SIRC Simpósio de Informática
XIV Simpósio de Informática
7 e 8 de novembro de 2017
Conjunto I Rua dos Andradas, 1614
TRABALHO COMPLETO
ISSN 2175-0955
CENTRO UNIVERSITÁRIO FRANCISCANO

JMP: Uma Solução para Definição de Topologias Mininet Utilizando JSON

Brenda Salenave Santana, Guilherme de Freitas Gaiardo, Leonardo Marcuzzo, Luísa Perin Lucca, Thales Nicolai Tavares, Vinícius Fülber Garcia

Núcleo de Pesquisa de Tecnologias em Redes (NoSTRUn)
Universidade Federal de Santa Maria (UFSM)
Santa Maria – RS – Brasil

{bsantana,ggaiardo,lmarcuzzo,llucca,tntavares,vfulber}@inf.ufsm.br

Abstract. *The fast growth and popularization of computer networks created big and complex interconnected equipment groups. These groups are ilustrated by network topologies. As a way to guarantee good instantiations of network topologies and to test new network technologies, network emulations, such offered by Mininet platform, may be used. This paper presents JMP, a solution thats adds an abstraction layer to create Mininet topologies by using JSON, facilitating the Mininet emulator use for the prototyping environment creation.*

Resumo. *O rápido crescimento e popularização das redes de computadores criaram grandes e complexos grupos de equipamentos interconectados. Esses grupos são ilustrados através das topologias de rede. Como forma de garantir boas instanciações dessas topologias e de teste de novas tecnologias de rede, emulações de rede como as oferecidas pela plataforma Mininet podem ser utilizadas. Este trabalho apresenta JMP, uma solução que adiciona uma camada de abstração a criação de topologias Mininet através da utilização de JSON, facilitando assim o uso do emulador Mininet para a criação de ambientes de prototipação.*

1. Introdução

A popularização das redes de computadores tornam suas organizações topológicas cada vez maiores e mais complexas, além disso uma maior variedade de componentes, como funções de rede - *firewall*, NAT, *proxy* - tanto em plano de hardware quanto de software, são disponibilizados para garantir o funcionamento eficiente da rede.

Processos de simulação são essenciais para prever o comportamento e possíveis resultados da execução e/ou implantação de sistemas. Em especial, existem várias plataformas destinadas a simulação de diferentes tecnologias e topologias de rede [Team 2012] [Csoma et al. 2014] que possibilitam testes de novos protocolos, processamento de pacotes e gerenciamento de dispositivos.

O Mininet [Team 2012] é um emulador de redes que nativamente possui elementos como *hosts*, *switches* e *links*. Além disso conta com suporte a tecnologia de *Software Defined Networks* (SDN) [Hu et al. 2014] através da emulação de controladores e *OpenFlow switches* desacoplando a lógica de encaminhamento de pacotes contida em *switches* tradicionais e atribuindo a mesma para os elementos de controladores.

O Mininet implementa uma API em Python [Van Rossum and Drake 2003] para a definição de topologias de rede e utilização da plataforma. Dessa forma é necessário que o usuário apresente conhecimento prévio dessa linguagem de programação para realizar as simulações desejadas.

Estruturas padronizadas como *Java Script Object Notation* (JSON) [Crockford 2006] são amplamente conhecidas e de simples construção. Essas estruturas apresentam bons níveis de expressividade, suficientes para definir topologias Mininet através de seus componentes e das relações entre eles. Além disso, a utilização de JSON permite manter as definições topológicas independentes de linguagem, sendo interpretadas por diferentes programas que intermedeiam o arquivo genérico de definição e a execução de fato na plataforma de emulação.

Este trabalho apresenta JMP, uma solução capaz de interpretar arquivos JSON com definições de topologias de rede e executar a mesma em código Mininet, traduzir a definição para linguagem Python e visualizar graficamente a mesma. O *script* Python gerado pela ação de tradução pode ser executado em qualquer ambiente Mininet. Esta solução é dividida em quatro grandes módulos: validação, tradução, execução e visualização. Além disso, uma forma de descrição de topologias de rede em JSON foi definida.

A definição JSON compreendida pelo JMP consiste de dois grandes blocos de estruturação da rede emulada: de componentes e de conexões. O bloco referente aos componentes define informações sobre os elementos emulados, como *hosts* e *switches*, enquanto que o bloco de conexões apresenta como esses elementos se relacionam entre si.

O restante deste artigo está estruturado como se segue: a Seção 2 traz um *background* com os principais conceitos e ferramentas utilizados. Na Seção 3 a solução JMP é apresentada através da descrição modelo JSON e de seus quatro módulos operacionais. Finalmente, a Seção 4 apresenta a conclusão e os trabalhos futuros.

2. *Background*

O Mininet [Team 2012] é um sistema para emulação de redes de computadores. Mais precisamente, esse sistema é capaz de realizar a orquestração dos componentes da rede. Tais componentes podem ser *hosts*, *switches*, controladores SDN e enlaces executando em uma mesma máquina física utilizando virtualização baseada em processo. Através do Mininet é possível realizar experimentações de topologias de rede complexas em uma única máquina de forma simples.

Ao realizar a configuração de diferentes parâmetros dos componentes de uma topologia Mininet é possível simular o atraso e velocidade dos enlaces, filas de roteadores e *middleboxes*, bem como acessar via *SSH* os hosts e rodar programas genéricos como uma máquina real. A plataforma Mininet também habilita a tecnologia de SDN utilizando o protocolo OpenFlow, dessa forma fornecendo grande liberdade sobre o encaminhamento dos pacotes e proporcionando um ótimo ambiente para pesquisadores realizarem testes e simulações [De Oliveira et al. 2014] [Kaur et al. 2014].

A criação de topologias no Mininet é feita utilizando uma API em Python. Esta API possui três níveis diferentes de abstração: baixo nível, nível intermediário e alto nível.

O nível mais baixo de abstração é composto pelas classes básicas de nodos e enlaces (*e.g. Host*, *Switch*, *Link*) que podem ser instanciados individualmente para formar uma topologia.

O nível de abstração intermediário adiciona a classe Mininet que serve como um contêiner para nodos e enlaces. Esta classe possui métodos para adicionar os componentes na rede, bem como iniciar e parar sua execução. O nível mais alto adiciona a abstração de modelos de topologias através da classe *Topo* que permite que classes que a herdam criem modelos reutilizáveis de topologias.

Dado a facilidade proporcionada pelo Mininet na prototipação e experimentação com redes, diversos projetos foram feitos para estender as capacidades do Mininet. Neste contexto destaca-se a plataforma ESCAPE[Csoma et al. 2014], um *framework* que utiliza o Mininet para criar topologias com Virtualização de funções de rede programadas utilizando o *Click Modular Router* [Kohler et al. 2000].

Outro projeto a ser mencionado é o Mininet-WiFi [Fontes et al. 2015], uma extensão do Mininet que permite a criação de cenários *Wireless* SDN para simulação realista de topologias WiFi. Esta extensão adiciona pontos de acesso e estações *wireless* aos módulos do Mininet, possibilitando testes em topologias sem fio com uso da tecnologia SDN/OpenFlow.

Todo esse arcabouço de ferramentas faz surgir a necessidade de definições de topologias Mininet em notações abstraídas da própria linguagem utilizada na plataforma, simplificando o processo de descrição da topologia. Neste contexto o padrão JSON apresenta-se como uma solução apropriada para a definição de elementos Mininet e suas relações.

O *JavaScript Object Notation* (JSON) [Crockford 2006] é um padrão aberto de formatação textual de objetos de dados fácil para humanos lerem e máquinas utilizarem. Sua notação é abstraída, de fácil entendimento e com suporte para as principais linguagens de programação atuais. Além da fácil utilização, a performance do JSON chega a ser cem vezes melhor que notações similares como *Extensible Markup Language* (XML) [Nurseitov et al. 2009].

3. JMP

JMP[1] é uma solução para transcrição e execução de topologias Mininet construídas através de JSON que foi desenvolvida com o intuito de adicionar uma camada de abstração para a utilização da plataforma de emulação de rede. Por se tratar de um conjunto de classes e métodos, essas funcionalidades são facilmente portadas para aplicações terceiras, funcionando assim como uma API.

A solução foi desenvolvida em linguagem Python e utiliza um conjunto de bibliotecas externas que possibilitam a comunicação entre JMP e Mininet (Mininet API) e a visualização das topologias (PyGame API), além de diferentes bibliotecas disponibilizadas nativamente pela linguagem.

A solução é dividida em quatro módulos: validação, tradução, execução e visualização. O módulo de validação oferece todas as informações necessárias para os

[1]Código e exemplos: https://github.com/ViniGarcia/JMP

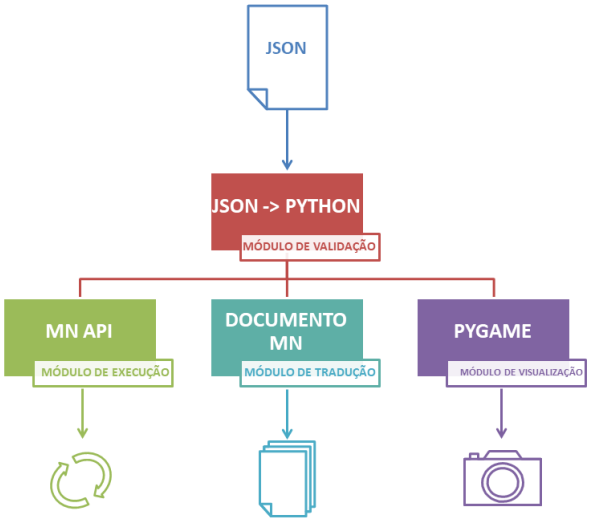

Figura 1. Relacionamento entre Módulos do JMP

outros três módulo, tendo assim sua execução como pré-requisito dos demais. Os demais módulos atuam de forma independente um do outro e geram diferentes saídas. A Figura 1 apresenta a relação entre os módulos da JMP.

3.1. Modelo de Definição Topológica

As topologias Mininet são definidas através de um modelo JSON que será interpretado pela JMP. A Figura 2 ilustra o modelo JSON com os nomes adequados para cada um de seus componentes, a sua hierarquia e organização e o tipo de cada um de seus elementos. O arquivo de configuração apresenta três blocos principais:

- *ID*: uma cadeia de caracteres que define um identificador único que nomeia a topologias, essa informação é utilizada tanto pelo módulo de tradução para nomear a função de instanciação gerada, quanto no módulo de visualização, para identificação da imagem gerada.
- *COMPONENTS*: um dicionário que apresenta todos os componentes Mininet que serão utilizados para montar a topologia. Os componentes Mininet suportados pela JMP são *Host*, *Switch*, *Controller* e *OVSwitch*.
- *CONNECTIONS*: apresenta a relação entre cada um dos componentes, essas relações são transformadas posteriormente em *links*. Um ponto inicial (*IN/OUT*) e um ponto final (*OUT/IN*) são necessários para definição das conexões.

3.2. Módulo de Validação

A validação é o processo inicial para possibilitar a execução de qualquer um dos demais módulos. Este processo consiste na análise do arquivo JSON fornecido ao módulo com o intuito de encontrar possíveis erros ou definições erradas. Além disso, busca-se determinar alguns tipos de definições ilegais, como uma conexão de um aparelho nele mesmo.

É importante ressaltar que este módulo busca validar o JSON de definição e apenas alguns aspectos topológicos de forma a possibilitar a tradução ou execução de código Mininet. Ou seja, possíveis erros relacionados aos componentes internos do Mininet serão verificados pela própria plataforma de emulação de redes. Exemplos de regras verificadas são:

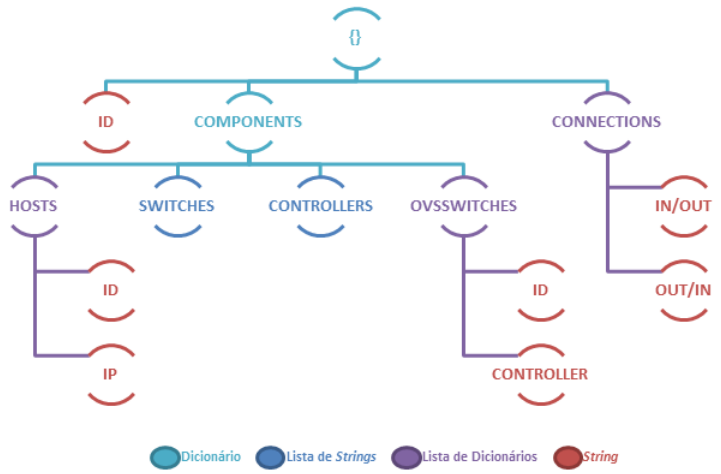

Figura 2. Modelo JSON para Definição Topológica

- Existência dos três blocos essências no dicionário principal (*ID*, *COMPONENTS* e *CONNECTIONS*).
- Tipos de dados definidos em cada um dos campos.
- IDs utilizados uma única vez na definição topológica.

A classe de validação, além de realizar o trabalho base de checagem, guarda as informações essenciais dos componentes em classes representativas dos mesmos. Essas classes, juntamente a lista de conexões, são utilizadas pelos módulos seguintes para tradução, instanciação ou visualização topológica.

3.3. Módulo de Tradução

O módulo de tradução recebe informações do módulo de validação e oferece duas formas para realizar a conversão do documento escrito com JSON para uma classe Python nativamente suportada pelo Mininet: a de baixo e a de alto nível. Ou seja, é possível traduzir o documento JSON utilizando as funções de baixo nível, onde o componente de geral de rede Mininet que predefine e abstrai diversas configurações (como atribuição de IPs) não é utilizado, ou de médio nível, onde ele é presente. A API Mininet de alto nível não foi utilizada pois não é adequada a traduções automáticas, visto que a mesma trabalha com topologias pré-definidas.

A utilização da API Mininet de baixo nível consiste no controle manual de toda a rede, isso inclui, por exemplo, processos para começar e terminar a execução de cada um dos componentes da rede. Esse tipo de escolha dará maior controle ao usuário porém exige maior expertise e planos claros do que será executado. Já a API de médio nível utiliza o componente de rede Mininet, adicionando uma camada de abstração no controle de componentes, além disso é possível executar uma *Command Line Interface* (CLI) nativa na plataforma de emulação de redes, possibilitando o controle dos componentes durante a execução dos mesmos.

De forma geral, o módulo de tradução possibilita que usuários sem conhecimento das funções de construção de topologias do Mininet possam criar as mesmas. Ao utilizar JSON retira-se o elemento de programação necessário sem ele, possibilitando a realização de definições através de um sistema puramente descritivo.

3.4. Módulo de Execução

O módulo de execução objetiva instanciar a topologia descrita em JSON e abrir um terminal de controle da mesma sem a necessidade do processo de tradução. Ou seja, em nenhum momento arquivos de definição topológica em Python são criados, ao invés disso, a solução cria uma rede Mininet, instancia e incia os elementos descritos e fornece uma CLI para o usuário realizar ações na plataforma de emulação de rede.

Para que esse processo seja possível, a API Mininet de médio nível foi utilizada. Dessa forma, apesar do elemento de rede abstrair boa parte das funções de controle de baixo nível, ainda se tem a referência para todos os componentes da rede, o que possibilita manipulações individuais. Essa API também permite a utilização da CLI Mininet nativa, passando o controle total da rede para o usuário final.

Após a instanciação da topologia, todo o controle interno e gerenciamento de erros é realizado pela plataforma Mininet. Ou seja, possíveis falhas de instanciação, problemas de virtualização e recuperações de sistema são de inteira responsabilidade do emulador.

Diferente do módulo de tradução, este módulo tem por objetivo possibilitar simulações de rede na plataforma Mininet abstraindo até mesmo o código que será executado. O resultado principal decorrente da existência do módulo de execução é a oportunidade de utilizar a plataforma de emulação de redes sem nenhum conhecimento da definição topológica da mesma, apenas descrevendo essas topologias em alto nível.

3.5. Módulo de Visualização

O módulo de visualização apresenta a topologia descrita em JSON de forma gráfica. Para isso a API PyGame foi utilizada com o intuito de criar a interface de visualização. Como nos demais módulos, as informações são retiradas da classe de saída do módulo de validação, garantindo uma topologia correta e a ilustração adequada da mesma.

A Figura 3 apresenta um exemplo de visualização de uma topologia. Os componentes são representados através de esferas e cada cor refere-se a um tipo de componente diferente: vermelho para *Host*, laranja para *Switch*, azul para *OVSwitch* e verde para *Controller*.

4. Conclusão

O rápido crescimento e popularização das redes resultam em topologias complexas e de grande tamanho. Processos de emulação ajudam a aumentar a confiança para que a implementação prática dessas topologias seja realizada. Além disso, ambientes emulados são necessários para a validação de tecnologias emergentes, verificando a aplicabilidade prática das mesmas.

O Mininet é um popular emulador de redes implementado em linguagem Python que nativamente possui suporte a redes SDN, dessa forma, compreende desde elementos tradicionais formadores das topologias, como *switches* tradicionais, até elementos dirigidos a tecnologia SDN, como controladores e *Open VSwitches*.

Este trabalho apresenta JMP, uma solução que cria uma camada de abstração através de JSON para a definições de topologias Mininet. Esta solução conta com quatro módulos principais: validação, tradução, execução e visualização. O módulo de validação

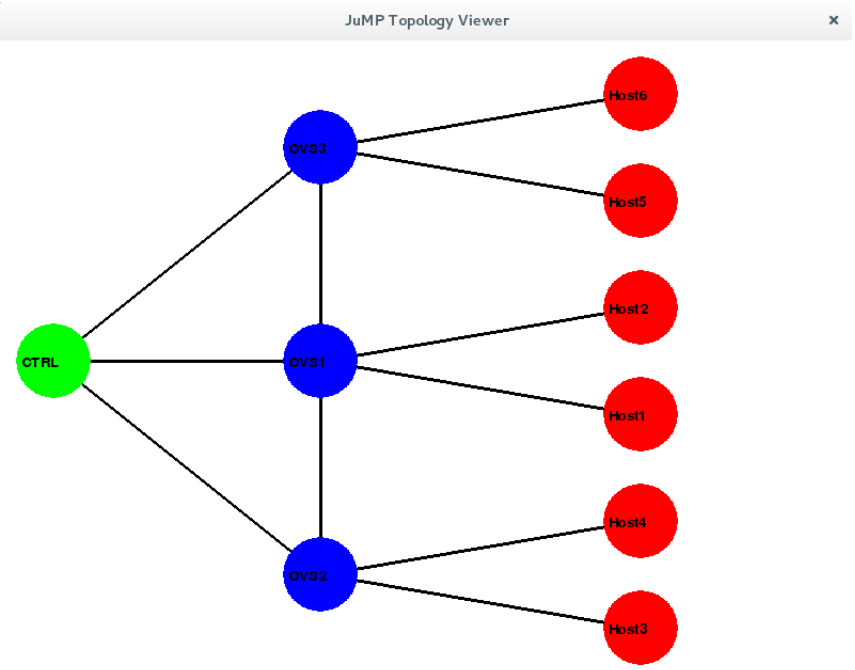

Figura 3. Exemplo de Visualização de Topologia de Rede

fornece todos os dados dos componentes Mininet, além das conexões entre os mesmos, para os demais módulos exercerem suas funções.

Em trabalhos futuros objetiva-se desenvolver uma interface gráfica para a construção das topologias Mininet. Dessa forma, o usuário poderá determinar componentes e relações de maneira gráfica traduzida posteriormente para JSON. O arquivo JSON gerado pode ser utilizado nos módulos já existentes da plataforma para execução ou tradução Mininet.

Referências

Crockford, D. (2006). The application/json media type for javascript object notation (json).

Csoma, A., Sonkoly, B., Csikor, L., Németh, F., Gulyas, A., Tavernier, W., and Sahhaf, S. (2014). Escape: Extensible service chain prototyping environment using mininet, click, netconf and pox. In *ACM SIGCOMM Computer Communication Review*, volume 44, pages 125–126. ACM.

De Oliveira, R. L. S., Shinoda, A. A., Schweitzer, C. M., and Prete, L. R. (2014). Using mininet for emulation and prototyping software-defined networks. In *Communications and Computing (COLCOM), 2014 IEEE Colombian Conference on*, pages 1–6. IEEE.

Fontes, R. R., Afzal, S., Brito, S. H., Santos, M. A., and Rothenberg, C. E. (2015). Mininet-wifi: Emulating software-defined wireless networks. In *Network and Service Management (CNSM), 2015 11th International Conference on*, pages 384–389. IEEE.

Hu, F., Hao, Q., and Bao, K. (2014). A survey on software-defined network and openflow: From concept to implementation. *IEEE Communications Surveys & Tutorials*, 16(4):2181–2206.

Kaur, K., Singh, J., and Ghumman, N. S. (2014). Mininet as software defined networking testing platform. In *International Conference on Communication, Computing & Systems (ICCCS)*, pages 139–42.

Kohler, E., Morris, R., Chen, B., Jannotti, J., and Kaashoek, M. F. (2000). The click modular router. *ACM Transactions on Computer Systems (TOCS)*, 18(3):263–297.

Nurseitov, N., Paulson, M., Reynolds, R., and Izurieta, C. (2009). Comparison of json and xml data interchange formats: a case study. *Caine*, 2009:157–162.

Team, M. (2012). Mininet: An instant virtual network on your laptop (or other pc).

Van Rossum, G. and Drake, F. L. (2003). *Python language reference manual*. Network Theory.