IP de Gerencia**: IP da utilizado pela interface de gerencia do *switch*. Este IP é utilizado para acessar o switch via *Telnet*.

O teste é realizado da seguinte maneira, primeiramente a tabela de fluxos é limpa. Em seguida, coleta-se a memória em uso pelo *switch*. Logo após, envia-se para o *switch* uma mensagem do tipo *flow add*. Posteriormente, incrementa-se os endereços MAC destino e origem. Este processo é repetido até que a quantidade máxima de fluxos definida no teste seja atingida.

4.3 Monitoramento da CPU

Este teste visa monitorar a utilização da CPU durante determinado período de tempo. Esta análise é feita utilizando o protocolo *Telnet* para coletar as informações da carga da CPU. Porém, assim como nos testes apresentado na sessão 4.2 Este teste possui a limitação de que o fabricante deve fornecer alguma ferramenta que informa a carga da CPU. Os parâmetros para este teste são:

- **DPID**: Datapath Identifier do *switch* que o teste será feito.
- **IP de Gerencia**: IP da utilizado pela interface de gerência do *switch*. Este IP é utilizado para acessar o switch via *Telnet*.
- **Time**: intervalo de tempo (em segundos) que a informação de carga da CPU será realizada.
- **Quantidade máxima de fluxos**: define quantos fluxos vão ser instalados para terem estatísticas coletadas.

Para a análise da sobrecarga da coleta de estatísticas, primeiramente, a tabela de fluxos é limpa em seguida, são instalados uma quantidade de fluxos especificados pelo

teste, posteriormente, é iniciado o monitoramento da carga da CPU, desta forma, é enviado periodicamente uma mensagem para o *switch* informar o *status* atual da carga da CPU. Paralelamente a isto, é feita a requisição das estatísticas de cada fluxo. Finalmente, como resultado, o teste deve informar a progressão da carga da CPU conforme o intervalo de tempo do *status* foi requisitado.

Com os testes apresentados neste capítulo pretende-se fornecer dados suficientes para uma análise sobre o comportamento dos recursos de hardware de um *switch Openflow*. No próximo capítulo são apresentados os resultados obtidos aplicando a metodologia apresentada.

5 RESULTADOS OBTIDOS

Este capítulo apresenta os resultado obtidos com a implementação da metodologia apresentada no capítulo 4. Para coleta e análise dos resultados, foram escolhidos quatro *switches*, onde os testes foram efetuados. Estes *switches* foram divididos em dois grupos: (*i*) *switches* virtuais e (*ii*) *switches* reais. O primeiro grupo, diz respeito a uma implementação em software que emula um *switch* real. Os *switches* utilizados para este grupo foram *OpenvSwich* (OPENVSWITCH, 2015) e a implementação do *switch* de Stanford utilizando NetFPGA (NAOUS et al., 2008). A Tabela 2 apresenta a comparação entre os *switches* virtuais utilizados, assim como, algumas informações das máquinas hospedeiras de cada um deles. Em todos os *switches* foi utilizado a versão 1.0 do protocolo *Openflow*

Características	**OpenvSwitch**	**Stanford**
Capacidade da Tabela de Fluxos	100K	32K
CPU	Core Intel i5-3337U	Intel Core 2 Duo
Memória	6G	3G
OS	Ubuntu 12.04	CentOs 4
Interfaces	4 x 10/100BASE-T RJ45 (todas Virtuais)	4 x 1000BASE-T RJ45 (NetFPGA)
Capacidade de emcaminhamento	Não especificado	1000M

Tabela 2: Comparativo entre *switches* virtuais

O segundo grupo, diz respeito a *switches* reais que são comercializados. A Tabela 3 mostra um comparativo entre os *switches* escolhidos, as informações foram retiradas da especificação disponibilizada pelo fabricante de cada switch. Para o *Switch* A foi utilizado o *firmware* Indigo (NETWORK, 2015) e para o *Switch* B foi utilizado o *firmware* original de fábrica.

Pode-se observar em uma primeira análise que os *switches* virtuais apresentam uma maior capacidade de armazenamento de fluxos, memória e maior robustez quanto ao seus processadores. Isso diz respeito ao fato dos *switches* virtuais estarem embarcados em máquinas de propósito geral, com uma grande quantidade de recursos disponíveis. Por outro lado, os *switches* reais possuem uma maior capacidade de encaminhamento de dados por possuírem uma maior quantidade de interfaces. Nas próximas seções serão discutidos os resultados obtidos, após a aplicação da metodologia de testes especificada no capítulo 4 e será feita uma

Feature	Switch A	Switch B
Capacidade da Tabela de Fluxos	Não especificado pelo fabricante	2.5K
CPU	P2020	Cavium CN5010
Memória	2G	256M
OS	Indigo	Debian
Interfaces	48 x 10/100/1000BASE-T RJ45 4 x 1 GbE (SFP)	48 x 100/1000BASE-T RJ45 4x 1 GbE (SFP)
Capacidade de emcaminhamento	132G	176G

Tabela 3: Comparativo entre *switches* reais

comparação entre os resultados obtidos com as especificações de cada *switch*. Na Seção 5.1 são apresentados os resultados referentes a capacidade da tabela de fluxos. E na Seção 5.2 são apresentados os resultados referentes aos testes de utilização da memória.

5.1 Capacidade da Tabela de Fluxos

Para a apresentação dos resultados obtidos, primeiramente será mostrado o resultado individual de cada *switch* e em seguida a comparação de desempenho de cada um. Para realizar os experimentos e verificar se a quantidade de *matches* que um fluxo possui influência na capacidade máxima da tabela de fluxos, primeiramente, foram definidos seis tipos de *matches*:

1. **Tipo 1**: apenas um campo é utilizado para para gerar o *match* na tabela de fluxos. Através deste teste pretende-se alcançar a capacidade máxima da tabela de fluxos. Para realizar este *match* foi escolhido o campo de MAC Destino do cabeçalho *Ethernet*. Sua escolha deve-se ao fato de ser um campo de endereçamento que possui 6 *bytes*, tornando este campo de fácil variação e com grande disponibilidade de endereços.

2. **Tipo 2**: neste tipo dois campos são variados para gerar o *match*. Para este tipo, foram escolhidos MAC origem e destino. Este teste tem como objetivo verificar o comportamento da tabela de fluxos ao adicionar mais um campo para *match*.

3. **Tipo 3**: neste tipo são utilizados para o *match* os campos do cabeçalho Ethernet: MAC destino, MAC origem, VLAN e VLAN Priority. O objetivo deste tipo de *match* é verificar o comportamento da tabela de fluxos para um *match* exclusivamente na camada L2.

4. **Tipo 4**: neste tipo são utilizados os mesmo campos utilizados no Tipo 3 e são adicionados ao *match* os campos de IP origem e IP Destino. Através deste teste, pretende-se verificar

se a adição de campos do cabeçalho IP influem na quantidade máxima de fluxos que podem ser instalados na tabela de fluxos.

5. **Tipo 5**: nesta avaliação são utilizados os mesmos campos do tipo 4, porém, é adiciona o campo type do cabeçalho Ethernet. Tenta-se verificar com este teste se existe algum tipo de otimização para *matches* que utilizam cabeçalho IP (valor de *type* igual a 0x0800).

6. **Tipo 6**: *match* realizado com todos os 12 campos possíveis do protocolo *Openflow* versão 1.0.0. o objetivo deste teste é verificar o comportamento da tabela de fluxos para o tipo de *match* que possui a maior combinação de campos.

5.1.1 Switch A

Com os testes realizados no *Switch* A, pode-se notar que a quantidade de campos utilizados para *match* influencia na capacidade total de armazenamento de regras da tabela de fluxos. Na Figura 5 são exibidos os resultados obtidos. Com a análise do gráfico percebe-se que existe uma relação entre a quantidade de fluxos que podem ser instalados e a quantidade de campos utilizados para *matches*. Quanto maior a quantidade de campos necessários para realizar um *match*, menor a quantidade máxima de fluxos que podem ser instalados. Percebe-se também que para *matches* utilizando cabeçalho L2 e IP origem e destino (Tipo 4) há uma grande queda na quantidade de fluxos que podem ser instalados. Entretanto, ao utilizar o campo *type* do cabeçalho Ethernet a quantidade máxima de fluxos instalados aumenta. Isso acontece devido a algum mecanismo de otimização para o armazenamento de fluxos na tabela de acordo com os campos utilizados para *match*.

5.1.2 Switch B

Os testes realizados no *Switch* B, mostram que a alteração da quantidade de campos utilizados para *matches* não influencia na capacidade máxima de fluxos que podem ser instalados. O fabricante deste *switch* especifica uma quantidade total de fluxos instalados na tabela de aproximadamente 2500 fluxos. Os testes realizados detectaram a instalação de 2560 fluxos. Um número de fluxos maior que a especificado pelo fabricante que se manteve constante mesmo com a variação da quantidade de campos utilizados para *match*.

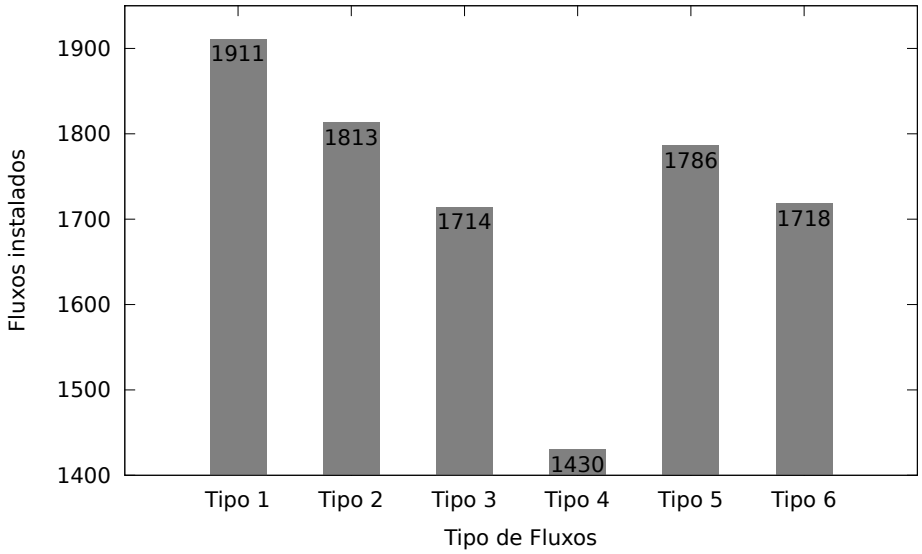

Figura 5: Comparativo entre tipos de *match* e quantidade de fluxos instalados para *switch A*

5.1.3 OpenvSwitch

O OpenvSwitch demonstrou possuir uma grande capacidade de armazenamento de fluxos. No total foram instalados 100K. Este comportamento acontece pelo fato do OpenvSwitch ser um *switch* virtual e como está hospedado em uma máquina de propósito geral, possui recursos muito mais poderosos que os *switches* reais utilizados. Assim como o *switch* B, não houve relação entre a quantidade de campos utilizados para *match* e a máxima capacidade de fluxos suportados na tabela de fluxos.

5.1.4 Stanford

. O *switch* referencia de Stanford implementado na NetFPGA também não demonstrou relação entre a quantidade de campos utilizados para *match* e a capacidade da tabela de fluxos. No total foram instalados 32792 fluxos na tabela de fluxos. Embora esta implementação possua um hardware especifico para o encaminhamento de dados e otimização da tabela de fluxos, apenas uma pequena parte da tabela de fluxos é implementada em hardware, cerca de 24 fluxos. Sua grande capacidade de armazenamento deve-se a implementação em software da tabela de fluxos que pode armazenar mais de 32K fluxos.

5.1.5 Comparação da Tabela de Fluxos

Ao final desta análise é notável que os *switches* virtuais possuem um maior capacidade de armazenamento de fluxos, devido a quantidade de recursos disponíveis. O *switch* Stanford embora possua uma quantidade de armazenamento de fluxos mais elevada que dos *switches* reais, não possui uma capacidade de encaminhamento de pacotes tão robusta. Isso deve-se ao fato deste *switch* possuir apenas quatro interfaces para o encaminhamento de dados. A Tabela 4 apresenta a quantidade máxima de fluxos instalados utilizando o teste Tipo 1.

Switch	Capacidade Máxima da Tabela de Fluxos
Switch A	1911
Switch B	2560
Stanford	32792
OpenvSwitch	1000000

Tabela 4: Quantidade máxima de fluxos instalados em cada *switch*

5.2 Utilização da Memória

A análise referente a utilização da memória RAM encontra-se, nesta seção. A análise foi feita de maneira similar a Seção 5.1. Para cada tipo de *match* foi realizada a medição da quantidade de memória utilizada conforme os fluxos fossem instalados. Os resultados serão apresentados em formado de tabela, de maneira que seja possível verificar a quantidade de fluxos instalados e a quantidade de memória RAM utilizada para a instalação de cada Tipo de *Match*.

A tabela 5 apresentam os resultados obtidos com os testes realizados com o *Switch* A. Embora este *switch* possua uma grande quantidade de memoria disponível para um dispositivo embarcado, nota-se que uma quantidade pequena de memória é utilizada para instalação de fluxos.

Tipo de Match	Quantidade de Fluxos	Memória RAM (KB)
1	1911	2356
2	1813	2356
3	1714	2480
4	1430	2108
5	1786	2620
6	1718	3264

Tabela 5: Quantidade de memória utilizada para instalação de fluxos *(*Switch A)

Os resultados de análise da memória RAM referentes ao *Switch* B são apresentados na

Tabela 6. Para este *swtich* é possível notar que a quantidade de memória utilizada para instalação dos 2560 fluxos diminui conforme há quantidade mais elevada de campos utilizados para *match*.

Tipo de Match	Memória RAM (KB)
1	4136
2	4096
3	4100
4	4008
5	3988
6	2856

Tabela 6: Quantidade de memória utilizada pra instalação de fluxos *Switch B*.

Finalmente na tabela 7 apresenta o resultado para o *OpenvSwtch* e Stanford. Em ambos os *switches* a quantidade de memória utilizada mostrou-se a mesma independente da quantidade de campos utilizados para *match*.

Tipo de Match	OpenvSwitch Memória RAM (MB)	Stanford Memória RAM (MB)
1	500	16
2	500	16
3	500	16
4	500	16
5	500	16
6	500	16

Tabela 7: Quantidade de memória utilizada pra instalação de fluxos *Switches* Virtuais

5.2.1 Comparação da Utilização da Memória

Ao final desta análise é notável que os *switches* virtuais utilizam mais memória para a instalação da quantidade máxima de fluxos suportados em suas tabelas. Isso ocorre, pois, os *switches* reais utilizam implementações em hardware da tabela de fluxos devido a pequena quantidade de memória que um dispositivo embarcado possui. Além disso, implementar a tabela de fluxos em hardware otimiza o processo de *match* de um fluxo. Na Tabela 8 é possível visualizar um comparativo entre os tipos de *matches* utilizados e a quantidade de memória utilizada em cada *switch*.

5.3 Utilização da CPU

Para realização deste teste foi enviado para cada *switch* a quantidade máxima de fluxos que podem ser instalados em suas tabela de fluxos. Como resultado, foi obtido a sobrecarga que

Tipo de Match	Switch A	Switch B	OpenvSwitch Memória RAM	Stanford Memória RAM
1	2356KB	4136KB	500MB	16MB
2	2355KB	4096KB	500MB	16MB
3	2480KB	4100KB	500MB	16MB
4	2108KB	4008KB	500MB	16MB
5	2620KB	3988KB	500MB	16MB
6	3264KB	2856KB	500MB	16MB

Tabela 8: Tabela comparativa entre utilização da Memória dos *switches*

a instalação seguida de vários fluxos possui em cada *switch*. A Tabela 9 apresenta os resultados obtidos. É possível notar que a instalação de seguida de várias fluxos nos *switches* reais não apresentam um sobrecarga significativa na CPU. Porém, nos *switches* virtuais a a utilização da CPU apresenta um acréscimo significativo. Isso ocorrê, pois os *switches* virtuais não possuem processadores otimizados para este tipo de funcionalidade. Além de que, os *switches* reais possuem um hardware específico para o processamento de mensagens Openflow.

	Switch A	Switch B	OpenvSwitch Memória RAM	Stanford Memória RAM
Sobrecarga de Utilização da CPU	2%	3%	27%	14%

Tabela 9: Sobrecarga da utilização da CPU no momento da instalação de fluxos

6 CONSIDERAÇÕES FINAIS E TRABALHOS FUTUROS

Uma implementação do protocolo *OpenFlow* apresenta características que tornam a metodologia de testes da RFC 2544 (BRADNER; MCQUAID, 1999) insuficientes para a precisa avaliação de desempenho para equipamentos baseados em SDN. O protocolo *OpenFlow* já possui uma especificação para testes de conformidade, entretanto, ainda são poucos os estudos sobre avaliação de desempenho deste protocolo. Testes de conformidade verificam se o DUT está de acordo com a especificação, porém, esse tipo de avaliação não agrega informações sobre o comportamento do DUT em uma rede real. Além disso, a avaliação de desempenho agrega benefícios, tanto para a indústria, onde é possível identificar pontos de falha e limitações do equipamento, quanto para a academia, onde é possível utilizá-los para comparação entre diferentes estudos de implementações.

Nesse trabalho de conclusão de curso foi proposto uma metodologia de testes de desempenho para avaliar o hardware de *switches* Openflow. Para a implementação dessa metodologia, foram estudadas as métricas que que influenciam no desempenho destes dispositivos. Além disso, foi analisado como as atuais metodologias de avaliação de desempenho são implementadas. Após a implementação, a metodologia de testes foi aplicada em quatro *switches* Openflow, sendo dois *switches* reais e dois *switches* virtuais. Os resultados obtidos foram apresentados comparando os *switches* utilizados, onde, pode-se verificar qual *switche* possui o melhor desempenho para uma determinada métrica.

Este trabalho, abre a possibilidade para a realização de projetos como otimizações na metodologia desenvolvida, aumentar a quantidade de tipos de testes, criação de uma base de dados com os resultados obtidos por cada *switch*. Dessa forma, pode-se aumentar o escopo da metodologia de avaliação, possibilitando a fácil comparação dos resultado entre diferentes *switches*.

REFERÊNCIAS

APACHE. *Apache License*. Disponível em: <http://www.apache.org/licenses/LICENSE-2.0>. Acesso em: Jun. 2015.

BENSON, T.; AKELLA, A.; MALTZ, D. A. Network Traffic Characteristics of Data Centers in the Wild. *Proceedings of the 10th ACM SIGCOMM Conference on Internet Measurement*, New York, NY, USA, p.267–280, 2010.

BRADNER, S.; MCQUAID, J. *Benchmarking Methodology for Network Interconnect Devices*. Disponível em: <http://www.ietf.org/rfc/rfc2544.txt>. Acesso em: Set. 2014, RFC 2544 (Informational).

JARSCHEL, M.; LEHRIEDER, F.; MAGYARI, Z.; PRIES, R. A Flexible OpenFlow-Controller Benchmark. *Proceedings of the 2012 European Workshop on Software Defined Networking*, Washington, DC, USA, p.48–53, 2012.

KANDULA, S.; SENGUPTA, S.; GREENBERG, A.; PATEL, P.; CHAIKEN, R. The Nature of Data Center Traffic: measurements & analysis. *Proceedings of the 9th ACM SIGCOMM Conference on Internet Measurement Conference*, New York, NY, USA, p.202–208, 2009.

KREUTZ, D.; RAMOS, F. M. V.; VERÍSSIMO, P.; ROTHENBERG, C. E.; AZODOLMOLKY, S.; UHLIG, S. Software-Defined Networking: a comprehensive survey. *Computing Research Repository*, v.abs/1406.0440, 2014.

MCKEOWN, N.; ANDERSON, T.; BALAKRISHNAN, H.; PARULKAR, G.; PETERSON, L.; REXFORD, J.; SHENKER, S.; TURNER, J. OpenFlow: enabling innovation in campus networks. *SIGCOMM Comput. Commun. Rev.*, New York, NY, USA, p.69–74, 2008.

MCKEOWN, N.; ANDERSON, T.; BALAKRISHNAN, H.; PARULKAR, G.; PETERSON, L.; REXFORD, J.; SHENKER, S.; TURNER, J. OpenFlow: enabling innovation in campus networks. *SIGCOMM Comput. Commun. Rev.*, New York, NY, USA, v.38, n.2, p.69–74, Mar. 2008.

NAOUS, J.; ERICKSON, D.; COVINGTON, G. A.; APPENZELLER, G.; MCKEOWN, N. Implementing an OpenFlow Switch on the NetFPGA Platform. , New York, NY, USA, p.1–9, 2008.

NETWORK, Big Switch. *Floodlight*. Disponível em: <https://projectfloodlight.org/floodlight/>. Acesso em: Set. 2014.

NETWORK, Big Switch. *Indigo*. Disponível em: <http://www.projectfloodlight.org/indigo/>. Acesso em: Jun. 2015.

ONF. *SDN Architecture*. Disponível em: <https://www.opennetworking.org/>. Acesso em: Mar. 2014.

OPENVSWITCH. *OpenvSwitch*. Disponível em: <http://openvswitch.org/>. Acesso em: Jun. 2015.

POSTEL, J.; REYNOLDS, J. *Telnet Protocol Specification*. Disponível em: <http://www.ietf.org/rfc/rfc854.txt>. Acesso em: Jun. 2015, RFC 854 (Informational).

ROTSOS, C.; SARRAR, N.; UHLIG, S.; SHERWOOD, R.; MOORE, A. W. OFLOPS: an open framework for openflow switch evaluation. *Proceedings of the 13th International Conference on Passive and Active Measurement*, Berlin, Heidelberg, p.85–95, 2012.

TOOTOONCHIAN, A.; GORBUNOV, S.; GANJALI, Y.; CASADO, M.; SHERWOOD, R. On Controller Performance in Software-defined Networks. In: 2012, Berkeley, CA, USA. *Anais...*. p.10–10.