OrchFlow - Uma Ferramenta para Orquestração de Múltiplos Controladores OpenFlow

Marcelo Frate[1], Marcelo K. M. Marczuk[2], Fábio L. Verdi[2]

[1]Instituto Federal de São Paulo (IFSP)
Boituva – SP – Brasil

[2]Universidade Federal de São Carlos (UFSCar)
Sorocaba, SP – Brasil

frate@ifsp.edu.br, marcelo.marczuk@dcomp.sor.ufscar.br, verdi@ufscar.br

Abstract. *The main purpose of software-defined networks is the centralization of control logic, but it is possible to divide this logic between two or more controllers in order to ensure scalability. The OpenFlow protocol defines the communication between switches and controllers, but does not provide for communication between controllers, required for any type of distribution in the control plan. Therefore it is necessary to develop independent solutions of the protocol, capable of delivering this logic within a single administrative domain. In this scenario, the OrchFlow emerges as a tool capable of orchestrating a network defined by software, with two or more controllers OpenFlow, enabling management and monitoring of real-time topology.*

Resumo. *O principal objetivo das redes definidas por software é a centralização da lógica de controle, porém é possível dividir esta lógica entre dois ou mais controladores com o intuito de garantir a escalabilidade. O protocolo OpenFlow define a comunicação entre switches e controladores, entretanto não prevê a comunicação entre controladores, necessária para qualquer tipo de distribuição no plano de controle. Faz-se necessário portanto o desenvolvimento de soluções independentes do protocolo, capazes de distribuir essa lógica dentro de um mesmo domínio administrativo. Neste cenário, o OrchFlow surge como uma ferramenta capaz de orquestrar uma rede definida por software, com dois ou mais controladores OpenFlow, permitindo a gerência e o monitoramento da topologia em tempo real.*

1. Introdução e Motivação

As redes baseadas no protocolo (IP) [Postel et al. 1981], estão evoluindo, deixando a orientação ao hardware para a orientação ao software, assim, *Software Defined Networking* (SDN) [Greenberg et al. 2005] e *Network Functions Virtualization* (NFV) [Cui et al. 2012] começam a se firmar como o futuro das infraestruturas de rede. Esses conceitos estão revolucionando a forma como se operam as redes de telecomunicações no mundo, respondendo às mudanças da crescente demanda de tráfego pelos usuários e empresas. Para um *Data Center* o SDN responde principalmente à infraestrutura e é uma opção para o aprovisionamento de serviços em nuvem, centralizando toda a administração dos equipamentos, definindo o funcionamento lógico da rede.

Há porém, razões para se usar um plano de controle distribuído, tais como: administração, distribuição geográfica, escalabilidade, redução de latência entre um switch e seu controlador, tolerância a falhas e balanceamento de carga. Isso dificulta a tarefa de gerenciamento dentro de um único domínio administrativo onde vários controladores podem dividir a tarefa da gerência e controle dos inúmeros switches. A programação aplicada aos controladores precisa levar em consideração a demanda das aplicações, a fim de garantir que a infraestrutura e cada uma das camadas de rede estejam disponíveis para prover de forma autônoma cada um dos serviços necessários para que uma aplicação esteja disponível o mais rápido possível. Para isso é necessário o desenvolvimento de sistemas integradores que realizem a comunicação entre dois ou mais controladores.

2. Trabalhos Relacionados

Atualmente, diversas propostas para as SDN têm feito uso de múltiplos controladores com o objetivo claro de se obter um plano de controle descentralizado. O protocolo OpenFlow a partir de sua versão 1.2, torna possível a comunicação entre os switches e diversos controladores, possibilitando a criação de sistemas de *backups* e redundância. Entretanto, o OpenFlow não define a comunicação necessária, entre controladores, para qualquer tipo de distribuição no plano de controle. Assim, é preciso que sejam desenvolvidas novas soluções, independentes do protocolo, capazes de distribuir essa lógica.

Alguns trabalhos encontrados na literatura tem por objetivo proporcionar a escalabilidade das redes, como em **HyperFlow** [Tootoonchian and Ganjali 2010], implementado como uma aplicação do controlador NOX [Gude et al. 2008], baseada em eventos, permite implantar qualquer número de controladores, em **DISCO** [Phemius et al. 2014], que procura compartilhar o estado da rede, mantendo a comunicação em cascata entre controladores vizinhos e em **Onix** [Koponen et al. 2010], que executado em um *cluster* de um ou mais servidores físicos, utiliza um modelo de dados chamado *Network Information Base* (NIB), este mantém o estado da rede através de uma estrutura de dados que armazena um grafo de todas as entidades da topologia.

Outras propostas buscam desenvolver os seus próprios controladores, com protocolos próprios de comunicação para as interfaces leste-oeste, como o **Kandoo** [Yeganeh and Ganjali 2012], que utiliza os controladores de forma hierárquica, passando a existir um controlador raiz no topo da rede, interligando os diversos controladores locais.

3. OrchFlow

OrchFlow, uma ferramenta que tem como objetivo, funcionar como um *Middleware*, um software orquestrador para as SDN baseadas no protocolo OpenFlow [McKeown et al. 2008], proporciona ao administrador da rede uma visão global, capaz de aprovisionar serviços, com monitoramento da topologia em tempo real. O OrchFlow, é capaz de receber solicitações de serviços através de uma interface Norte, *(Northbound)*, processá-las e então mapeá-las através de uma interface Sul *(Southbound)*, de forma a prover o serviço solicitado em uma rede controlada por múltiplos controladores OpenFlow. Assim, quando um determinado serviço é solicitado, o OrchFlow define como deverá ser o tratamento por parte dos diversos controladores na rede até que sejam aplicadas

todas as regras OpenFlow aos elementos de rede, permitindo que os recursos estejam disponíveis.

3.1. Funcionamento do OrchFlow

Na primeira camada da arquitetura estão os switches, formando toda a infraestrutura necessária para a comunicação entre os hosts. Para cada subconjunto de switches, definidos aqui como subdomínio, existe um controlador, responsável pela configuração, por manter as informações da topologia e monitorar o estado da rede. Cada controlador está conectado ao OrchFlow através de uma interface Central.

O OrchFlow, atua como um orquestrador, definindo a programação para todos os controladores do domínio administrativo e possibilita a comunicação entre as diferentes aplicações através de uma interface Norte, uma interface padronizada, capaz de receber solicitações e invocar serviços pré-determinados. Essas interfaces fazem parte de um único sistema WEB, utilizando *Representational State Transfer* (REST) [Fielding 2000], um protocolo que torna possível a troca de informações entre aplicativos e serviços web, pela qual são solicitados todos os recursos necessários para o estabelecimento das rotas fim a fim. Ao receber tais solicitações, o OrchFlow as processa e de forma orquestrada atuará sobre cada um dos controladores por duas maneiras possíveis, conforme o subdomínio a ser alcançado.

Pró-ativo: A aplicação inicia, via interface Norte, a solicitação de um serviço ao OrchFlow, que faz toda a orquestração necessária aos controladores via interface Central. Neste modo, toda a programação é executada pelo OrchFlow e transferida aos controladores, via interface REST padrão, todas as regras de fluxos necessárias, para que haja a comunicação fim a fim, diretamente aos switches envolvidos;

Reativo: Um novo evento na rede, chegando através da interface Sul, faz com que, um controlador busque em sua programação a correta correspondência e faz a inserção dos fluxos necessários. Para isso, o OrchFlow disponibilizará uma interface de programação central que possibilitará ao administrador da rede, criar a sua programação de forma centralizada e descarregá-las nos controladores. Neste modo, os controladores OpenFlow, respondem diretamente às requisições de seu subdomínio e encaminham para a porta de saída correspondente ao subdomínio vizinho todas as requisições que não pertençam a este subdomínio, possibilitando assim a entrega de pacotes entre subdomínios, independentemente de quantos forem os subdomínios que compõem o domínio administrativo.

3.2. Arquitetura do OrchFlow

Como se pode ver na Figura 1, o OrchFlow atuará como um agente integrador entre as diversas aplicações disponíveis na rede e os diferentes controladores OpenFlow, sob um mesmo controle administrativo, definido aqui como Domínio Administrativo .

4. Detalhes de implementação

A arquitetura de rede proposta, tem para efeito de testes, um domínio administrativo composto por três subdomínios, com um controlador OpenFlow cada. Cada controlador é responsável pelo controle e gerenciamento de um único subdomínio e cabe ao OrchFlow a orquestração, gerencia e a visão completa do Domínio Administrativo.

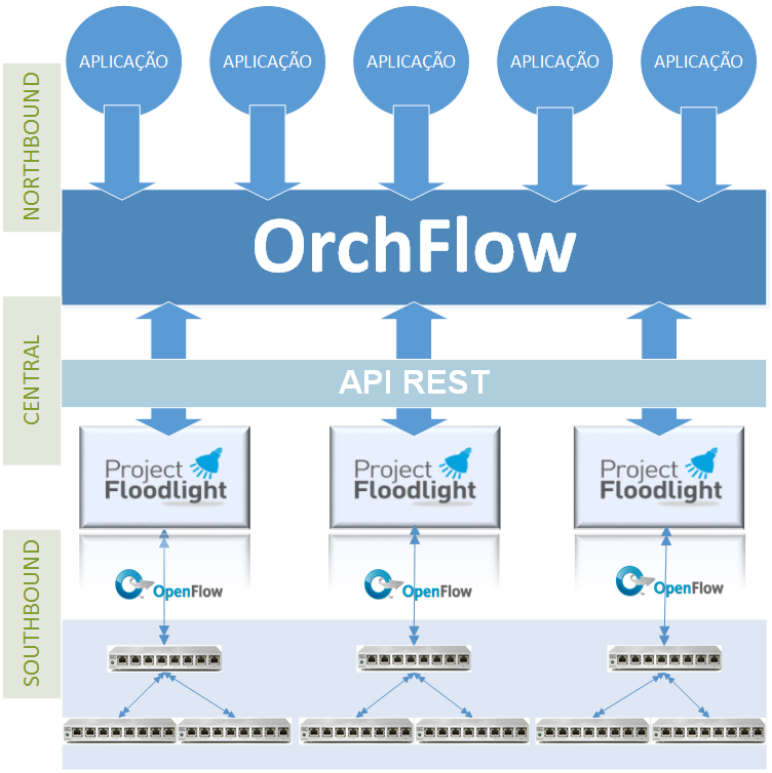


Figure 1. Arquitetura utilizada nos testes.

A Figura 2, ilustra a topologia utilizada nos testes e no desenvolvimento do OrchFlow. Quatro máquinas virtuais (VM) são utilizadas:

VM1: Uma rede é emulada através do emulador Mininet [Lantz et al. 2010], contendo três redes interligadas em forma de anel, com 7 switches cada em forma de árvore e 4 hosts;

VM2, VM3 e VM4: Em cada uma, um controlador Floodlight está funcionando e pronto para receber as chamadas dos switches das redes instaladas na VM 1;

Toda a infraestrutura necessária para a implantação e testes do OrchFlow está baseada na plataforma Linux, desenvolvido e implementado em linguagem Java e um banco de dados não relacional, o Neo4j. As três redes estão definidas e interligadas através de links específicos, denominados aqui de links externos, utilizando ainda switches virtuais OVS-Switch e o protocolo OpenFlow na versão 1.3. E para a realização dos testes, alguns dos serviços de rede mais conhecidos estão sendo propostos, como os serviços de roteamento fim a fim, como forma de validar esta ferramenta.

5. Desenvolvimento do Módulo

O desenvolvimento do OrchFlow busca proporcionar a integração, a gerência, a administração e a programação para SDN baseada em múltiplos controladores OpenFLow, para isso ele deverá ser capaz de ler e gravar as diversas programações em cada um dos controladores, de forma a garantir a perfeita comunicação entre quaisquer hosts pertencentes ao mesmo domínio administrativo.

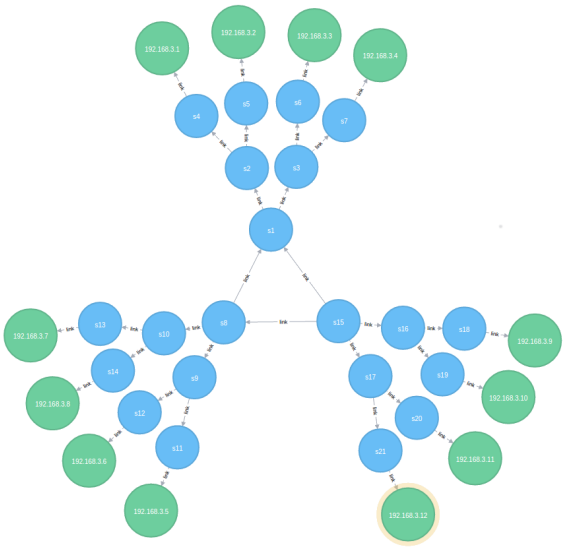


Figure 2. Topologia de rede.

5.1. Banco de Dados - Neo4j

Para que haja a correta leitura e consolidação dos dados é necessário armazená-los de forma eficiente e que os dados estejam bem estruturados. Assim, foi escolhido o banco de dados Neo4j, que como se pode ver em [Eifrem 2009], trata-se de um software livre baseado em Java, com banco de dados em grafo, que oferece persistência, alto desempenho, escalabilidade, uma comunidade ativa e uma boa documentação, capaz de receber os dados de cada um dos hosts e switches e representá-los como nós, obter os dados dos links e criar um relacionamento entre eles, estabelecendo um conjunto completo de dados, oferecendo ainda uma interface gráfica, permitindo ao administrador obter a visão completa da rede, de todo o domínio administrativo.

Para que o sistema funcione, é necessário que os controladores reconheçam a topologia da rede, para isso todos os controladores contam com um módulo de reconhecimento de topologia de seu subdomínio, assim, cada elemento de rede, bem como os hosts são identificados e depois esses dados são encaminhados através de uma interface REST para o OrchFlow que fará a composição das informações obtidas de todos os controladores da rede, integrando-as numa só base de dados, o NEo4j, à partir de onde serão realizadas todas as consultas.

Todas as rotas definidas pelo administrador ou por solicitação de alguma aplicação ao OrchFlow, serão criadas tomando por base o algoritmo de Dijkstra [Dijkstra 1959], que faz uma busca pelo caminho mais curto e determina aos controladores que criem os fluxos seguindo conforme o resultado encontrado por ele. Assim, qualquer rota, mesmo as que passam por dois ou mais subdomínios, serão definidas por seus controladores seguindo conforme o que foi encontrado pelo OrchFlow em sua topologia.

Para efeito de testes e funcionamento exclusivo com o OrchFlow, foram desabilitados todos os módulos *forward* que permitam o funcionamento como switches legados, assim para todo e qualquer fluxo que chegue à rede, é necessário que haja um correspondente na tabela de fluxos, caso não haja, os controladores devem possuir uma programação prévia, para determinarem a configuração à todos os switches envolvidos na comunicação.

Porém, caso um fluxo novo chegue à rede e o mesmo não encontrar correspondência na tabela de fluxos ou não houver uma programação que possa tratá-lo de forma correta, este será descartado.

5.2. Módulo ARPReply

Toda a comunicação na rede, baseada no protocolo TCP-IP exige o uso do endereço IP, que é utilizado para roteamento, ou seja, para a escolha do melhor caminho entre dois hosts, Para isso, é necessário mapear o endereço de nível superior IP para endereço físico Ethernet, como foi proposto pela RFC826, o Address Resolution Protocol (ARP) [Plummer 1982]. O ARP permite que um host encontre o endereço físico de um host destino, tendo apenas o seu endereço IP. Entretanto, numa rede OpenFlow, é necessário que os fluxos ARP sejam tratados previamente, para que os fluxos TCP sejam encaminhados.

Nesse sentido, cada controlador, reconhece a topologia de seu subdomínio e a repassa ao OrchFlow, de posse das informações de todo o domínio administrativo cria-se então uma tabela de conversão MAC/IP e a devolve aos controladores através de uma interface REST, onde um módulo previamente instalado irá possibilitar que os controladores respondam às requisições ARP, permitindo que todos os hosts na rede recebam os endereços MAC destino para cada novo fluxo que solicitarem, assim, mesmo que um host solicite uma comunicação com outro host fora de seu subdomínio, o controlador já terá os dados necessários para devolução da correta mensagem de retorno do protocolo ARP. Essa solução traz um ganho à rede, eliminando o broadcast de mensagens ARP desnecessárias.

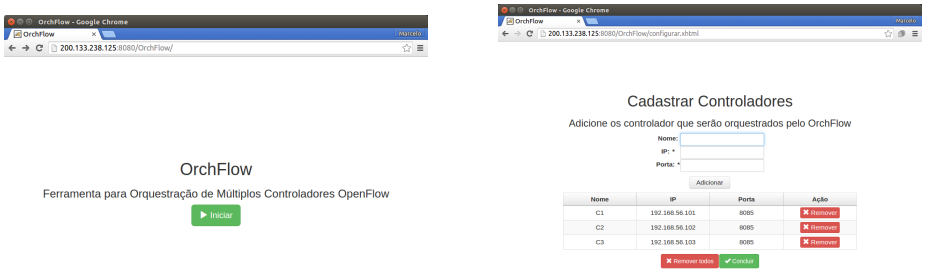
5.3. Módulo Reactive

Como vimos na seção 3.1 a programação dos controladores é realizada de duas maneiras, nos modos proativo e reativo. No modo proativo, um serviço ou ação do administrador da rede, solicita ao OrchFlow que encaminhe os parâmetros aos controladores, para que sejam gravados diretamente na tabela de fluxos dos switches, possibilitando assim, que todos os fluxos sejam encaminhados diretamente às portas de destino.

Entretanto, para que o modo reativo funcione é necessário a instalação do módulo *Reactive* em cada um dos controladores, esse módulo permite às aplicações ou ao administrador da rede que faça a programação de seu domínio administrativo em tempo de execução. Neste modo, um serviço ou ação do administrador da rede, solicita ao OrchFlow que encaminhe aos controladores a programação necessária para que, quando um fluxo chegar e não encontrando nas tabelas de fluxos do primeiro switch um fluxo correspondente, este receba do controlador do subdomínio, o preenchimento de sua tabela, liberando o fluxo a partir deste momento.

6. Interface

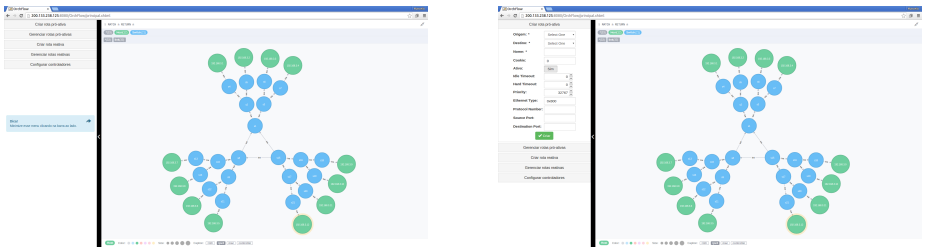
A ferramenta apresenta uma interface limpa e objetiva, após o click no botão iniciar em 3(a) você será redirecionado para a tela de cadastro dos controladores da rede 3(b), é preciso cadastrar todos os controladores, colocando nome, endereço IP da interface web e porta http, como por exemplo: Controlador1, 192.168.1.1, 8085. Este procedimento é obrigatório, pois só assim o OrchFlow poderá realizar as consultas e determinar as configurações para cada subdomínio.



(a) Home (b) Configurar

Figure 3. OrchFlow: Inicialização do sistema.

Uma vez cadastrado todos os controladores, basta clicar em concluir e o sistema estará pronto para uso, já é possível realizar todas as programações possíveis.



(a) Principal (b) Programação

Figure 4. OrchFlow: Programação e Uso.

Você terá do lado direito da página 4(a) a topologia da rede, uma interface Neo4j, onde é possível executar os comandos desse banco de dados, tais como uma mudança de cor para os nós e arestas, mudança no tamanho dos nós, identificadores, entre outros. É possível realizar também comandos de busca de melhor caminho entre dois hosts, enfim, todos os comandos do Neo4j estão livres para uso.

Do lado esquerdo da interface é possível identificar os dois modos de operação do OrchFlow no menu de opções e criar as regras conforme a necessidade da aplicação. Para ambos os modos o procedimento é o mesmo, basta preencher os campos necessários para a comunicação.

Para ajudar o leitor, deixaremos um manual de instalação e uso disponível em https://github.com/marcelofrate/OrchFlow/ onde todos os detalhes de implantação e possíveis testes estarão explicados passo a passo, de forma a contribuir com comunidade acadêmica em busca de novas soluções aos problemas de escala em SDN.

7. Conclusão

O OrchFlow, possibilita que as aplicações tirem proveito de todas as funcionalidades dos recursos físicos da infraestrutura de uma rede baseada em software de forma automatizada, espera-se demonstrar que a integração entre controladores diferentes e a sua orquestração responda aos problemas de escala que atualmente existem nas SDN baseadas no protocolo OpenFlow.

O OrchFlow, uma ferramenta capaz de automatizar o processo de programação de múltiplos subdomínios, integrados à um único domínio administrativo, com uma visão global, centralizada, possibilitando um funcionamento hibrido, nos modos proativo e reativo, dando aos fluxos de maior frequência a velocidade necessária e aos fluxos esporádicos regras que possuem um tempo delimitado. Tais características aproveitam melhor os recursos do hardware, como o uso da memória TCAM, possibilitando maior vida útil aos switches OpenFlow, com uma consequente redução dos custos com a tecnologia.

References

Cui, C., Deng, H., Telekom, D., and Michel, U. (2012). Network Functions Virtualisation. *Citeseer*, (1):1–16.

Dijkstra, E. (1959). A note on two problems in connexion with graphs:(numerische mathematik, _1 (1959), p 269-271).

Eifrem, E. (2009). Neo4j-the benefits of graph databases. *no: sql (east)*.

Fielding, R. T. (2000). Architectural Styles and the Design of Network-based Software Architectures. *Building*, 54:162.

Greenberg, A., Hjalmtysson, G., Maltz, D. A., Myers, A., Rexford, J., Xie, G., Yan, H., Zhan, J., and Zhang, H. (2005). A clean slate 4d approach to network control and management. *SIGCOMM Comput. Commun. Rev.*, 35(5):41–54.

Gude, N., Koponen, T., Pettit, J., Pfaff, B., Casado, M., McKeown, N., and Shenker, S. (2008). Nox: Towards an operating system for networks. *SIGCOMM Comput. Commun. Rev.*, 38(3):105–110.

Koponen, T., Casado, M., Gude, N., Stribling, J., Poutievski, L., Zhu, M., Ramanathan, R., Iwata, Y., Inoue, H., Hama, T., Others, and Shenker, S. (2010). Onix: A distributed control platform for large-scale production networks. *OSDI, Oct*, pages 1—-6.

Lantz, B., Heller, B., and McKeown, N. (2010). A network in a laptop: rapid prototyping for software-defined networks. . . . *Workshop on Hot Topics in Networks*, pages 1–6.

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

Phemius, K., Bouet, M., and Leguay, J. (2014). DISCO: Distributed multi-domain SDN controllers. *IEEE/IFIP NOMS 2014 - IEEE/IFIP Network Operations and Management Symposium: Management in a Software Defined World*.

Plummer, D. C. (1982). Rfc 826: An ethernet address resolution protocol. *InterNet Network Working Group*.

Postel, J. et al. (1981). Rfc 791: Internet protocol.

Tootoonchian, A. and Ganjali, Y. (2010). Hyperflow: a distributed control plane for openflow. *Proceedings of the 2010 internet network . . .* , pages 3–3.

Yeganeh, S. H. and Ganjali, Y. (2012). Kandoo: a framework for efficient and scalable offloading of control applications. *Proceeding HotSDN '12 Proceedings of the first workshop on Hot topics in software defined networks*, pages 19–24.