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 the software defined networks is to centralize the control logic. However, it is possible to split this logic among 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 plane. It is necessary, therefore, to develop independent protocol solutions, capable of distributing this logic within the same administrative domain. In this scenario, we developed OrchFlow, a tool capable of orchestrating a software defined network with two or more OpenFlow controllers, enabling the management and monitoring of real-time topologies.*

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, desenvolvemos o OrchFlow, 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 pelo mundo, respondendo às mudanças da crescente demanda de tráfego pelos usuários e empresas. Para um *Data Center* o SDN responde principalmente à infraestrutura como 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, dificultando a tarefa de gerenciamento dentro de um único domínio administrativo.

O objetivo deste trabalho é propor o uso de um software orquestrador para o aprovisionamento de serviços em redes definidas por software com múltiplos controladores OpenFlow, proporcionando ao administrador da rede uma visão global de seu domínio administrativo e um controle total das ações, independentemente de quantos e quais controladores serão utilizados. O OrchFlow permite, portanto, que os serviços fim a fim dentro do domínio administrativo continuem sendo fornecidos, assim como, serviços exclusivos podem ser criados para cada subdomínio.

2. Trabalhos Relacionados

Atualmente, diversas propostas para as redes SDN têm feito uso de múltiplos controladores com o objetivo 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 entre controladores, necessária 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 têm por objetivo proporcionar a escalabilidade das redes. O **HyperFlow** [Tootoonchian and Ganjali 2010] é implementado como uma aplicação do controlador NOX [Gude et al. 2008], requerendo alterações de código no núcleo do controlador, para interceptar comandos e serializar eventos. **DISCO** [Phemius et al. 2014] procura compartilhar o estado da rede, mantendo a comunicação em cascata entre controladores vizinhos, apresentando também a necessidade de se reescrever o código do controlador Floodlight, tornando a proposta totalmente dependente deste. Por fim, **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), com uma contribuição relevante, mas, mesmo tendo por base a utilização de softwares de código aberto como o controlador NOX e do Apache ZooKeeper, foi desenvolvido em código fechado, o que impossibilita a integração e o desenvolvimento de novas aplicações.

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. Embora esta abordagem tenha seus méritos, ela restringe ao controlador Kandoo, o que significa utilizar unicamente o controlador definido pelo grupo e atualmente o Kandoo suporta apenas a versão 1.0 do protocolo OpenFlow, dificultando o desenvolvimento de novos experimentos.

3. OrchFlow

As redes SDN concentram a inteligência através de um controlador, oferecendo um controle logicamente centralizado, possibilitando a existência de um ou mais controladores físicos. Entretanto, há diversas 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 dividindo a topologia da rede.

Porém, apesar de ser evidente a necessidade de subdividir a topologia para que cada parte seja gerenciada de maneira independente, há ainda a necessidade de gerenciamento e oferecimento de serviços globais que envolvem os diversos subdomínios. Um desses clássicos exemplos é o serviço de roteamento fim a fim.

Assim, desenvolvemos o OrchFlow, uma ferramenta que tem como objetivo funcionar como um *Middleware*, um software orquestrador para as redes SDN baseadas no protocolo OpenFlow [McKeown et al. 2008], 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. Quando um determinado serviço é solicitado, o OrchFlow define como deverá ser o tratamento por parte dos diversos controladores até que sejam aplicadas as regras OpenFlow a todos os elementos de rede.

3.1. Funcionamento do OrchFlow

Como mostrado na Figura 1, a primeira camada da arquitetura contém 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, 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. O OrchFlow possibilita a comunicação entre as diferentes aplicações através de uma interface Norte, 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 de serviços fim a fim. Ao receber tais solicitações, o OrchFlow as processa e de forma orquestrada atua sobre cada um dos controladores conforme o subdomínio a ser alcançado, de duas maneiras possíveis:

Proativo: 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, que aplica todas as regras de fluxos necessárias para o estabelecimento de serviços fim a fim, diretamente aos switches envolvidos. 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. Os parâmetros a serem enviados aos switches são aqueles padronizados pela especificação OpenFlow, que permitem um controle total dos fluxos conforme as políticas do administrador do domínio.

Reativo: No modo reativo, as regras são criadas e enviadas para os controladores, porém, diferentemente do modelo Proativo, elas não são implantadas imediatamente nos

switches. Tais regras serão implantadas em resposta a um evento *packet-in*. Este evento é enviado através da interface Sul (OpenFlow) para o controlador do subdomínio. O controlador fará então uma busca em sua programação e, caso haja uma correspondência, ele adiciona as regras nas tabelas de fluxos de todos os switches de seu subdomínio, necessários para o estabelecimento do serviço solicitado.

Para que o modelo Reativo funcione, o OrchFlow disponibiliza uma interface que possibilita ao administrador da rede criar a programação de forma centralizada, descarregando as regras nos controladores conforme o serviço a ser fornecido.

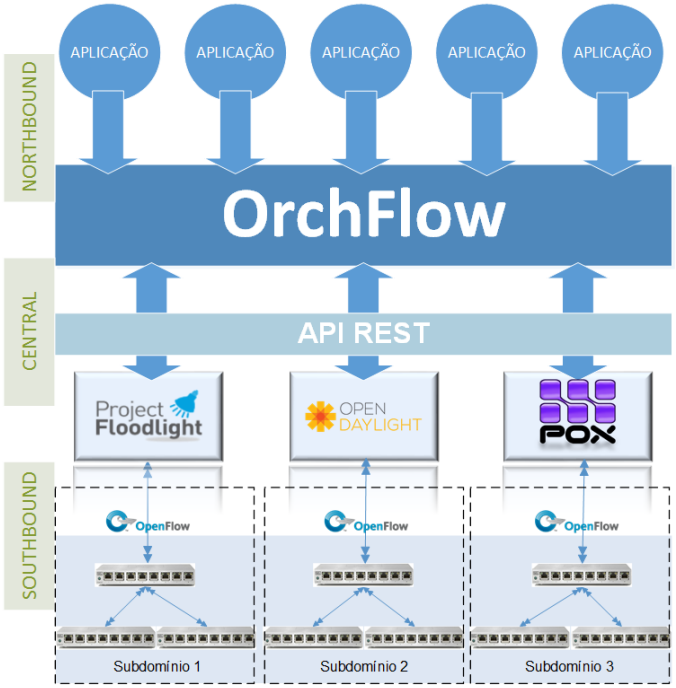

Figura 1. Arquitetura do OrchFlow.

4. Implementação e Validação

A topologia proposta tem, para efeito de testes, um domínio administrativo composto por três subdomínios, com um controlador OpenFlow em cada subdomínio. 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.

A Figura 2 ilustra a topologia utilizada nos testes 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: Para o atual estágio de desenvolvimento e testes, temos em cada VM um controlador Floodlight funcionando e pronto para receber os eventos vindos dos switches das redes emuladas na VM1;

O OrchFlow foi desenvolvido e implementado em linguagem Java e utiliza um banco de dados não relacional, o Neo4j [Eifrem 2009]. As três redes estão definidas

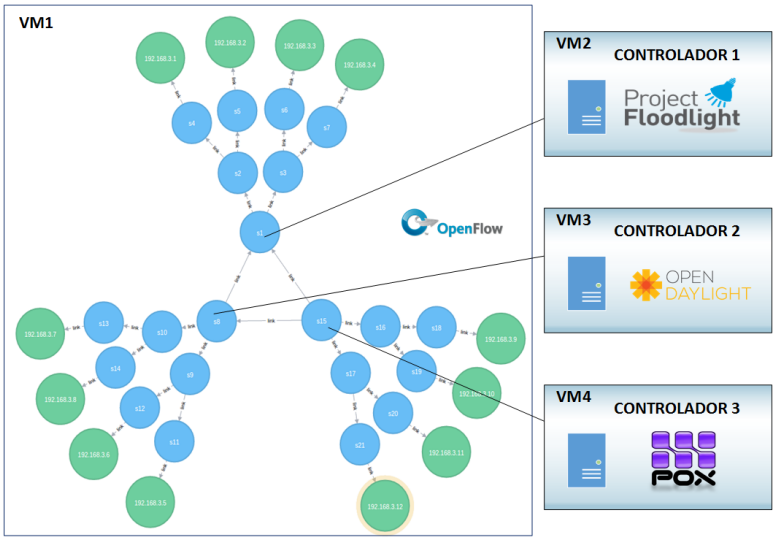

Figura 2. Topologia de rede.

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. Para a realização dos testes, escolhemos um dos serviços de rede mais conhecidos, o serviço de roteamento fim a fim. Neste artigo, cada subdomínio possui o controlador Floodlight. O uso de diferentes controladores será explorado em trabalhos futuros.

O desenvolvimento do OrchFlow busca proporcionar a integração, a gerência, a administração e a programação para as redes SDN baseada em múltiplos controladores OpenFLow. Para isso ele deve ser capaz de ler e gravar as diversas programações em cada um dos controladores, de forma a garantir o funcionamento dos serviços entre quaisquer hosts pertencentes ao mesmo domínio administrativo, mesmo estando localizados em diferentes subdomínios.

4.1. Banco de Dados - Neo4j

Para que haja a correta leitura e consolidação dos dados é necessário armazená-los de forma eficiente e bem estruturada. Assim, foi escolhido o Neo4j [Eifrem 2009], um banco de dados orientado a grafos baseado em Java que oferece persistência, alto desempenho, escalabilidade e possui uma comunidade ativa e uma boa documentação. Neste trabalho, o Neo4j recebe os dados de cada um dos hosts e switches e representa-os como nós, obtém os dados dos links e cria um relacionamento entre eles, possibilitando ao administrador obter uma visão completa de todo o domínio administrativo, através de uma interface gráfica.

Para que o sistema funcione, é necessário o reconhecimento de toda a topologia da rede. O OrchFlow obtém estes dados de cada controlador através da API REST já definida em cada controlador. Note que a maioria, se não todos os controladores, possuem uma chamada em REST para obtenção da topologia. Sendo assim, o OrchFlow se aproveita desta interface já existente para construir a topologia completa do domínio administrativo e armazená-la no Neo4J.

Uma vez que se tenha armazenado todos os dados relacionados à rede, pode-se obter o melhor caminho entre dois hosts e implementar o serviço de roteamento fim a

fim. Neste trabalho, todas as rotas definidas pelo administrador ou por solicitação de alguma aplicação ao OrchFlow são 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 conforme o resultado encontrado. Assim, qualquer rota, mesmo as que passam por dois ou mais subdomínios, serão definidas por seus controladores conforme o que foi definido pelo OrchFlow.

Para efeito de testes e funcionamento exclusivo com o OrchFlow, foram desabilitados todos os módulos *forward* de forma que não interfiram no funcionamento da rede. Assim, para todo e qualquer fluxo que chegue à rede, é necessário que haja uma regra correspondente na tabela de fluxos (modo Proativo). Caso não haja, os controladores devem possuir uma programação prévia para determinarem a configuração a todos os switches envolvidos no serviço solicitado (modo Reativo). 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.

4.2. Módulos ARPReply e Reactive

O modelo tradicional de abordagem para tratamento do pacote ARP REQUEST visa o envio da mensagem em broadcast. Nesse modelo, quando um switch não conhece o destino, realiza broadcast para todas as portas de saída, menos aquela onde recebeu a mensagem. Neste trabalho, desenvolvemos um módulo responsável por responder às mensagens de ARP dentro do domínio administrativo. Este módulo visa a criação de um pacote ARP REPLY para que a requisição ARP REQUEST seja respondida pelo controlador. Nela, o controlador usa todas as estruturas que possui para montar a resposta de um pacote ARP e enviá-lo de maneira transparente ao host solicitante.

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. Tal tabela é então enviada para cada controlador através de uma interface REST estendida. O módulo ARPReply previamente instalado irá armazenar esta tabela para montar respostas às requisições ARP. Assim, mesmo que um host solicite um serviço 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.

Além do ARPReply, também desenvolvemos o módulo *Reactive* responsável por receber as regras do OrchFlow e armazená-las em uma estrutura interna. Estas regras serão posteriormente utilizadas para tratamento de eventos de *packet-in* originados na rede. Este modo de ação permite que se use as tabelas de fluxos de maneira mais eficaz diminuindo as entradas na memória TCAM de cada switch na rede.

4.3. Interface

A ferramenta apresenta uma interface simples onde os serviços a serem aprovisionados no domínio administrativo podem ser solicitados. Inicialmente, os controladores precisam ser cadastrados. Após isso, regras OpenFlow podem ser definidas e aplicadas nos subdomínios através de seus respectivos controladores.

Uma vez cadastrados todos os controladores, o sistema estará pronto para uso, carregando a página conforme a Figura 3. Nesta página é possível analisar a topologia da

rede, alterando cores, mudando posições de nós e links dentre outras opções.

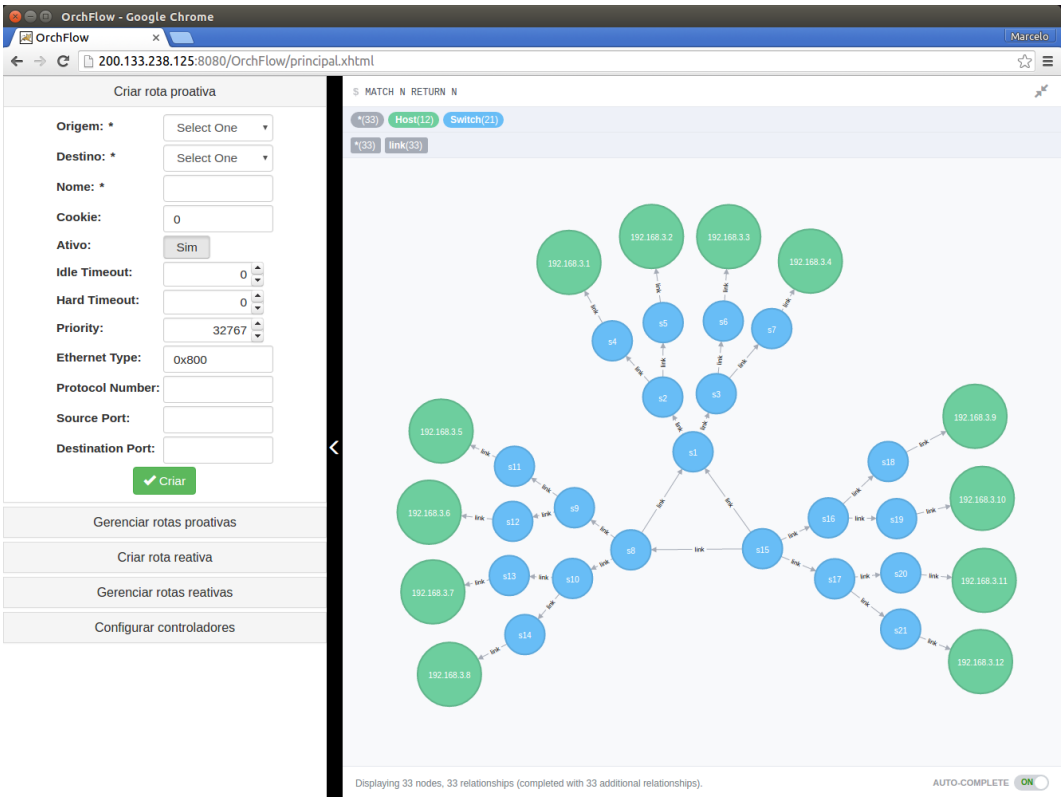

Figura 3. OrchFlow: Programação e Uso.

Do lado esquerdo da interface é possível identificar os dois modos de operação do OrchFlow no menu de opções: modos proativo e reativo. Para ambos os modos o procedimento para solicitar o serviço de roteamento fim a fim é o mesmo. O que diferencia um modo do outro é a forma pela qual serão criadas as regras. No modo proativo as regras são enviadas aos controladores que, por sua vez enviam imediatamente para todos os switches envolvidos no serviço solicitado. No modo reativo as regras são enviadas para cada controlador, porém ficam armazenadas em uma estrutura interna e serão enviadas aos switches apenas em eventos de *packet-in* ocorridos nos controladores.

O manual do usuário assim como o código fonte dos módulos necessários para Floodlight estão disponíveis em: `https://github.com/marcelofrate/OrchFlow/`

5. 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. Neste artigo, esperamos demonstrar que a integração entre controladores diferentes e a sua orquestração responda aos problemas de escala que atualmente existem nas redes SDN baseadas no protocolo OpenFlow.

O OrchFlow é uma ferramenta capaz de automatizar o processo de programação de múltiplos subdomínios, integrados a um único domínio administrativo, com uma visão

global, centralizada, possibilitando um funcionamento híbrido, nos modos proativo e reativo.

Os trabalhos futuros incluem o uso de diferentes controladores já que neste artigo apenas o controlador Floodlight foi usado. Além disso, testes de escalabilidade deverão ser realizados a fim de avaliar o potencial do OrchFlow em gerenciar domínios administrativos grandes, com centenas de elementos de rede e hosts.

Referências

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*.

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.