Uma Ferramenta para Monitoração de Fluxos em Redes Definidas por Software

David Anderson Bezerra Silva, Ramon Rodrigues Moreira e Helcio Wagner da Silva

Centro de Ciências Exatas e Naturais
Universidade Federal Rural do Semi-Árido (UFERSA)
Caixa Postal 59.625-190 – Mossoró – RN – Brasil

{david.anderson, helcio}@ufersa.edu.br,ramonrodriguesmo@gmail.com

Abstract. *This article describes an application for flow monitoring in Software Defined Netwoking. In this context, two REST APIs have been developed: one of them is offered by the controller to allow the application to install flow monitors to specific application-layer protocols; the other is offered by the application to allow the controller to indicate the beginning and the ending of a monitored flow. A strategy was also developed to allow the controller to receive packets related to the monitored flows, for further investigation.*

Resumo. *Este artigo descreve uma ferramenta para monitoração de fluxos em Redes Definidas por Software. Neste contexto, foram desenvolvidas duas APIs REST: uma delas é disponibilizada pelo controlador para que a ferramenta instale monitores de fluxo relacionados a determinados protocolos do nível de aplicação; a outra é disponibilizada pela ferramenta para que o controlador possa sinalizar o início e o fim de um fluxo monitorado. Uma estratégia também foi desenvolvida para que o controlador passasse a receber pacotes que correspondam os fluxos monitorados, para investigação posterior.*

1. Introdução

Redes Definidas por Software (SDN – *Software Defined Networking*) é uma nova estratégia desenvolvida para a operação e gerência de redes de computadores. Ela é baseada no desacoplamento espacial dos planos de controle e de dados em equipamentos de comutação, conforme ilustrado pela Figura 1. O plano de controle é representado por um artefato de software denominado controlador que se comunica com os switches para preencher suas respectivas tabelas de fluxo de acordo com políticas de gerência estabelecidas por aplicações localizadas no plano de aplicações.

Há várias opções para se estabelecer a comunicação entre o controlador e os switches – que é genericamente denominada de API *Southbound*. O representante mais popular daquela API é o protocolo OpenFlow [Open Networking Foundation, 2012]. Similarmente, há também várias formas de estabelecer a comunicação entre as aplicações de gerência e o controlador – que é genericamente denominada de API *Northbound*. Neste caso, porém, não há consenso a respeito de qual é o representante mais popular daquela API. É possível, por exemplo, desenvolver uma API especializada baseada na estratégia REST, a ser abordada na próxima seção.

ISBN 978-85-7669-419-9

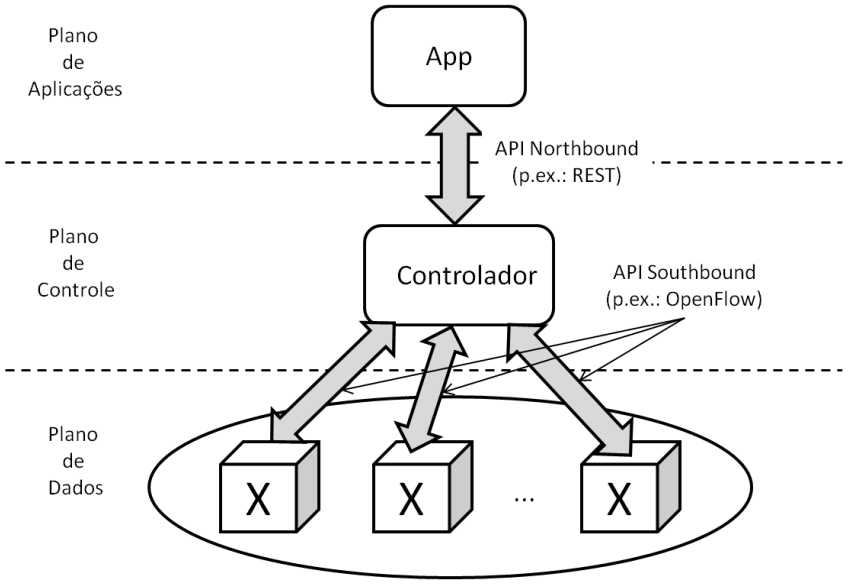

Figura 1: Separação dos planos de controle e de dados na arquitetura SDN

A adoção do paradigma SDN promete diminuir os custos de equipamentos de comutação, haja vista que eles necessitariam apenas comportar a API *Southbound*. Além disso, a separação entre os planos de controle e de dados permite a adoção de políticas de gerência que são impossíveis de serem adotadas hoje com os protocolos de roteamento existentes.

2. REST

REST (*Representational State Transfer*) é um estilo arquitetônico definido em [Fielding, R., 2000] para o projeto de sistemas distribuídos. Ele consiste na verdade em um conjunto de requisitos. Dentre eles está a adoção do paradigma cliente-servidor em que servidores mantêm recursos identificados por URIs (*Uniform Resource Identifiers*). Conforme ilustrado pela Figura 2, a especificação de uma API REST na prática baseia-se na utilização de URLs (*Uniform Resource Locators*, um tipo específico de URI) para definição de recursos e do protocolo HTTP [Fielding, R. *et al*, 1999] para realização de operações CRUD (*Creation, Read, Update, Delete*) naqueles recursos em um servidor.

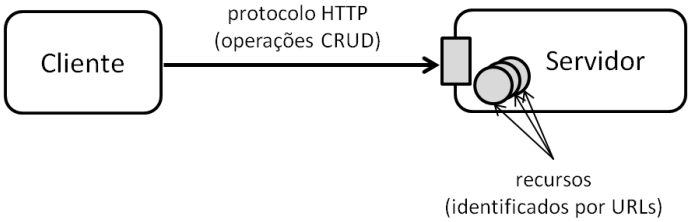

Figura 2: aspectos práticos de uma API REST

Outra restrição definida para sistemas REST é a natureza *stateless* nas interações entre clientes e servidores. Na prática, isto significa que a solicitação HTTP enviada por um cliente deve conter todas as informações de estado necessárias para a criação, leitura, atualização ou remoção de um recurso em um servidor. Na prática, as informações de estado estão contidas no corpo de mensagens HTTP e expressas em

JSON (*JavaScript Object Notation*) [Bray, T., 2014] ou XML [Bray, T. *et al*, 2008] – sugere-se o primeiro em detrimento do segundo pela sua simplicidade.

Há várias fontes que fornecem linhas mestre para a construção de APIs REST, tais como [Allamaraju, S., 2010], [Webber, J. et al, 2010] e [Massé, M., 2012]. Unânimes em todas elas estão as recomendações de que recursos sejam nomeados de forma que façam sentido para o consumidor da API (os clientes), e que uma ação a ser realizada com base em um recurso seja definida pelo método HTTP utilizado na solicitação correspondente. Neste contexto, recomenda-se a utilização do método POST para a criação de um recurso, a utilização do método GET para obtenção das informações de estado daquele recurso, a utilização do método PUT para atualização daquelas informações de estado e a utilização do método DELETE para a remoção do recurso. Redes sociais populares, tais como Twitter, Facebook e LinkedIn, oferecem APIs REST para que desenvolvedores possam criar aplicações que as utilizem. A própria Google disponibiliza sua API REST para acesso aos seus serviços.

3. Visão Geral da Arquitetura

A ferramenta de monitoração de fluxos é ilustrada pela Figura 3. Seus principais componentes são uma aplicação e um controlador personalizados. A aplicação disponibiliza ao usuário uma interface gráfica que lhe permite inicializar e encerrar a monitoração de fluxos. O controlador, além da sua atribuição normal de preencher as tabelas de fluxos dos switches para que eles realizem a comutação de pacotes corretamente, é capaz de alterar aquelas tabelas para que os switches realizem também a monitoração de fluxos que são solicitados pela aplicação. As APIs REST disponibilizadas pelo controlador e pela aplicação, bem como os procedimentos executados pelo controlador para realizar a monitoração de fluxo, são descritos pelas seções seguintes.

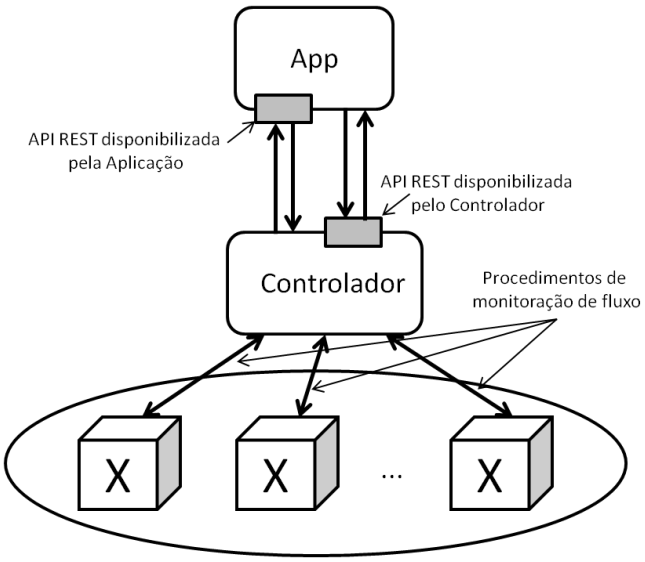

Figura 3: Visão geral da arquitetura

4. A API REST disponibilizada pelo Controlador

Um monitor de fluxo é modelado como um recurso armazenado pelo controlador. Neste contexto, a Figura 4 ilustra a API disponibilizada pelo controlador para a criação (Figura 4a) e a remoção (Figura 4b) do monitor. O parâmetro *protocol* especifica o protocolo de aplicação cujo fluxo se deseja monitorar.

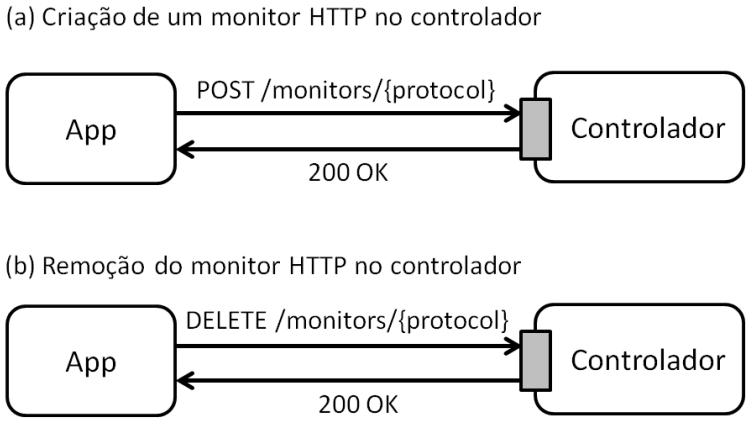

Figura 4: API disponibilizada pelo controlador

Para um determinado protocolo, há apenas um monitor de fluxo mantido pelo controlador. Assim, a tentativa de criação de um monitor de fluxo quando ele já existe deve retornar uma mensagem de erro.

5. A API REST disponibilizada pela Aplicação

Cada fluxo monitorado é modelado como um recurso na aplicação. Neste contexto, a Figura 5 ilustra a API disponibilizada pela aplicação. Uma vez que um fluxo associado a um protocolo de aplicação é percebido pelo seu respectivo monitor, o controlador envia uma solicitação HTTP. Utilizando o método POST, a solicitação ilustrada pela Figura 5a caracteriza o fluxo em termos dos endereços IP e portas de origem e destino, além de um identificador único para cada fluxo, representados em JSON pelos parâmetros *ip_src*, *port_src*, *ip_dst*, *port_dst* e *id*, respectivamente.

Na Figura 5b, a remoção do recurso que representa um fluxo na aplicação é feito utilizando uma solicitação HTTP possuindo o método DELETE e contendo os parâmetros que caracterizam o fluxo expressos em JSON. A Figura 5c ilustra também a inclusão de detalhes relativos aos fluxos monitorados através de uma solicitação HTTP utilizando o método POST. A solicitação caracteriza o fluxo através de seu identificador, e inclui neste caso também um espaço destinado ao armazenamento de informações úteis (expressas em JSON) para a análise do fluxo, como URL para fluxos HTTP, endereços de e-mail de origem e destinatário para mensagens enviadas através do protocolo SMTP, além de nome de sentido de movimentação (envio ou recebimento) de arquivo para operações FTP.

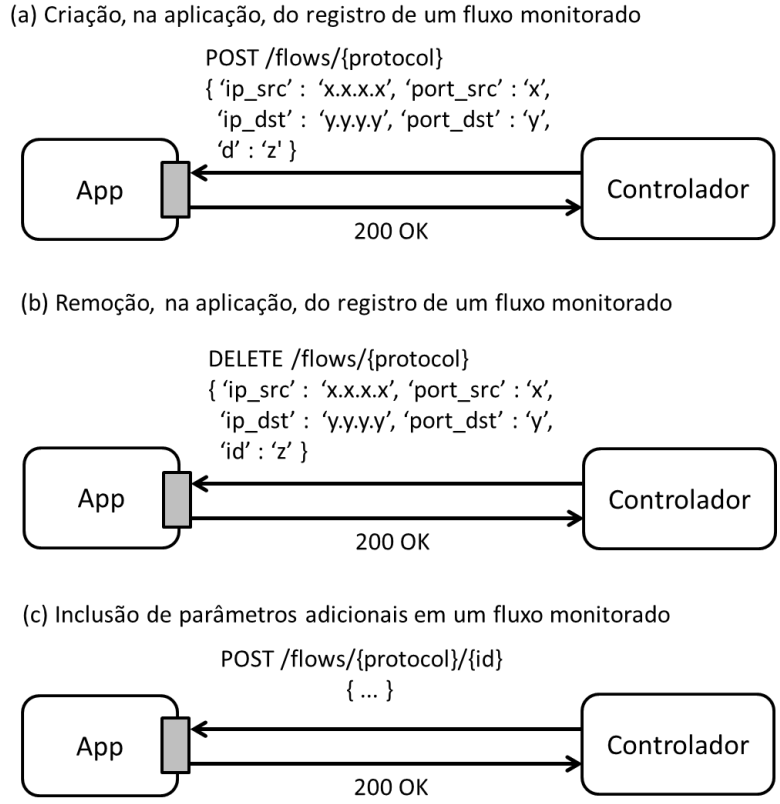

Figura 5: API REST disponibilizada pela aplicação

6. Procedimentos para Monitoração de Fluxo

Para monitoração do fluxo, cada switch manipula duas tabelas de fluxo. Ambas são ilustradas pela Figura 6. A Tabela 0 contém inicialmente a entrada padrão e_3, que não possui nenhum campo para *match* de pacotes e cuja instrução aponta para o exame da Tabela 1. Trata-se de uma entrada do tipo *Table Miss* – ou seja, ela é examinada apenas quando não ocorre nenhum *match* nas demais entradas da Tabela. Quando o controlador é instruído pela aplicação a iniciar a monitoração de fluxos relativos a um determinado protocolo de aplicação, novas entradas são adicionadas à Tabela 0. Elas possuem campos ajustados para *match* de pacotes que correspondam àqueles fluxos – isto é, datagramas que encapsulam segmentos TCP ou UDP cujas portas de origem e destino possuam valores específicos. A Figura 6 exemplifica a forma destas entradas (representadas por e_1 e e_2) para monitoração de fluxos relacionados ao protocolo HTTP.

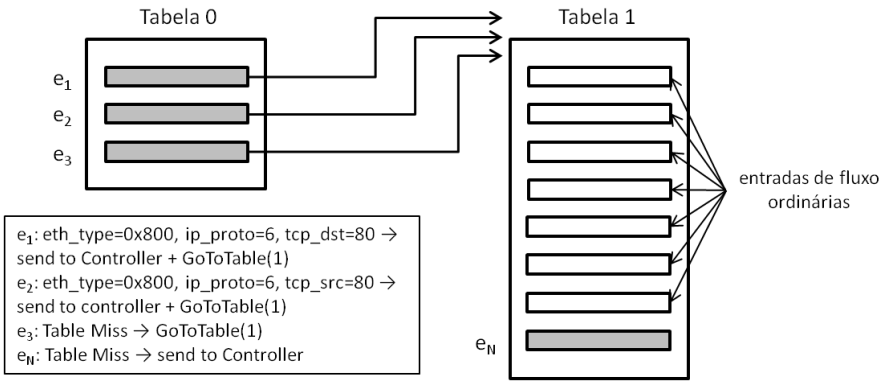

Figura 6: manipulação das tabelas de fluxo para monitoração de fluxos HTTP

As entradas recém-inseridas possuem um conjunto de duas instruções, que são executadas quando ocorre um *match*: a primeira delas corresponde ao envio do pacote para o controlador; a segunda instrução corresponde ao direcionamento do processamento para a Tabela 1, onde são armazenadas as entradas de fluxo ordinárias que fazem com que o switch se comporte como um switch convencional. Neste contexto, a entrada e_N representa uma entrada *Table Miss* a partir da qual um pacote que não obteve nenhum *match* nas entradas anteriores é encaminhado para o controlador.

7. Aspectos de Implementação

Dentre os vários controladores disponíveis, o controlador escolhido para a ferramenta foi o Ryu [Ryu SDN Framework Community, 2017]. Trata-se de um controlador inteiramente gratuito, que comporta as mais recentes versões do OpenFlow e que possui uma comunidade ativa de desenvolvedores e usuários. O Ryu é executado com base em um ou mais módulos Python fornecidos em tempo de execução.

O código-fonte do Ryu traz consigo vários módulos. Um destes módulos faz com que os switches gerenciados aprendam a encaminhar quadros da mesma forma que switches ordinários. Sua lógica inclui a instalação de entradas de fluxo ordinárias na Tabela 0. Para o desenvolvimento da ferramenta, foi utilizada uma pequena variação deste módulo na qual as entradas de fluxo passam a ser adicionadas na Tabela 1, conforme ilustrado pela Figura 6.

Nesta primeira versão da ferramenta, um módulo foi desenvolvido especificamente para monitorar fluxos relativos aos protocolos HTTP, FTP e SMTP em um ambiente emulado pela ferramenta mininet [Mininet Team, 2017]. Em particular, todos estes protocolos utilizam os serviços prestados pelo TCP, e o início do fluxo é detectado pela presença de segmentos com as *flags* SYN e ACK levantadas simultaneamente – o que ocorre durante o processo de abertura de conexão TCP, conhecido como *three-way handshake*. A partir daí, os parâmetros adicionais, que variam para cada protocolo, são monitorados. O final do fluxo é detectado pela presença de segmentos com as *flags* FIN e ACK levantadas simultaneamente, o que ocorre durante o processo de encerramento de conexão TCP.

Para teste de monitoração de fluxo HTTP, foi utilizado a implementação nativa de um servidor HTTP escrito em Python denominada *SimpleHTTPServer*, e o aplicativo *wget* como cliente. Para teste de monitoração de fluxo FTP, foi utilizado o *inetd*

(*internet service daemon*), um superservidor capaz de emular serviços FTP; o cliente utilizado para este protocolo foi o aplicativo homônimo, *ftp*. Por fim, para teste de monitoramento de fluxo SMTP, foi utilizado a implementação nativa de um servidor SMTP escrito em Python, denominada *DebuggingServer*; o aplicativo *telnet* foi utilizado como cliente SMTP através da abertura de conexão TCP com a porta 25 do servidor supracitado.

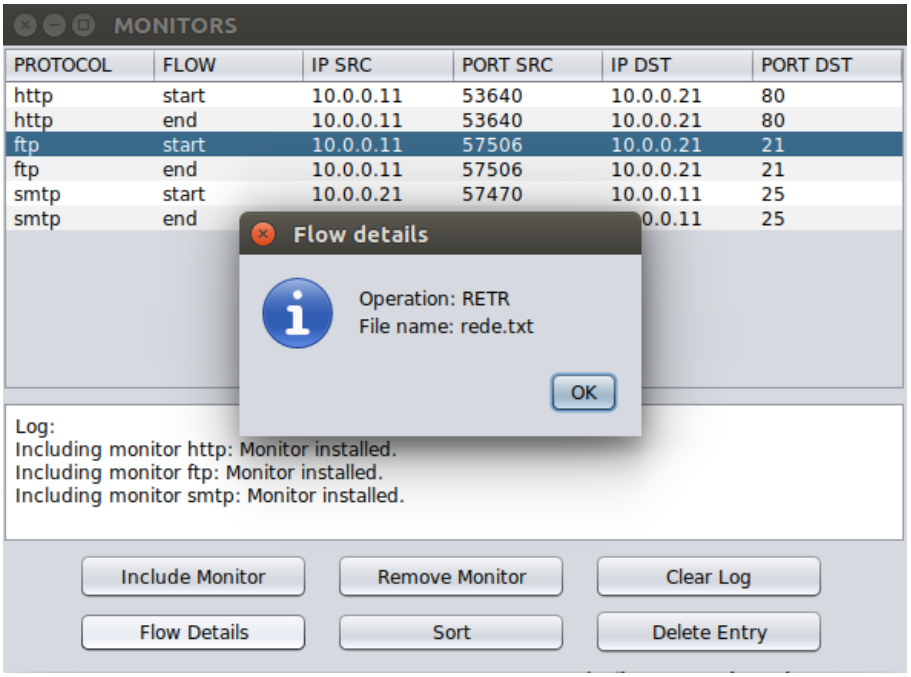

Figura 7: Captura de tela da ferramenta desenvolvida

A Figura 7 ilustra uma captura de tela da aplicação desenvolvida para comunicação com o controlador. Ela inclui recursos para instalação e remoção de monitores de fluxo. Na figura, uma lista sinalizando o inicio e fim de três fluxos pode ser visualizada – pelo exame das portas utilizadas, percebe-se facilmente que se trata de fluxos HTTP, FTP e SMTP, respectivamente. Além disso, a aplicação permite a visualização de parâmetros adicionais na seleção de um determinado fluxo. Por exemplo, na figura, é possível visualizar que um arquivo denominado *rede.txt* foi descarregado durante a sessão FTP monitorada.

8. Conclusões

Este artigo descreveu uma ferramenta desenvolvida para a monitoração de fluxos em um ambiente de Redes Definidas por Software. Este desenvolvimento incluiu, além da lógica de monitoração em si, o projeto de uma API REST que é utilizada para comunicação entre um controlador SDN e uma aplicação executada em um nível de abstração conceitual mais elevado. Nesta versão inicial, a ferramenta é capaz de monitorar fluxos relativos a apenas três protocolos de aplicação. Nas versões posteriores, pretende-se ampliar este conjunto – incluindo-se nele protocolos baseados em UDP. Outra melhoria é a sinalização assíncrona dos parâmetros adicionais relativos a cada fluxo – na versão atual, estes parâmetros só podem ser visualizados mediante a

seleção prévia do fluxo. Em um futuro próximo, estes parâmetros serão informados na medida em que eles são transmitidos através da rede.

9. Referências

Open Networking Foundation (2012) "OpenFlow Switch Specification – version 1.3.0 (Wire Protocol 0x04)". Disponível a partir de http://www.opennetworking.org.

Fielding, R. (2000) "Architectural Styles and the Design of Network-based Software Architectures", Tese de Doutorado. Universidade da Califórnia, Irvine.

Fielding, R. *et al* (1999) "HyperText Transfer Protocol – HTTP/1.1". RFC 2616. Disponível em https://www.ietf.org/rfc/rfc2616.txt.

Bray, T. (2014) "The JavaScript Object Notation (JSON) Data Interchanged Format", RFC 7159. Disponível em https://www.ietf.org/rfc/rfc7159.txt.

Bray, T. *et al* (2008) "Extensible Markup Language (XML) 1.0 (Fifth Edition)", World Wide Web Consortium Recommendation REC-xml-20081126". Disponível em https://www.w3.org/TR/2008/REC-xml-20081126/.

Allamaraju, S. (2010) "RESTful Web Services Cookbook". O'Reilly Media.

Webber, J. *et al* (2010) "REST in Practice". O'Reilly Media.

Massé, M. (2012) "REST API – Design Rulebook". O'Reilly Media.

Ryu SDN Framework Community (2017). "Component-Based Software Defined Networking Framework: Build Agilely". Disponível em https://osrg.github.io/ryu/.

Mininet Team (2017) "An Instant Virtual Network on your Laptop (or other PC)". Disponível em http://mininet.org/.