Avaliação de Desempenho de Plataformas para Validação de Redes Definidas por Software

Alextian B. Liberato[1 2], **Diego Mafioletti**[1], **Eros Spalla**[1],
Magnos Martinello[1], **Rodolfo Villaça**[1]

[1] Programa de Pós-Graduação em Informática (PPGI)
Universidade Federal do Espírito Santo (UFES) – Vitória, ES

[2]Grupo de Pesquisa em Informática Aplicada – Departamento de Informática
Instituto Federal do Espírito Santo (IFES) – Colatina, ES

alextian@ifes.edu.br, diego@saorc.com.br, spalla@ifes.edu.br
magnos@inf.ufes.br, rodolfo.villaca@ufes.br

Abstract. *Testbed on designs with in Software Defined Networks are traditionally performed in virtualized and emulated environments, however the results do not faithfully represent the real environment. Manufacturers have provided closed platforms in which the designer of networks becomes dependent on this implementation. This paper provides a comparative analysis of performance between three open platforms shipped in shelf equipment and native closed platform. The results show similarity in the throughput and CPU consumption, however the open platform presented in this paper significantly outperforms in programmability.*

Resumo. *Experimentos em projetos com Redes Definidas por Software tradicionalmente são realizados em ambientes virtualizados e emulados, contudo os resultados obtidos não representam com fidelidade o ambiente real. Os fabricantes têm disponibilizado plataformas fechadas nas quais o projetista de redes torna-se dependente dessa implementação. Este artigo fornece uma análise comparativa de desempenho entre três plataformas abertas embarcadas em equipamentos de prateleira e a plataforma fechada nativa. Os resultados mostram similaridade na vazão e no consumo de CPU, todavia a plataforma aberta apresentada neste artigo supera significativamente em programabilidade.*

1. Introdução

As Redes Definidas por Software (SDN - *Software-Defined Networking*) surgiram como um promissor conceito para o projeto de novas arquiteturas para a Internet. SDN é uma proposta baseada na separação dos planos de dados e de controle em uma interface uniforme, independente de fornecedor, para o mecanismo de encaminhamento (ex.: OpenFlow [McKeown et al. 2008]) e com um plano de controle logicamente centralizado.

O OpenFlow (OF) é um padrão aberto que permite a programação dos elementos ativos da rede, tais como roteadores, *switches* ou pontos de acesso sem fio. Trata-se de uma proposta pragmática com grande potencial para suportar o grau de programabilidade que as SDN necessitam [Kotronis et al. 2012].

A construção e a validação de protótipos em SDN têm acontecido essencialmente por meio de ferramentas de emulação, como o Mininet [Lantz et al. 2010]. Essas ferramentas viabilizam uma rápida prototipação utilizando apenas um computador através de primitivas de virtualização do sistema operacional, tais como processos e *namespaces* de rede. Com essas primitivas, as ferramentas de emulação permitem, rapidamente, criar, interagir e customizar protótipos de redes utilizando *switches* OF, independentemente do *hardware* a ser utilizado. Contudo, o ambiente emulado não garante um alto grau de fidelidade, quando comparado a um ambiente real. Além disso, pequenas alterações nestes ambientes afetam drasticamente a latência e a taxa de transferência nas redes, tornando o uso do ambiente emulado um grande limitador para os projetistas em SDN validarem as aplicações e terem garantias de como elas irão se comportar com *switches* OF [Huang et al. 2013].

Em relação aos equipamentos habilitados com OpenFlow, muitos fabricantes dão suporte ao protocolo OF, entretanto o disponibilizam em um sistema fechado, no qual o projetista de redes torna-se dependente da implementação do fabricante (e.g. Mikrotik RouterOS). Se a inovação do projetista requer modificação no plano de dados, por exemplo quando uma operação de encaminhamento não precisa de *lookup* na tabela de fluxo, o equipamento não permite tal operação, e impede sua validação experimental. Esses fatos limitam significativamente o potencial de inovação na arquitetura SDN, tanto pela restrição imposta pelo modelo de encaminhamento, quanto pelo sistema fechado do fabricante.

A ideia de criar plataformas abertas para experimentação em SDN/OpenFlow tem sido um grande motivador para permitir aos projetistas de redes inovar livremente e, consequentemente, testar com confiança seus projetos em *hardware* de prateleira não especializado. É preciso que essa plataforma seja programável pelo projetista, tenha um desempenho aceitável e custo relativamente baixo.

Este trabalho contribui com o estado da arte nas seguintes direções:

- Avaliando o desempenho de uma plataforma aberta para prototipação de redes definidas por software. Nossa proposta é projetada pela composição de i) equipamentos de prateleira RouterBoard (modelos RB2011UAS-IN, RB433 e RB450G), ii) um sistema operacional aberto OpenWRT, e iii) um motor de encaminhamento programável Open vSwitch (OvS).
- Comprovando imprecisões nos experimentos validados em ambiente emulado em relação i) a validação de experimentos em equipamentos reais; ii) ao próprio ambiente de emulação utilizado.

Nosso objetivo principal neste artigo é apoiar decisões quanto ao uso destas plataformas, revelando uma opção que permita aos operadores de redes executarem seus experimentos em um cenário com equipamentos reais de forma totalmente flexível, e com desempenho similar aos produtos encontrados no mercado.

O trabalho está estruturado da seguinte forma: a Seção 2 expõe um breve referencial teórico das ferramentas utilizadas; a Seção 3 apresenta as orientações para criação da plataforma aberta que utiliza *hardware* não especializado para redes SDN, além da estrutura e procedimentos metodológicos utilizados na realização dos experimentos; a Seção 4 mostra os resultados obtidos; a Seção 5 tece algumas considerações finais e trabalhos futuros.

2. Referencial Teórico

Lantz [Lantz et al. 2010] explica que em uma rede definida por *software* o plano de controle (ou "sistema operacional de rede") é separado do plano de dados. Normalmente, o sistema operacional de rede observa e controla o estado de toda a rede a partir de um ponto centralizado, oferecendo recursos como: protocolos de roteamento, controle de acesso, virtualização de rede, gestão de energia e também prototipação de novos de protocolos. A principal consequência das SDN é que as funcionalidades da rede podem ser alteradas ou mesmo definidas após a rede ter sido implantada. Novas funcionalidades podem ser adicionadas, sem a necessidade de se modificar o *hardware*, permitindo que o comportamento da rede evolua na mesma velocidade que o *software*.

Casado [Casado et al. 2012] apresenta características desejáveis para SDN, separando-as em características de *hardware* e de *software*. O *hardware* deve ser simples, barato, fácil de operar e independente de fabricante, evitando que os usuários sejam obrigados a usar equipamentos de um único fornecedor. Também deve suportar inovações futuras, evitando atualizações desnecessárias. Com relação ao *software*, Casado entende que ele deve ser flexível, estruturado e capaz de suportar uma ampla variedade de requisitos (isolação, virtualização, engenharia de tráfego, controle de acesso, etc). Também deve ser modular e expansível, permitindo inclusão e alteração de módulos.

O OpenFlow é um padrão aberto para redes definidas por *software* cujo principal objetivo é permitir que sejam utilizados equipamentos de rede comerciais para pesquisa e experimentação de novos protocolos, em paralelo com a operação normal das redes.

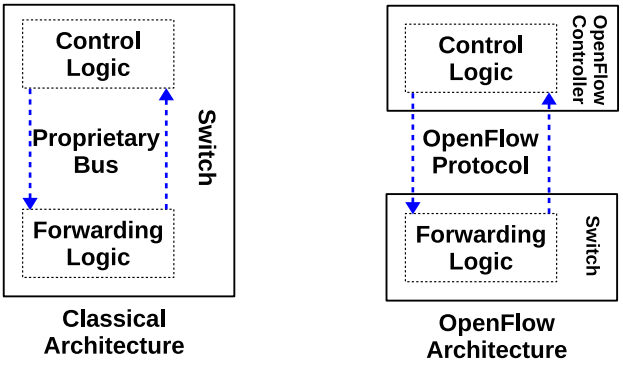


Figura 1. *Hardware* tradicional de rede *versus hardware* OpenFlow Fonte: [Sherwood et al. 2009]

Sherwood [Sherwood et al. 2009] apresentou uma comparação entre a arquitetura tradicional *versus* arquitetura que utiliza o protocolo OpenFlow (Figura 1). Percebe-se que à esquerda, na arquitetura clássica, tanto a lógica de controle quanto a lógica de encaminhamento estão localizadas internamente no *switch*, comunicando-se através de um barramento proprietário do fabricante. Na arquitetura OpenFlow, o *switch* executa apenas a lógica de encaminhamento. A lógica de controle é executada no controlador, e a comunicação entre elas se dá por meio do protocolo OF.

A utilização de um ambiente emulado para prototipação em SDN do protocolo OpenFlow é uma prática atualmente dominante [Conterato et al. 2013]. No entanto, al-

gumas situações exigem que essa abordagem seja repensada, pois ela pode deturpar o resultado final quando portada para equipamentos reais, devido a fatores como limitações de memória e CPU em equipamentos embarcados [Huang et al. 2013].

Em [Handigol et al. 2012] foi avaliada uma abordagem denominada *Container-Based Emulation* (CBE), onde um ambiente de *hosts* virtuais, *switches* e *links* rodam em um servidor. O CBE combina muitas das melhores características dos emuladores de *software* e bancos de ensaio em *hardware*, mas sua fidelidade no desempenho e na vazão não foram provados.

2.1. Projeto da Plataforma Aberta Programável

Nossa proposta de plataforma aberta programável não restringe-se ao ambiente emulado. A ideia é explorar a capacidade de alguns equipamentos existentes no mercado atual, que poderão ser usados para prototipação, com desempenho compatível e a um custo acessível. Com base em resultados validados em experimentos reais, abre-se novas perspectivas que não seriam possíveis por meio do ambiente emulado tradicional.

Utilizamos equipamentos da Mikrotik, uma empresa da Letônia fundada em 1995, responsável pelo desenvolvimento do sistema operacional RouterOS, e posteriormente pela construção do *hardware* embarcado Mikrotik RouterBoard. Combinados, eles são destinados ao mercado de pequenos a médios provedores de serviço Internet.

A partir de sua versão 6.0, o RouterOS possibilita a utilização do protocolo OpenFlow versão 1.0, através da instalação de seu respectivo pacote. Por ser um *software* proprietário de plataforma fechada, o RouterOS não permite modificações e atualizações além do OpenFlow 1.0 padrão, disponibilizado pelo fabricante.

Em nossos experimentos utilizamos o OpenWRT, que é uma distribuição GNU/Linux altamente extensível para dispositivos embarcados, dentre eles, o Routerboard utilizado neste trabalho. Diferentemente dos demais *softwares* originais desses equipamentos, o OpenWRT foi construído a partir do zero para ser um sistema operacional completo e facilmente modificável, utilizando um *kernel* do Linux mais recente que a maioria das outras distribuições [OpenWRT 2014a].

Também utilizamos dois *switches* virtuais. O primeiro foi o Softswitch13 CPqD[1], que se trata de um *switch* em modo usuário, compatível com as especificações contidas no protocolo OpenFlow versão 1.3. O segundo, o Open vSwitch (OvS) que é um *switch* virtual que segue a arquitetura OpenFlow, também é implementado em *software*, mas com o plano de dados implementado diretamente no *kernel* do Linux, enquanto funções de controle são implementadas em modo de usuário [Pfaff et al. 2009, Subramanian et al. 2009].

Outros estudos avaliaram o desempenho do protocolo OpenFlow em comparação com abordagens convencionais de encaminhamento e roteamento [Mateo 2009]. Além desses, há estudos que detalham aspectos funcionais de maneira comparativa entre implementação OpenFlow em *software*, no caso OvS, e implementação em NetFPGA, que é uma plataforma aberta em *hardware*, e implementações com *switches* comerciais, basicamente utilizando a versão 1.0 do protocolo OF [Appelman et al. 2012].

Em [de Oliveira et al. 2012] o foco do trabalho estava em analisar o tempo de

[1]Projeto disponível em: https://github.com/CPqD/openflow-openwrt

resposta do encaminhamento no plano de dados de diferentes implementações de comutadores OpenFlow disponíveis, verificando o comportamento de operações básicas, como por exemplo troca de VLANs, fornecendo assim parâmetros balizadores para experimentos com comutadores OpenFlow através de resultados comparativos entre o OvS e implementações em NetFPGA com alta precisão na escala de microssegundo.

No conhecimento dos autores deste trabalho, é a primeira proposta de arquitetura aberta que utiliza equipamentos de prateleira de baixo custo, tais como Routerboards, e customizando componentes de *software* abertos, que incluem Open vSwitch, Softswitch (CPqD) e OpenWRT, de forma a permitir o uso do protocolo OF nas versões 1.X (0, 1, 2 e 3).

3. Ambiente de Experimentação

Dividimos esta seção de forma a organizar o ambiente de experimentação, descrito em três subseções. A primeira tem como objetivo orientar a construção da plataforma aberta que permite executar nossos experimentos; a segunda ilustra o arranjo topológico usado para explorar as características dos equipamentos disponíveis; e a terceira descreve as ferramentas e procedimentos metodológicos utilizadas nos experimentos.

3.1. Criando a Plataforma Aberta em *Hardware* não Especializado

Iniciamos o projeto criando um ambiente para compilação cruzada (*cross-compiling*), uma vez que a arquitetura da RouterBoard baseia-se em MIPS, o que torna imprescindível a geração desse ambiente específico para que seja possível a construção do OpenWRT e seus pacotes. Tal ambiente foi criado em uma distribuição Linux x86, utilizando-se as ferramentas disponíveis na distribuição do código fonte do OpenWRT, como compilador e bibliotecas necessárias.

A próxima etapa foi a substituição do sistema operacional RouterOS, embarcado por padrão nas RouterBoards, pelo sistema aberto OpenWRT. Os passos abaixo descrevem esse processo [OpenWRT 2014b]:

1. Instalação e configuração dos serviços de BOOTP, TFTP e HTTP; deste modo, quando os *switches* inicializarem, poderão se conectar a este servidor, afim de receber a gravação definitiva do novo *firmware* na RouterBoard;
2. acesso dos equipamentos através de comunicação serial padrão RS-232 durante sua inicialização, alterando a sequência de *boot*, para que eles possam inicializar via BOOTP, ao invés da inicialização tradicional via memória *flash*;
3. inicialização da RouterBoard no sistema OpenWRT, assistida via console serial, onde o sistema é carregado por meio da memória RAM do equipamento, utilizando a imagem criada anteriormente durante o processo de compilação cruzada;
4. efetuação da gravação definitiva do OpenWRT na RouterBoard, com uso do aplicativo *wget2nand*, disponível no próprio sistema, executando assim a gravação do *firmware* na memória *flash* do equipamento.

A partir desse momento, o OpenWRT está embarcado na RouterBoard, e sua configuração deve ser definida de acordo com cada modelo de equipamento, visto que cada qual possui suas próprias especificações e características. O acesso ao equipamento e suas configurações pode ser feito via linha de comando, com utilização do console serial, ou conexão *shell* segura (ssh). Algumas configurações são compartilhadas por todos

os modelos, entre elas configurações de VLAN e habilitação do protocolo OpenFlow, cujo procedimento é descrito a seguir:

1. Configuração das interfaces de rede do equipamento para funcionamento independente através de VLANs: cada porta é associada a uma VLAN que se comunica com o *switch chip*, por meio do aplicativo *swconfig* que determina qual VLAN é responsável pela(s) porta(s);
2. ativação do *switch* virtual específico para o experimento, Softswitch13 ou Open vSwitch;
3. definição das interfaces VLANs a serem associadas ao encaminhador e suas configurações, através dos arquivos de configuração apresentados nesta Seção 3.

Após esses passos, os equipamentos estão aptos para funcionamento com OpenFlow e poderão ser utilizados na implementação do protótipo.

3.2. Arranjo Topológico da Plataforma

Com os equipamentos disponíveis para os experimentos, adotamos como prova de conceitos a configuração topológica ilustrada na Figura 2, de forma a explorar como a plataforma suporta a vazão variando os caminhos entre origem e destino, com um caminho mais longo, de menor taxa (Fast Ethernet) e com mais saltos, e outro com maior taxa (Gigabit Ethernet) e menor quantidade de saltos. Com esta variação foi possível analisar se a CPU dos equipamentos poderia ser um fator limitante no desempenho da plataforma aberta utilizada.

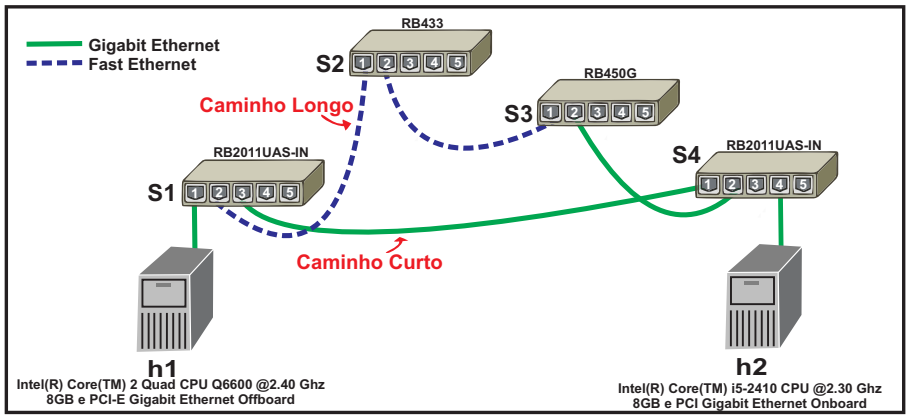


Figura 2. Topologia base utilizada nos experimentos no ambiente real e emulado

Usamos os seguintes equipamentos: quatro RouterBoards, sendo dois RB2011UAS-IN, um RB433 e um RB450G. A esse ambiente, formado pelos 4 equipamentos citados, daremos o nome de "ambiente real". Para o "ambiente emulado", que consiste na execução do Mininet diretamente no SO, utilizamos dois *hosts* físicos: i) denominado "Fedora", com um processador Core i5 2.9GHz, 8GB RAM, HD 500GB 5400rpm, SO Linux Fedora Core 20 64bits; e ii) denominado "Ubuntu", com um processador Core i5 3.2GHz, 6GB RAM, HD 500GB 5400rpm, SO Linux Ubuntu 12.04 64bits. E por fim, o "ambiente emulado virtualizado", que executa o Mininet em uma máquina virtual, sobre um *host* físico com processador Core i5 3.2GHz, 8GB RAM, HD 500GB 5400rpm e SO Windows 7 64bits, VirtualBox 4.02.14 com VM Linux Ubuntu 12.04 32bits, com 1 CPU, 8GB vdisk e 4096MB RAM.

Os equipamentos foram posicionados na topologia da seguinte forma: **S1** e **S4** = RB2011UAS-IN, **S2** = RB433 e **S3** = RB450G. A Tabela 1 apresenta os detalhes do *hardware* dos *switches* RouterBoards.

Observe que o RB450G possui processador com maior *clock* que o RB2011UAS-IN, no entanto sua arquitetura é inferior; por isso o RB2011UAS-IN apresenta maior desempenho.

Tabela 1. Configuração Mikrotik Routerboard

Modelo	Clock	Arq. Processador	Memória	Interfaces
RB2011UAS-IN	600 MHz	MIPS 74Kc V4.12	128 MB	5 portas Fast, 5 portas Gigabit* e 1 porta SFP
RB450G	680 MHz	MIPS 24Kc V7.4	256 MB	5 portas Gigabit*
RB433	300 MHz	MIPS 24Kc V7.4	64 MB	3 portas Fast e 3 slots miniPCI

*Vazão máxima alcançada entre portas Gigabit ligadas ao mesmo *switch chip* chega a 650Mbps [RouterBOARD 2014].

3.3. Ferramentas e Procedimentos Metodológicos

Para garantir igualdade entre os cenários, todos os experimentos foram realizados com *setup* reiniciado, garantindo que informações contidas no cache não influenciem nos resultados. Por justiça na avaliação de desempenho, todos os cenários utilizaram o mesmo número de regras nos *switches* (mínimo para comunicação).

A inclusão das regras que permitem aos *switches* executarem o encaminhamento dos pacotes (regras de fluxo) foi realizada sem a intervenção do controlador, através das ferramentas DPCTL 1.3.0 e OVS-OFCTL 2.1.0, para o Softswitch13 e Open vSwitch, repectivamente. Tendo em vista que nosso objetivo não era avaliar latência na atualização das regras nos *switches*, optamos em mantê-las estáticas durante o período dos experimentos.

Como parâmetros de desempenho, foram avaliados vazão na rede (*throughput*) e percentual de uso do processador durante o período dos testes nos ambientes emulado e real.

Para avaliar a vazão fim-a-fim, utilizamos o Iperf [Goldoni and Schivi 2010] na versão 2.05 com o protocolo UDP e o Iperf3 [Jon Dugan et al. 2014] na versão 3.0.2 com o protocolo TCP. Na realização dos experimentos, o *host* 2 ficou definido como servidor, e o *host* 1 como cliente.

Para medir o percentual de uso do processador (CPU), usamos a ferramenta de monitoramento mpstat [Lange 2013], reportando as estatísticas desejadas (User+Nice+Sys+Irq+Soft) que, se totalizadas, informam o percentual de uso do CPU nos ambientes emulado e real (apenas no OpenWRT). Como o sistema RouterOS não permite a instalação dessa ferramenta, foi necessário o emprego de um comando similar, "*system resource cpu print interval=1*", por meio do qual foi possível coletar informações

menos detalhadas (load + irq, que correspondem a carga total e interrupções do sistema, respectivamente), mas com mesmo teor de confiabilidade.

Coletaram-se 120 amostras para cada cenário, em intervalos de 1 segundo, totalizando assim 96 experimentos com 11.520 registros. Por essa quantidade de amostras, foi possível obter um intervalo de confiança de 95% aceitável para as experimentações.

Para o ambiente emulado e emulado virtualizado utilizou-se o Mininet 2.1.0 com suporte ao OF 1.3. Os arquivos utilizados para emulação dessa topologia no Mininet com as respectivas orientações, incluindo *scripts* e as regras de fluxo, estão disponíveis em https://github.com/alextianliberato/ambienteemulado.git.

Não foram realizados experimentos no plano de controle, uma vez que o foco do artigo é a avaliação do desempenho do plano de dados em diferentes cenários.

As aferições realizadas com o TCP foram coletadas no cliente, já as medidas com o UDP, no servidor, pois o cliente com UDP envia rajadas para o servidor, não reportando os dados efetivamente transferidos. Todos os resultados foram armazenados em arquivos-texto para posterior análise, seguindo as etapas que se apresentam:

1. Setup: etapa responsável pela configuração topológica nos ambientes, incluindo arranjos físicos (ligação dos *patch cords*, *reset* dos equipamento, etc.) e restart dos serviços no ambiente emulado;
2. configuração enlace: etapa que tem como objetivo a inclusão das regras de fluxo nos *switches*;
3. experimentação: nesta etapa são executadas as configuração das ferramentas e armazenado os resultados em arquivos no formato de texto.

4. Resultados Experimentais

Nesta seção discutem-se os resultados obtidos nos experimentos. Para organizar os resultados, optamos por subseções envolvendo o cenário característico e o objetivo de cada experimento, onde as métricas de desempenho são avaliadas.

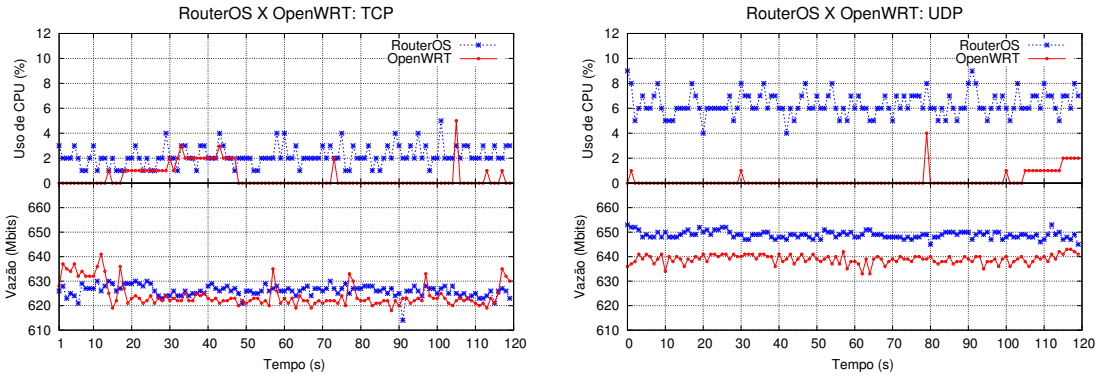
4.1. Plataforma Fechada RouterOS versus Aberta OpenWRT

Usamos nesse experimento o menor caminho com o objetivo de obter o máximo de vazão alcançado pelos equipamentos e o percentual de uso da CPU do *switch* de origem (S1). Não se fez necessário avaliar o percentual de uso da CPU do *switch* de destino (S4), pois não detectamos perdas significativas (desvio padrão de 0,01%) entre o total de datagramas transmitidos versus recebidos nas duas plataformas (fechada e aberta).

Apresentamos na Figura 3, a avaliação de vazão e percentual de uso do CPU entre as plataformas fechada (RouterOS) e aberta (OpenWRT) em modo *switching chip*. Comparamos utilizando o protocolo TCP, ilustrado na Figura 3(a), em que h1 foi definido como cliente e h2 como servidor; a vazão e o percentual de uso do CPU foram capturados em h1 e S1, respectivamente. A Figura 3(b) mostra o protocolo UDP; avaliamos a transferência entre o cliente (h1), transmitindo rajadas de datagramas limitadas em 650Mbps (limite imposto pelo *hardware*) para o servidor (h2), no qual armazenamos em um log os valores totais recebidos.

Observa-se que o RouterOS exigiu mais consumo de CPU e considerando a vazão no protocolo TCP, esse foi levemente superior ao OpenWRT. No caso da vazão com UDP,

o RouterOS se mostrou mais estável e com uma menor variação na vazão, enquanto o OpenWRT teve uma leve variação. No entanto, quando avaliamos o percentual relativo entre as vazões, notamos que o RouterOS foi apenas 0,28% superior à plataforma aberta no TCP, e no UDP a vazão no RouterOS foi superior em 1,53% ao OpenWRT.



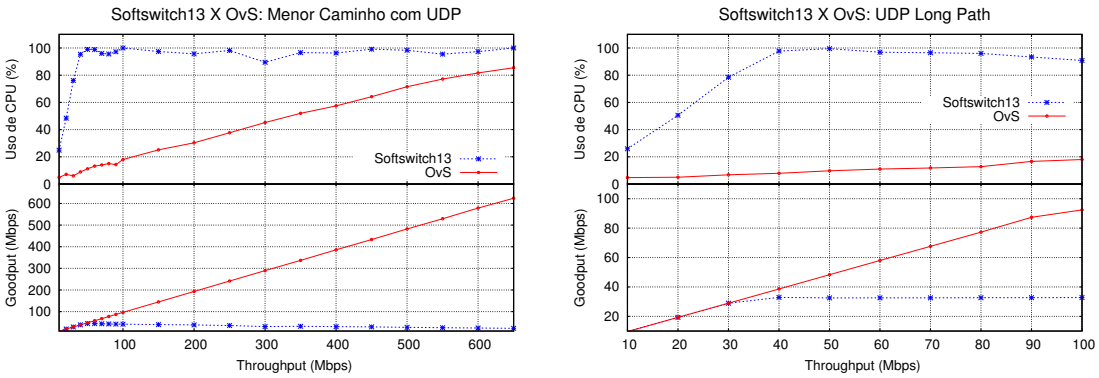
(a) Resultado com Iperf3 e protocolo TCP (b) Resultado com Iperf e protocolo UDP

Figura 3. Vazão e CPU – plataforma fechada x aberta

4.2. Softswitch13 versus Open vSwitch

Nosso objetivo nesta subseção é a comparação do desempenho entre duas máquinas de encaminhamento (*forwarding engine*) considerando a plataforma aberta OpenWRT. Com esse propósito, as duas máquinas de encaminhamento escolhidas foram Softswitch13 e OvS.

Para aferir estas medidas, enviamos rajadas incrementais com duração de 1 segundo, conforme distribuição dos pontos nos gráficos nas Figuras 4(a) e 4(b); Observa-se que conforme aumentamos as rajadas transmitidas (eixo x) a vazão (eixo y) atinge seu limite, diminuindo sua taxa tanto no menor caminho, quanto no maior caminho.



(a) Resultado com Iperf menor caminho (b) Resultado com Iperf maior caminho

Figura 4. Softswitch13 com protocolo UDP – Routerboards

Também observamos os datagramas perdidos com o menor e maior caminho. Conforme aumentamos as rajadas com o protocolo UDP na origem para o destino com

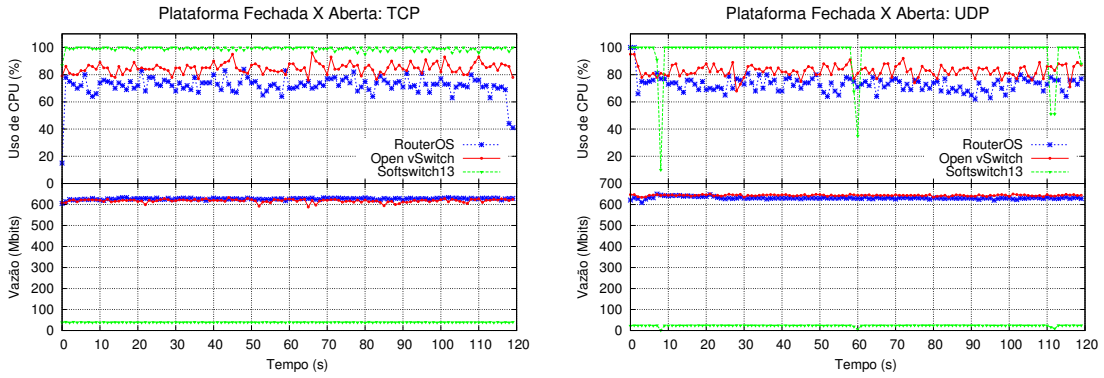
Softswitch13, a quantidade de pacotes perdidos aumenta; já com o OvS a perda de pacotes é menor. Quando comparamos o total de datagramas perdidos, a diferença entre o Softswitch13 e o OvS no menor caminho chega a 206,35%. Todavia no maior caminho, que percorre equipamentos com menor poder de processamento, a diferença entre OvS e o Softswitch13 é considerável, atingindo 211,54%.

Os resultados mostram que o fator limitador de vazão é a CPU dos equipamentos utilizados, por isso o Softswitch13 fica limitado à máximo vazão em 44.79Mbps no UDP e 40.9Mpbs no TCP. Vale notar que o limite assintótico de vazão denotada por X é: $X \leq \min(\frac{1}{D_{max}}, \frac{N}{\sum_1^k D_i})$. Nesse caso, a CPU é o recurso com a maior demanda D_{max}.

4.3. Routerboards com diferentes versões do OpenFlow

O objetivo desse experimento é ter uma referência comparativa partindo de RouterBoard fechada RouterOS com OF 1.0. Como não há ainda suporte ao OF 1.3 no RouterBoard, nosso propósito é implantar o OF 1.3 no OpenWRT e medir a vazão no encaminhamento com OvS e Softswitch13, utilizando o menor caminho com maior taxa de transmissão. Seguimos os mesmos passos metodológicos descritos anteriormente.

Na Figura 5(a), pode-se notar diferenças no consumo de CPU. Conforme esperado, o resultado foi o Softswitch13 consumindo mais recursos, seguido pelo OvS e, por último, o RouterOS. A vazão do RouterOS foi muito similar com diferença de apenas 1,49% em relação ao OvS no protocolo TCP; Em contrapartida, no protocolo UDP, o *goodput* no OvS foi 1,91% superior à plataforma fechada, apresentado na Figura 5(b).



(a) Resultado com Iperf3 e protocolo TCP

(b) Resultado com Iperf e protocolo UDP

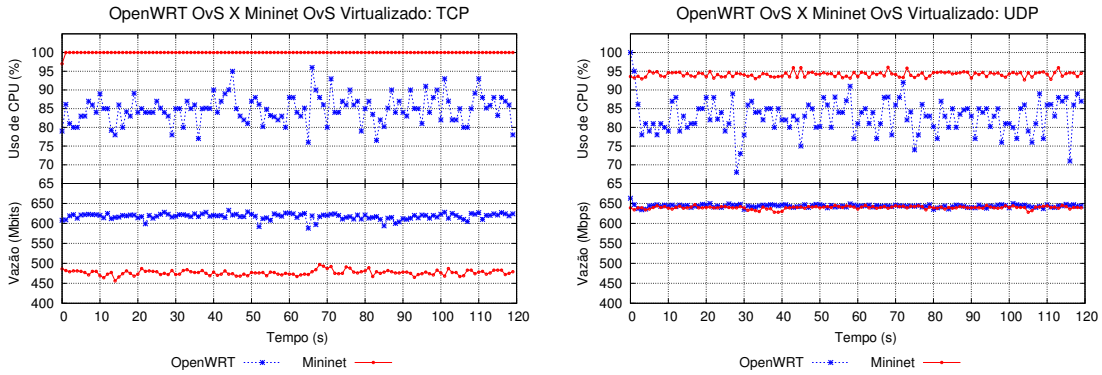
Figura 5. Vazão e CPU – plataforma fechada x aberta

Novamente comprovamos que o desempenho em quesito de vazão na rede entre a plataforma fechada e aberta foi similar. No entanto, é importante destacar que na plataforma fechada não podemos efetuar nenhuma alteração, como por exemplo no encaminhamento dos pacotes; isso limita o desenvolvimento de novos projetos e sua respectiva validação em ambiente real.

4.4. Ambiente Emulado versus Real

O objetivo desse experimento é avaliar o grau de fidelidade na validação de redes em ambiente emulado. Utilizamos os mesmos procedimentos apresentados anteriormente,

que incluem configuração da mesma topologia com as mesmas taxas para os enlaces, e mesmo tráfego gerado a partir de protocolos TCP e UDP. Neste ambiente, avaliamos apenas o Open vSwitch com o protocolo OF 1.3 no menor caminho, uma vez que o Softswitch13 apresentou desempenho limitado, conforme Figuras 5(a) e 5(b).



(a) Resultado com Iperf3 e protocolo TCP

(b) Resultado com Iperf e protocolo UDP

Figura 6. Vazão e CPU – ambiente emulado x real

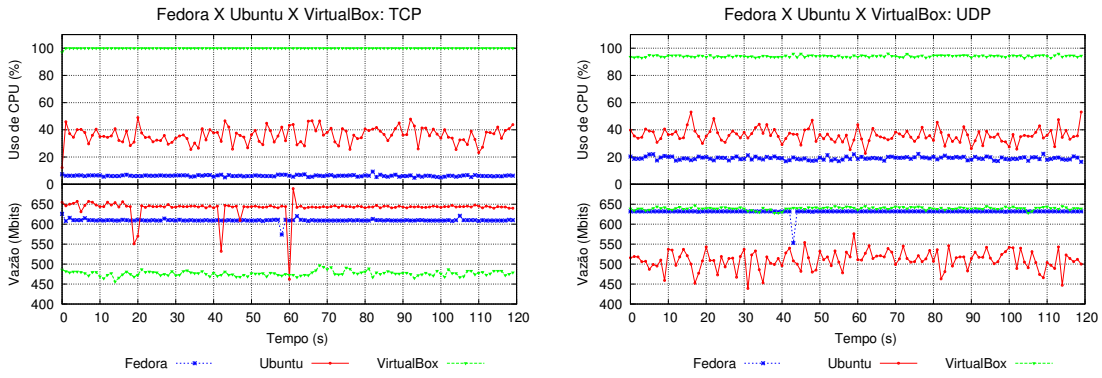
Nota-se que a vazão com o protocolo UDP, ilustrado na Figura 6(b), foi similar nos ambientes emulado virtualizado e real. A diferença principal foi o consumo de CPU, que no ambiente emulado virtualizado se manteve próximo aos 95%, durante todo o experimento. Todavia, quando analisamos a Figura 6(a), é possível comprovar que no ambiente emulado virtualizado a vazão foi 22,81% menor que no ambiente real; essa imprecisão na validação emulada pode levar a uma avaliação incorreta, prejudicando a tomada de decisão pelo projetista de rede a partir do ambiente emulado virtualizado.

Com o maior caminho, os resultados com o protocolo TCP foram similares ao anterior, tanto na vazão quanto no consumo de CPU. Porém o *goodput* com o protocolo UDP no ambiente real foi 84,80% maior que no ambiente emulado, ou seja, novamente o ambiente emulado virtualizado poderia conduzir o projetista a erros na tomada de decisão.

4.5. Comparação entre os Ambientes Emulados

Nosso objetivo nesse experimento é mostrar que com as mesmas configurações de rede no ambiente emulado, os resultados (vazão e consumo de CPU) podem variar substancialmente. Com esse propósito, avaliamos diferentes ambientes de emulação usando Mininet 2.1.0, considerando: i) sistemas operacionais sem virtualização Ubuntu e Fedora, ii) ambiente virtualizado “VirtualBox”.

Na Figura 7(a) ilustramos os resultados das amostras com o protocolo TCP, e na Figura 7(b) apresentamos os resultados com o protocolo UDP, utilizando o menor caminho, todos com a mesma configuração topológica.



(a) Resultado com Iperf3 e protocolo TCP

(b) Resultado com Iperf e protocolo UDP

Figura 7. Vazão e CPU – ambiente emulado x ambiente emulado virtualizado

Observamos que os ambientes emulados Fedora e VirtualBox tiveram pouca variação no consumo do CPU. Por outro lado, no Ubuntu esta variação é maior, entre 34,86% e 36,98% com o TCP e entre 35,08% e 36,89% com o UDP, considerando um intervalo de confiança de 95%. Em relação a vazão, nota-se também maior variação no Ubuntu com o UDP, entre 505,94 e 514,37 Mbps, com o mesmo intervalo de confiança.

Apesar da recomendação do Mininet ser de usar o sistema operacional Ubuntu [Mininet 2014], observamos que a vazão, em particular, no caso do UDP, mostrou-se menor que no Fedora. Todavia, com o TCP, o Ubuntu teve maior vazão que os demais ambientes consumindo um pouco mais de CPU do que no Fedora (Figura 7(a)). Esses resultados comprovam consideráveis imprecisões que podem acontecer em ambientes emulados. Essas imprecisões podem confundir o projetista na validação da sua proposta.

5. Conclusão

Este trabalho avaliou uma plataforma aberta para prototipação de redes definidas por *software*. A proposta inova ao explorar alguns equipamentos disponíveis no mercado, apoiando-se em um sistema operacional aberto OpenWRT composto por um motor de encaminhamento programável Open vSwitch (OvS). A plataforma permite aos projetistas de redes gerar, analisar e prototipar em equipamentos não especializados, de forma flexível e com desempenho compatível com arquiteturas proprietárias.

Os resultados mostram que de maneira geral, o desempenho da plataforma aberta foi similar à plataforma fechada Router OS e as vezes se mostrou superior. Nossa proposta, entretanto, supera significativamente em flexibilidade se comparada às opções de equipamentos com soluções proprietárias. A plataforma fechada RouterOS consome mais CPU, quando comparada a plataforma aberta OpenWRT com tráfego TCP, todavia a vazão foi apenas 0,28% superior. Na comparação com tráfego UDP, a vazão com o OpenWRT foi levemente menor e o consumo de CPU inferior em todas as amostras. Quanto as diferentes versões do protocolo OpenFlow nas plataformas, a vazão no RouterOS com OF 1.0 foi maior em 1,49% em relação ao OvS com OF 1.3 para tráfego TCP. No entanto, o resultado se inverte quando o tráfego gerado foi UDP, a vazão no OvS foi 1,91% superior do que no RouterOS.

Quanto à validação de experimentos no *software* de emulação (Mininet), houveram diferenças significativas com relação aos ambientes reais. Os resultados comprovam limitações no grau de fidelidade de ambientes emulados e variações importantes nos resultados entre ambientes virtualizados. Por exemplo, a vazão foi 22,81% menor no Mininet do que no ambiente real. Em um caminho com mais saltos, a vazão no ambiente real foi 84,80% superior ao ambiente emulado.

Como trabalhos futuros, pretendemos utilizar os recursos de tabela de grupos do OF 1.3, como o *select* e *fast failover*, para balanceamento e resiliência, respectivamente, ambos de forma pró-ativa, ou seja, sem a dependência do controlador. Também pretendemos explorar alterações no OvS com abordagem centrada em caminhos embarcados na origem, fornecendo escalabilidade e flexibilidade diretamente no plano de dados.

Agradecimentos: aos parceiros Christian Esteve Rothenberg (UNICAMP) e Eder Leão Fernandes (CPqD) pelas orientações com o Softswitch13 e ao professor Ali Mohammad Salahuddin (*Western Michigan University*) no OvS.

Referências

Appelman, M., Boer, M. D., and Pol, R. V. D. (2012). Performance analysis of openflow hardware. Website. `http://staff.science.uva.nl/~delaat/rp/2011-2012/p18/report.pdf`.

Casado, M., Koponen, T., Shenker, S., and Tootoonchian, A. (2012). Fabric: A retrospective on evolving sdn. In *HotSDN*.

Conterato, M., Oliveira, I., Ferreto, T. C., and de Rose, C. A. F. (2013). Avaliação do suporte à simulação de redes openflow no ns-3. *Anais da 11a. Escola Regional de Redes de Computadores*.

de Oliveira, R. E. Z., Filho, P. P. P., Ribeiro, M. R. N., and Martinello, M. (2012). Parâmetros balizadores para experimentos com comutadores openflow: avaliação experimental baseada em medições de alta precisão. *XXX Simpósio Brasileiro de Telecomunicações*.

Goldoni, E. and Schivi, M. (2010). End-to-end available bandwidth estimation tools, an experimental comparison. In *Proceedings of the Second International Conference on Traffic Monitoring and Analysis*, TMA'10, pages 171–182, Berlin, Heidelberg. Springer-Verlag.

Handigol, N., Heller, B., Jeyakumar, V., Lantz, B., and McKeown, N. (2012). Reproducible network experiments using container-based emulation. In *Proceedings of the 8th International Conference on Emerging Networking Experiments and Technologies*, CoNEXT '12, pages 253–264, New York, NY, USA. ACM.

Huang, D. Y., Yocum, K., and Snoeren, A. C. (2013). High-fidelity switch models for software-defined network emulation. In *Proceedings of the Second ACM SIGCOMM Workshop on Hot Topics in Software Defined Networking*, HotSDN '13, pages 43–48, New York, NY, USA. ACM.

Jon Dugan, E., Elliott, S., Prabhu, K., and Jef Poskanzer, E. (2014). Iperf3: A tcp, udp, and sctp network bandwidth measurement tool. Website. `https://github.com/esnet/iperf`.

Kotronis, V., Dimitropoulos, X., and Ager, B. (2012). Outsourcing the routing control logic: better internet routing based on sdn principles. In *Proceedings of the 11th ACM Workshop on Hot Topics in Networks*, pages 55–60. ACM.

Lange, T. A. P. (2013). Avaliação dos impactos de um novo paradigma de virtualização de banco de dados. In *Dissertação de Mestrado*, Fac. de Informática, PUCRS, Porto Alegre.

Lantz, B., Heller, B., and McKeown, N. (2010). A network in a laptop: rapid prototyping for software-defined networks. In *Proceedings of the Ninth ACM SIGCOMM Workshop on Hot Topics in Networks*, page 19. ACM.

Mateo, M. P. (2009). Openflow switching performance. In *Tesi di Laurea Magistrale*, III Facoltà di Ingegneria dell'Informazione – Corso di Laurea in Telecommunication Engineering.

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

Mininet (2014). An instant virtual network on your laptop (or other pc). Website. `http://mininet.org/`.

OpenWRT (2014a). Openwrt. Website. `"http://www.openwrt.org"`.

OpenWRT (2014b). Openwrt wiki: Mikrotik routerboard 450g (rb-450g). Website. `"http://wiki.openwrt.org/toh/mikrotik/rb450g"`.

Pfaff, B., Pettit, J., Amidon, K., Casado, M., Koponen, T., and Shenker, S. (2009). Extending networking into the virtualization layer. In [Subramanian et al. 2009].

RouterBOARD (2014). Product catalog q3-q4 2009. Website. `http://www.mikrotik.com/pdf/what_is_routerboard.pdf`.

Sherwood, R., Gibb, G., Yap, K., Appenzeller, G., Casado, M., McKeown, N., and Parulkar, G. (2009). Flowvisor: A network virtualization layer. *OpenFlow Switch Consortium, Tech. Rep*.

Subramanian, L., Leland, W. E., and Mahajan, R., editors (2009). *Eight ACM Workshop on Hot Topics in Networks (HotNets-VIII), HOTNETS '09, New York City, NY, USA, October 22-23, 2009*. ACM SIGCOMM.