Reprodução de Experimentos com Reescritor de IP Click em rede SDN Emulada

José Wilson Vieira Flauzino[1]

[1] Departamento de Informática – Universidade Federal do Paraná (UFPR)

jwvflauzino@inf.ufpr.br

Abstract. *Recent paradigms such as SDN and NFV bring new concepts and possibilities to build and manage computer networks. In this way, solutions that might not be viable before, will be explored. The paper 'Rapid IP rerouting with SDN and NFV' proposes a transparent user redirection solution for ideal CDN nodes in cases of network failures or congestion. This report aims to reproduce some experiments conducted in the paper, using Mininet as SDN emulator, POX as OpenFlow controller and Click Modular Router to elaborate NFV solution.*

Resumo. *Recentes paradigmas como SDN e NFV trazem novos conceitos e possibilidades para construir e gerenciar redes de computadores. Com isso, soluções que antes poderiam não ser viáveis, passam a ser exploradas. O artigo 'Rapid IP rerouting with SDN and NFV' propõe uma solução de redirecionamento transparente de usuários para nós CDNs ideais em casos de falhas ou congestionamento de rede. O presente relatório tem como objetivo reproduzir alguns experimentos conduzidos no artigo, utilizando o Mininet como emulador SDN, POX como o controlador OpenFlow e o Click Modular Router para elaborar a solução NFV.*

1. Introdução

As redes de computadores tradicionais apresentam uma infraestrutura rígida em relação a inovação, tendo em vista que os dispositivos geralmente possuem plataformas fechadas e proprietárias. O conceito de Redes Definidas por *Software* (SDN) propõe a separação entre o plano de dados e o plano de controle, tornando os dispositivos de rede simples comutadores de pacotes e passando o controle da rede para uma entidade logicamente centralizada e programável.

Outro importante paradigma é a Virtualização de Funções de Redes (NFV), capaz de permitir que funções de rede comumente executadas por *hardwares* específicos passem a ser realizadas em *software* e dinamicamente implantadas, podendo ser executadas em qualquer servidor comum.

Desse modo, tanto SDN quanto NFV trazem novos conceitos, bem como, novas possibilidades e abordagens para projetar, implantar e gerenciar redes computacionais. Explorando esses conceitos, no artigo *'Rapid IP rerouting with SDN and NFV'* os autores Jeffrey Lai, Qiang Fu e Tim Moors apresentam um novo método de redirecionamento transparente de usuários para nós CDNs ideais em determinadas circunstâncias.

Para compreender o problema tratado no artigo é interessente lembrar que, no cenário atual, quando um usuário solicita acesso a um *site* hospedado em um *host* presente em uma Rede de Distribuição de Conteúdo (CDN), os nomes de *hosts* são resolvidos usando técnicas de DNS que retornam ao usuário o endereço de um nó que melhor o

atenderá no momento, no entanto, futuras alterações das condições da rede e/ou do servidor só serão comunicadas ao usuário quando uma nova solicitação DNS for realizada. Buscando contornar isso, os CDNs definem um baixo tempo de vida (TTL) para as respostas DNS. Porém, a maioria dos principais navegadores *web* atuais ignoram o TTL e mantém em cache a resposta DNS por mais tempo.

Um ponto positivo dessa cache é que evita requisições DNS excessivas (ou desnecessárias), mas em contrapartida, caso o servidor e/ou *link* venha a falhar ou ficar sobrecarregado o usuário continuará tentando se conectar a esse servidor, mesmo que o DNS já esteja configurado para não direcionar mais os usuários para tal.

A proposta do artigo é implementar um sistema que possibilite as operadoras de rede redirecionar usuários de forma transparente para servidores ideais durante os períodos de congestionamento da rede, falha ou alta carga do servidor que esses usuários pretendem acessar. Para alcançar tal objetivo a solução oferecida é uma Função de Rede Virtualizada (VNF) utilizando o *Click Modular Router*. Essa VNF é basicamente um reescritor de IP (similar ao NAT) que substitui endereços de pacotes com o objetivo de encaminhá-los para os servidores que melhor atendem o usuário no momento. Assim, mesmo se o navegador *web* continuar tentando acessar um servidor lento/inativo (por conta de sua cache DNS), essa VNF irá alterar o endereço de destino dos pacotes para o endereço do servidor ideal.

Esse relatório apresenta os resultados obtidos ao reproduzir alguns dos experimentos conduzidos no artigo e descreve as ferramentas e metodologia utilizada.

As seções estão organizadas da seguinte forma. A Seção II descreve os cenários e especifica quais experimentos estão sendo reproduzidos. Na Seção III são apresentadas a metodologia utilizada e detalhes da implementação. Os resultados obtidos são apresentados na Seção IV. Na Seção V são discutidas algumas dificuldades durante o processo de reprodução. Por fim, a Seção VI conclui o relatório.

2. Cenários

No artigo, para avaliar o sistema proposto, os autores elaboraram uma rede SDN de testes com o emulador Mininet e utilizando o controlador Ryu. Essa rede é apresentada na Figura 2 do artigo, sendo composta por um controlador, um roteador, dois servidores *web*, um cliente *web*, um computador de controle e um dispositivo para executar o recurso NFV. No entanto, essa rede base e pode sofrer leves alterações de acordo com o experimento que está sendo realizado[1].

Todos os experimentos realizados no artigo estão contidos em uma única imagem denominada ”Figura 3”, que apresenta os resultados de testes executados em três distintos cenários.

O primeiro cenário é chamado de ”Falha do *Link* (A)”. Nele, o *link* entre um servidor *web* e o roteador é desativado no instante em que o cliente está acessando esse servidor. Assim é desejável redirecionar todo o tráfego para outro servidor que provê o mesmo conteúdo e esteja ativo.

[1]Por exemplo, um *link* pode ser desativado ou limitado, configurações de rotas podem ser alteradas no roteador, em alguns casos o Reescritor de IP pode não ser usado, etc.

Um cenário denominado ”Congestionamento do *Link* (B)”é o segundo apresentado. Nesse, em um instante específico o *link* entre um servidor e o roteador é limitado pela largura de banda (através de parâmetros no Mininet), simulando um congestionamento. Assim, o servidor continua funcionando, porém de forma precária. Portanto, o ideal é redirecionar o tráfego para o outro servidor, mas tentando manter ativas as conexões TCP pré-existentes, sendo redirecionadas apenas novas conexões.

O terceiro e último é o ”Adicionando um servidor extra (C)”. A princípio esse cenário contém apenas um servidor, que está sob forte congestionamento. Em seguida o outro servidor é adicionado e então é observado quanto tempo leva para o cliente passar a utilizar recursos do servidor adicional.

Esse relatório pretende demonstrar a reprodução dos experimentos conduzidos apenas no cenário A.

3. Metodologia de Implementação

3.1. A Rede SDN

A construção do cenário A foi realizada com o emulador SDN Mininet, em sua versão 2.3.0d4, pois além de popular e simples de usar, é a opção adotada pelos autores do artigo.

No artigo original, o controlador da rede é o Ryu, no entanto, os autores mencionam que embora essa tenha sido a escolha deles, qualquer outro controlador SDN poderia ser utilizado, até mesmo porque o controlador não é o foco da pesquisa e nem faz parte da proposta. Desse modo, por questão de praticidade, a escolha foi o POX, já que ele pode ser instalado junto ao Mininet quando se opta por uma instalação completa.

A topologia da rede ficou similar à apresentada na Figura 2 do artigo, a exceção é o computador de controle, que não foi adicionado. Isso porque a rede apresentada na Figura 2 do artigo é genérica o bastante para dar suporte a todos os cenários, mas no cenário A esse componente não é utilizado. A topologia usada para a reprodução pode ser vista logo a seguir na Figura 1.

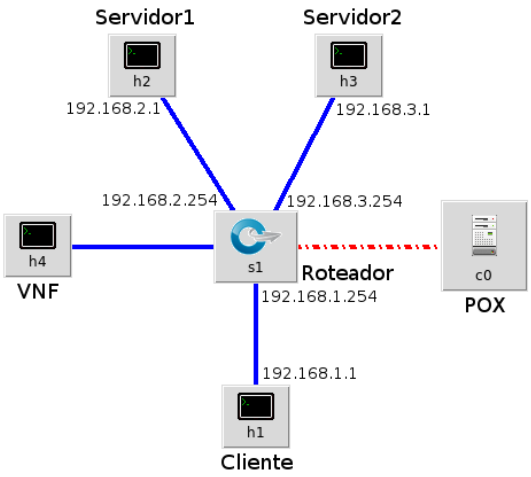

Figura 1. Topologia de rede SDN

Apesar de simples, essa rede representa bem o que pode ser feito com a proposta. Seu ponto central é um *switch OpenFlow* que está conectado a um controlador POX, esse

por sua vez instalará regras de fluxo que farão com que o simples *switch* opere como um roteador. Ligados diretamente ao roteador estão dois servidores *web* que contém o mesmo conteúdo, porém estão em faixas de redes distintas. Também ligado ao roteador, porém em outra faixa IP (diferente de ambos servidores), está o computador cliente que fará as requisições de conteúdo durante os experimentos.

O reescritor de IP (VNF) está contido em outro dispositivo, conectado diretamente ao roteador. Observe que o fluxo entre cliente e servidor não necessáriamente passa através da VNF. Isso indica que esse serviço pode ser alocado em outros pontos da rede (por exemplo, ter outros roteadores entre o reescritor de IP e esse roteador que se conecta aos servidores). Mas para manter a consistência do cenário a VNF foi mantida fisicamente próximo ao roteador (testes com ela em outras posições ficaram como trabalhos futuros no artigo).

3.2. O Roteador e as Regras de Fluxo

O controlador POX foi implementado de modo que no momento de sua inicialização, faça a leitura de um arquivo contendo as regras de fluxo iniciais. Em seguida o controlador instala essas regras no *switch* através de mensagens OpenFlow, tornando-o um roteador capaz de encaminhar corretamente os pacotes entre as dispositivos conectados à ele. Em outras palavras, o *switch* se torna um roteador com rotas estáticas.

Durante os experimentos que não envolvem o Reescritor de IP, os dados trafegam de forma natural, ou seja, saem da origem (cliente), passam pelo roteador e são entregues ao destino (a resposta do servidor segue a mesma lógica). Todavia, quando um servidor fica inativo/congestionado, o POX instala outras regras no roteador, fazendo com que, sempre que pacotes tiverem como destino esse servidor falho, primeiramente eles devem ser encaminhados ao Reescritor de IP (novamente, a resposta segue o mesmo raciocínio). Dessa forma, o Reescritor é capaz de fazer seu papel.

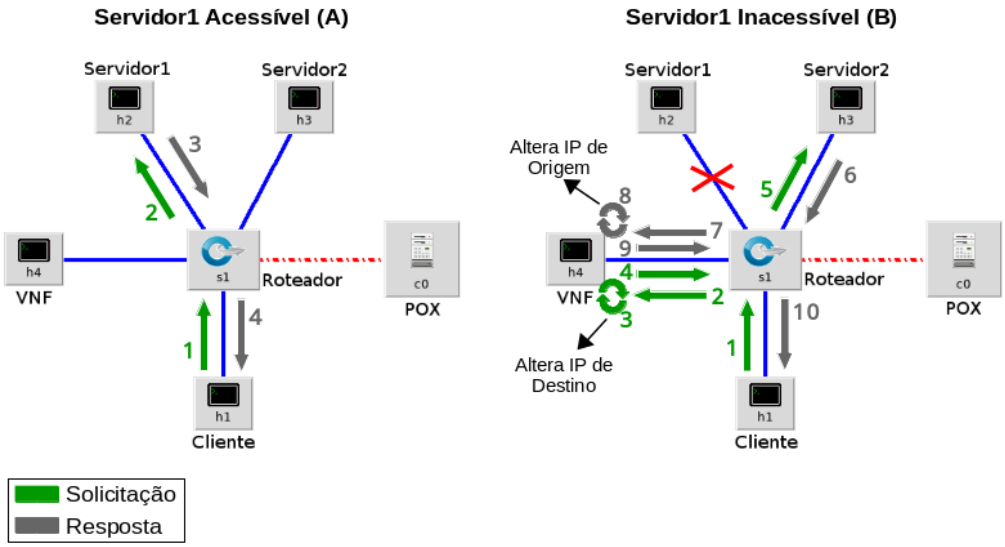

Figura 2. Regras de Fluxos

A Figura 2 demonstra dois casos de exemplo. No Caso A, o Cliente deseja se conectar ao Servidor1, como esse servidor está acessível a comunicação é realizada de

forma direta. Já no caso B, o Servidor1 está inacessível, por isso o roteador foi configurado para encaminhar o tráfego com destino a esse servidor para a VNF. Quando esse tráfego chega a VNF, ela altera o endereço de destino dos pacotes para o IP do Servidor2 e os devolve para o roteador, que faz a entrega ao Servidor2.

3.3. Serviço e Conteúdo Web

3.3.1. Servidores

Para prover o serviço *web* a opção adotada é um simples módulo Python chamado SimpleHTTPServer, que responde a solicitações com métodos GET e HEAD (não trabalhando, por exemplo, com POST, DELETE e outros).

3.3.2. Cliente

Esse computador é um *host* comum do Mininet. Durante os experimentos é a partir dele que as requisições HTTP são feitas usando os mesmos dois navegadores do artigo, Mozilla Firefox e Google Chrome. Portanto, faz-se necessário que ambos navegadores estejam devidamente instalados no sistema hospedeiro do Mininet (computador real).

3.3.3. Página

Para avaliar o desempenho da solução proposta é preciso ter uma carga de trabalho que seja realmente significativa. No artigo é mencionado o uso de um conteúdo de aproximadamente 100 Megabits. No site pessoal de um dos autores esse conteúdo é disponibilizado, o que tornou possível utilizar a mesma carga de trabalho.

A página é constituída basicamente de arquivos de imagem em formato JPG de tamanhos variados, mas que somados alcançam a carga necessária.

3.4. Atualizações de DNS

Nos experimentos as requisições HTTP são realizadas a partir de URLs com o nome de domínio (como geralmente acontece em um ambiente real), não endereços IP. Em casos assim, o cliente precisa descobrir qual é o endereço IP do servidor que hospeda aquele site. Para isso, o primeiro passo é buscar essa informação em um arquivo *'hosts'*, caso não for encontrada, uma solicitação é efetuada para um servidor DNS.

Nessa reprodução não se faz o uso de um servidor DNS, pois as informações são diretamente definidas no arquivo *'hosts'*. Isso facilita a implementação do ambiente e simula perfeitamente o caso em que determinado servidor *web* ficou inativo e o DNS já tem essa informação. A diferença é que essa atualização terá sido feita diretamente no arquivo *'hosts'*[2].

3.5. Reescritor de IP como uma VNF Click

Esse é o principal componente da rede, pois coloca em prática a proposta em sí. Ele é executado como uma VNF, assim (principalmente em um ambiente real) não fica atrelado

[2]Vários testes foram realizados, constatando que o comportamento da *cache* do navegador não se altera.

a uma *midlebox*, tendo em vista que várias VNFs poderiam rodar em um único *hardware* e serem instânciadas de acordo com a demanda. No ambiente experimental para essa reprodução, existe um *host* específico para executá-la.

Da mesma forma que no artigo, o reescritor é implementado com o *Click Modular Router*, que é um roteador de *software* de código aberto. Serviços construídos com Click são elaborados de forma modular, isso porque as funcionalidades são realizadas por módulos distintos. Esses módulos são denominados Elementos, que normalmente possuem uma ou mais portas de entrada de dados (nesse caso, pacotes) e uma ou mais portas de saída. Dados que saem de um elemento (através de uma porta de saída) podem ser encaminhados a uma porta de entrada de outro elemento.

3.5.1. Elementos Click Utilizados

Para capturar os pacotes que chegam no *host* que está executando a função Click é utilizado o elemento *FromDevice*, passando como parâmetro qual é a interface de rede em que ele deve atuar. Os pacotes obtidos através desse elemento podem ser processados pelos demais e ao término podem ser descartados através do elemento *Discard* ou direcionados a uma fila de pacotes montada com o elemento *Queue* (que forma por padrão uma FIFO) e então devolvidos à rede por meio do *ToDevice*.

Após um pacote ser obtido (pelo *FromDevice*), é preciso saber qual é o tipo desse pacote. Tal ação é realizada através do elemento *Classifier*, que nessa implementação foi configurado para direcionar pacotes IP para uma de suas portas de saída (que os entrega a um próximo elemento) e demais pacotes para outra porta (que os descarta).

O próximo passo é identificar se um pacote IP capturado tem como destino um determinado endereço IP (por exemplo, o endereço de um servidor inativo/congestionado). Para isso, pacotes IP são direcionados ao elemento *Strip* que retira o conteúdo *ethernet* do pacote (ação necessária para os próximos elementos agirem), em seguida para o *CheckIPHeader* (que faz uma verificação) e depois ao *IPClassifier*. Esse por sua vez é capaz de identificar quais são os endereços IP de origem e destino, protocolo (TCP ou UDP) e portas de origem e destino do pacote.

Quando o *IPClassifier* identifica que determinado pacote deve ser reescrito (alterado seu destino) o encaminha ao elemento *IPRewriter*, que é capaz de reescrever endereços IP. No entando, o *IPRewriter* precisa do *RoundRobinMapper* para mapear quais alterações estão sendo realizadas (e em quais pacotes) para futuramente ser capaz de desfazer as alterações quando uma resposta chegar.

Após passar pelo *IPRewriter* o pacote é encaminhado ao elemento *EtherEncap*, que adiciona o conteúdo *ethernet* novamente no pacote. O endereço MAC de origem é definido como o da própria interface do dispositivo em que a função está sendo executada, o de destino é o MAC da interface de rede do roteador. Por fim, o pacote é enviado a interface de rede (através do *ToDevice*) que o lançará de volta a rede, mas agora com um novo IP de destino.

Quando um pacote é caracterizado como uma resposta do servidor ao cliente, ele passa por um processo similar, porém o *IPRewriter* identifica que o pacote responsável pela solicitação a qual está sendo respondida passou por aqui e teve seu IP de destino

alterado. Logo o *IPRewriter* modifica o IP de origem desse pacote de resposta para o antigo IP de destino. Isso passa a impressão ao cliente de que está se comunicando com um determinado servidor (que na realidade está inativo/congestionado no momento), sendo que a comunicação está realmente ocorrendo com outro servidor (que oferece o mesmo serviço e está em pleno funcionamento).

4. Resultados

A avaliação do desempenho da ferramenta implementada foi efetuada em um ambiente descrito na Figura 1.

Os passos de execução em todos os cenários são iguais. Um usuário no computador Cliente acessa pelo navegador um *site* com domínio info7015.com e é direcionado a uma pequena página inicial no Servidor1, em seguida uma pausa de 10 segundos é induzida. Nesse período, o *link* entre o Servidor1 e o roteador é desativado e as informações do arquivo *hosts* do Cliente são atualizadas indicando que o domínio info7015.com está hospedado no Servidor2. Após a pausa, o usuário clica em um *link* que o levará a uma página no mesmo domínio de aproximadamente 100 Megabits. A diferença entre um cenário e outro é qual navegador está sendo usado e se a VNF faz parte ou não do cenário.

Os resultados mostrados na Figura 3 representam o tempo gasto para carregar essa página de 100 Mb utilizando os navegadores Google Chrome e Mozilla Firefox, tanto no ambiente sem a VNF quanto com ela. O tempo usado como referência é uma média entre o tempo dos dois navegadores para carregarem a página em um ambiente em que o Servidor1 está em pleno funcionamento.

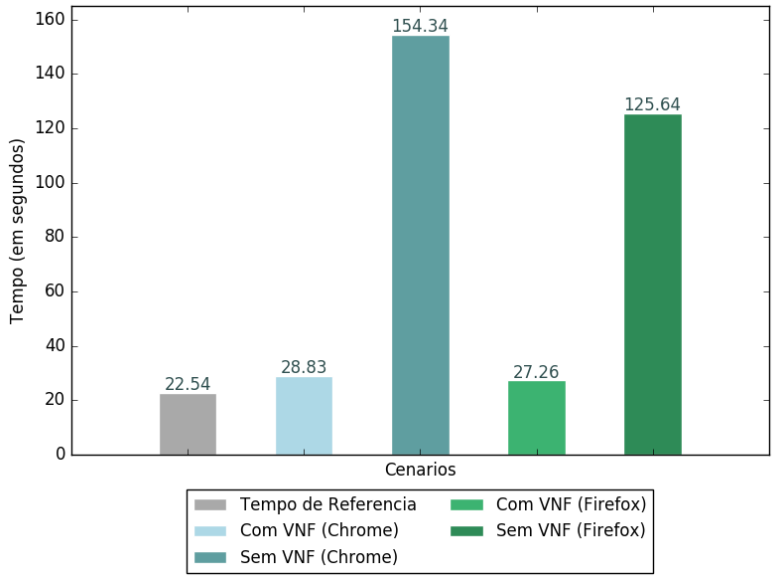

Figura 3. Resultados

Ambos os cenários com o reescritor de IP sendo executado como uma VNF, o tempo dos dois navegadores foram próximos do tempo de referência. Isso demonstra que, apesar do aumento de saltos dos pacotes para chegarem até o destino (como pode ser visto na Figura 2), essa solução melhora consideravelmente o desempenho da conexão.

Em relação aos resultados dos experimentos do artigo, embora todos os valores tenham se divergido, o comportamento permaneceu similar. No entanto, uma pequena diferença ocorreu em relação ao tempo entre os navegadores quando a comunicação é intermediada pelo reescritor de IP, pois no artigo o Chrome obteve desempenho levemente superior ao Firefox e na reprodução a situação se inverteu. Contudo, a diferença é mínima e o resultado confirma a efetividade da solução.

5. Principais Desafios

Ao decorrer do desenvolvimento várias adversidades surgiram. A dificuldade inicial estava relacionada a duas atividades: encaminhar pacotes entre distintas faixas IP e direcionar determinado tráfego para a VNF quando necessário. Embora soubesse que a solução para ambas era utilizar um roteador, não foi possível localizar algo pré-pronto para ser usado. Na realidade, alguns exemplos são disponibilizados em *sites* oficiais dos controladores Ryu (usado no artigo) e POX (instalado junto ao Mininet), mas são roteadores automatizados (sem configuração prévia de rotas) ou que não oferecem suporte a entradas de fluxos mais específicas.

Para atender às necessidades, foi desenvolvido um roteador estático simplificado, que ao inicializar lê um arquivo de regras de fluxo (escrito em JSON) e as emite ao *switch* via mensagens *OpenFlow*.

Outro desafio foi encontrar uma forma confiável para medir o tempo de carregamento da página. O objetivo era conseguir medir sem intervenção humana para evitar distorções. O método adotado foi capturar pacotes no computador Cliente de modo a salvar os resultados em um arquivo e posteriormente executar um *script* que, através do tempo registrado nos pacotes (*timestamp*), calcula o tempo total de carregamento.

6. Conclusão

Esse relatório descreveu o desenvolvimento e reprodução dos experimentos de avaliação de um reescritor de IP que aprimora os mecanismos de redirecionamento de usuários a servidores em CDNs.

A implementação do ambiente e a ferramenta possibilitou observar os efeitos colaterais da cache DNS realizada pelos navegadores, bem como a otimização proporcionada pela solução proposta.

Os resultados foram similares ao artigo original e indicam que o reescritor de IP pode refletir possitivamente em ambientes SDN reais, melhorando consideravelmente o tempo de acesso a conteúdos em momentos de falha ou congestionamento tanto do servidor, quanto do *link* ao qual está conectado.

Todos os códigos usados para construir o ambiente emulado e executar os experimentos (incluindo um detalhado tutorial) estão disponíveis em https://github.com/joseflauzino/INFO7015-TP3.

Referências

Kohler, E., Morris, R., and Poletto, M. (2002). Modular components for network address translation. IEEE Open Architectures and Network Programming Proceedings.

Lai, J., Fu, Q., and Moors, T. (2015). Rapid ip rerouting with sdn and nfv. IEEE Global Communications Conference (GLOBECOM).

Lantz, B., Handigol, N., Heller, B., and Jeyakumar, V. (2018). Introduction to mininet. https://github.com/mininet/mininet/wiki/Introduction-to-Mininet. Acessado em dezembro de 2018.

McCauley (2015). Openflow in pox. https://noxrepo.github.io/pox-doc/html/#openflow-in-pox. Acessado em dezembro de 2018.