Uma Solução para Potencializar a Adaptação e Gerenciamento de Funcionalidades de Infraestruturas Baseadas em SDN

Emídio P. Neto[1], **Felipe S. Dantas Silva**[1,2], **Marcilio O. O. Lemos**[1,2]
Augusto Venâncio Neto[2,3]

[1]LaTARC Research Lab
Instituto Federal de Educação, Ciência e Tecnologia do Rio Grande do Norte (IFRN)
Natal-RN, Brasil

[2]Departamento de Informática e Matemática Aplicada (DIMAp)
Universidade Federal do Rio Grande do Norte (UFRN) – Natal-RN, Brasil

[3]Instituto de Telecomunicações (IT), Aveiro, Portugal

emidio.neto@ifrn.edu.br, felipe.dantas@ifrn.edu.br
marcilio.cc.lemos@gmail.com, augusto@dimap.ufrn.br

Abstract. *The lack of standardization of the SDN Controller APIs requires high-level customization of the applications built under the control plane for the APIs specific to each adopted controller. This results in poor utilization of network features and functionality as well as reworking for infrastructure operators. In this context, this work proposes ATOM, an architecture for managing SDN controllers that allows network management operations to be performed by an external application that enables the management of network resources dynamically and ubiquitously. The proposal was validated through implementations to manage the Ryu controller and the preliminary results, considering specific tasks of the creation of flows, revealed a reduction in processing time for related operations.*

Resumo. *A ausência de padronização das APIs dos controladores SDN exige um alto nível de customização das aplicações contruídas sob o plano de controle para as APIs específicas de cada controlador adotado. Isso resulta em mau aproveitamento de recursos e funcionalidades da rede bem como retrabalho aos operadores da infraestrutura. Neste contexto, este trabalho propõe o ATOM, uma arquitetura para gerenciamento de controladores SDN que permite que as operações de gerenciamento da rede possam ser realizadas por uma aplicação externa que viabiliza o gerenciamento dos recursos de rede de forma dinâmica e ubíqua. A proposta foi validada através de implementações para gerenciamento do controlador Ryu e os resultados preliminares, considerando tarefas específicas de criação de fluxos, relevaram uma redução no tempo de processamento para as operações relacionadas.*

1. Introdução

Embora o paradigma das Redes Definidas por Software (SDN, do inglês *Software-Defined Networking*) [Haleplidis et al. 2015] proporcione inúmeras facilidades e flexibilidade para adaptação de novos serviços e aplicações, vários aspectos relacionados

ao gerenciamento da infraestrutura permanecem como grandes desafios, principalmente tratando-se da integração de diferentes mecanismos [Sezer et al. 2013]. Isto aplica-se, particularmente, à falta de sistemas de controle que possam suportar os administradores de rede quando confrontados com a crescente complexidade da infraestrutura e com a vasta gama de requisitos de qualidade de experiência dos usuários [Sharma et al. 2013].

Outro ponto, normalmente relacionado ao gerenciamento de infraestruturas SDN, é a necessidade da visualização da topologia e consequentemente interação com os dispositivos. Neste caso, os operadores de rede que não têm muito conhecimento no desenvolvimento de ferramentas e no manuseio de determinadas Interfaces de Programação de Aplicação (APIs, do inglês *Application Program Interface*) providas pelos controladores SDN irão encontrar dificuldades para lidar com a alta granularidade de informações que constituem o ecossistema SDN. Como resultado desta alta complexidade do gerenciamento das informações, os controladores SDN atualmente disponíveis exigem que o operador da rede tenha um profundo conhecimento relacionado as suas APIs e principais métodos de extensão de funcionalidades. Tal exigência pode acarretar em diversos problemas, tais como o mau aproveitamento das funcionalidades de controle, aumento da complexidade de gerenciamento e menor flexibilidade para adaptação de novas aplicações. Esta limitação se dá pelo motivo que cada controlador é implementado de maneira distinta, sem a preocupação com questões de padronização e interoperabilidade.

Em se tratando da orquestração de serviços de rede, a obtenção de informações da infraestrutura em tempo real é um fator crítico para qualquer cenário [Guck et al. 2017]. No entanto, a grande maioria das APIs de serviços disponibilizados pelos controladores não dispõe informações de *status* da topologia em tempo real, exigindo do desenvolvedor a implementação de métodos que atuem como ouvintes de eventos (do inglês *event listener*), que irão receber informações em tempo real dos eventos da rede e disponibilizá-las através de uma determinada aplicação. Tais informações serão posteriormente utilizadas como fonte de dados em tempo real. A ausência de mecanismos capazes de lidar com tais limitações implica uma série de problemas referentes ao monitoramento de serviços de rede, restringindo o acompanhamento em tempo real de performance de rede, escalabilidade e utilização de recursos.

Mecanismos relacionados a visualização e interação com a topologia de rede são funcionalidades fundamentais de uma ferramenta de orquestração SDN, cujo propósito é auxiliar o gerenciamento da topologia e reduzir a complexidade das operações realizadas em diversos tipos de dispositivos [Gomes et al. 2016]. Desta forma, uma solução viável para sanar as limitações mencionadas anteriormente é a implementação de interfaces de gerenciamento que, através das *Northbound Interfaces* dos controladores SDN, permitem a interação entre o operador de rede e a topologia em execução, além de disponibilizar interfaces padronizadas para desenvolvimento de inovadoras funções de rede.

Por meio de um levantamento recente, conduzido para estudo do estado da arte referente a orquestração de infraestruturas SDN, foi identificado que ambos a comunidade científica e industrial têm se dedicado a desenvolver novas soluções concentradas no controlador em uso. Neste contexto, as ferramentas são orientadas em cada mecanismo de orquestração, o que confirma as limitações descritas anteriormente e, consequentemente, podem restringir a atuação destas soluções em cenários mais complexos.

Tais limitações são as principais motivações deste estudo, uma vez que o processo de orquestração de infraestruturas SDN requer um conjunto de funcionalidades integradas que possam fornecer o suporte adequado para as atividades de gerência da rede. Diante disso, este trabalho supre esta lacuna ao propor o ATOM (A*n **A**rchitecture for Ubiqui**TO**us **M**anagement of SDN Controllers*), uma arquitetura modular baseada em componentes que, por meio de interfaces de comunicação em alto nível, é capaz de integrar múltiplas funcionalidades de gerenciamento de rede, permitindo que operações não triviais sejam executadas de forma prática e em tempo de execução. Através de uma interface única de gerenciamento, o ATOM viabiliza a gerência de estado dos dispositivos da rede e disponibiliza mecanismos para gerir os recursos da rede em diversos controladores SDN de forma ubíqua.

Com o intuito inicial de fornecer uma prova de conceito (PoC, do inglês *Proof of Concept*) da proposta da arquitetura ATOM, as implementações concentraram-se no controlador Ryu[1], amplamente utilizado no meio acadêmico-científico. As validações foram conduzidas em simulações (por meio da virtualização das infraestruturas no emulador Mininet[2]) e experimentações através de um *testbed* SDN real. O PoC foi validado por meio da implementação de dois módulos de serviço principais do ATOM: *(i)* Controlador de Topologia (*Topology Manager*), que inclui funcionalidades de gerência de fluxos e recuperação de falhas (*failover*); e *(ii)* Controlador de QoS (*QoS Manager*), que inclui funcionalidades de criação de políticas de QoS e configuração de filas por classes de tráfego.

O restante deste trabalho está organizado da seguinte maneira: a seção 2 aponta os trabalhos relacionados ao tema de pesquisa descrito, de modo a delinear as contribuições desta proposta. A seção 3 apresenta a proposta da arquitetura ATOM, expondo seus principais componentes e funcionalidades. A seção 4 descreve em detalhes uma das principais operações do ATOM e apresenta resultados da avaliação no que diz respeito as operações de inserção de fluxos. Por fim, a seção 5 apresenta as considerações finais e delineia as perspectivas de trabalhos futuros.

2. Trabalhos Relacionados

A literatura revela que várias das soluções de orquestração de controladores SDN existentes têm o escopo voltado apenas para o provimento de meios de visualização gráfica dos dispositivos de rede. Em algumas propostas as ferramentas exibem a forma como estão organizados os dispositivos presentes na topologia mas não são capazes de permitir a interação do operador com os elementos de rede por meio da interface de gerenciamento, como é o caso das propostas apresentadas em [Huang et al. 2014] e [Salsano et al. 2015]. Em [Schultz et al. 2015] os autores propõem o OpenGUFI, um sistema que disponibiliza uma interface gráfica (GUI, do inglês *Graphical User Interface*) para abstração de topologia de rede: a ferramenta permite que o operador visualize as conexões entre clientes móveis e pontos de acesso sem fio, com suas respectivas regras de fluxo, mas sem viabilizar meios de interação ao operador. Desta maneira, a proposta não aborda a problemática relacionada a complexidade de gerenciamento da topologia, que ainda tem que ser operada manualmente pelo administrador.

[1]https://osrg.github.io/ryu/

[2]http://mininet.org/

Em outro grupo de trabalhos anteriores as propostas foram projetadas por meio da integração com ferramentas disponibilizadas por terceiros. Essas soluções, na maioria das vezes, estão disponíveis através de uma licença de uso restrito, como é o caso do trabalho proposto em [Shalimov et al. 2015], no qual os autores propõem o controlador RUNOS. Tal proposta faz uso dos mecanismos disponibilizados pelo software de álgebra computacional *Maple* [Voellmy et al. 2013] para prover otimização às aplicações e adota o conceito de *multi-threading* efetivo para gerenciamento de processos como meio de evitar sobrecarga do sistema. No entanto, ao exigir que o operador de rede (que deseje fazer uso da solução) tenha um conhecimento especializado dos métodos restritos a determinadas APIs do próprio controlador, ele limita seu escopo no que se refere ao gerenciamento flexível e ubíquo.

Outra abordagem consiste no desenvolvimento de soluções de orquestração integradas em controladores específicos, que precisam ser adaptados ao contexto da solução de orquestração. Isso está sujeito a restrições graves de gerenciamento de rede, uma vez que aplicações que foram desenvolvidas para um controlador específico só podem ser usadas por ele, impedindo a interoperabilidade entre os controladores. Algumas soluções baseadas nesta perspectiva são descritas abaixo.

A proposta em [Koponen et al. 2010] apresenta o sistema Onix, que propõe a estruturação em *cluster* de um ou mais controladores NOX [Gude et al. 2008]. Tal abordagem, faz uso de uma Base de Informações de Rede (NIB, do inglês *Network Information Base*) em conjunto com o Apache ZooKeeper [The Apache Software Foundation 2016] e utiliza um protocolo proprietário para comunicação horizontal entre os controladores que formam o *cluster*. Além disso, a implementação principal não está disponível como um software de código aberto, o que certamente desencoraja a integração e o desenvolvimento de novos serviços e aplicações baseados nesse sistema.

Em [Choi et al. 2015] os autores apresentam o IRIS-CoMan como uma solução para o gerenciamento de infraestrutura SDN em larga-escala. A proposta apresentada é baseada no controlador Floodlight e requer a migração de todas as aplicações já desenvolvidas para manter conformidade com a API disponibilizada.

Os autores em [Hassas Yeganeh and Ganjali 2012] introduzem o Kandoo, que propõe uma estruturação para os controladores baseada em uma arquitetura hierárquica e tem como objetivo fornecer um controlador principal na parte superior da rede. Desta maneira espera-se viabilizar a interconexão de vários outros controladores. Esta abordagem é restrita ao controlador Kandoo e faz uso de um protocolo proprietário para comunicação entre as entidades controladoras, sendo inadequada para infraestruturas que fazem uso de outros controladores SDN.

DISCO [Phemius et al. 2013] propõe compartilhar o *status* das conexões de rede, mantendo a comunicação entre todos os controladores vizinhos. Além disso, esta abordagem requer que o código fonte das aplicações sejam atualizados no controlador Floodlight, o que torna a proposta completamente dependente deste controlador.

O HyperFlow [Ganjali and Tootoonchian 2010] está disponível como uma aplicação do controlador NOX [Gude et al. 2008] e requer a atualização de várias linhas de código no controlador, de modo que seja possível interceptar comandos e serializar

eventos. Por esta razão, esta abordagem revelou-se como sendo uma solução que está disponível apenas para sanar as adversidades de um controlador específico (o NOX).

Os autores em [Frate et al. 2018] propuseram uma ferramenta para gerenciamento de controladores OpenFlow que utiliza-se das REST APIs dos controladores Ryu e Floodlight para obtenção de informações de eventos da rede. Para tal, faz uso de um banco de dados orientado a grafos (Neo4J[3]) para armazenamento dos dados da topologia. Em consequência, a proposta torna-se totalmente dependente das funcionalidades providas pelo Neo4J para disponibilização da interface de gerenciamento, o que dificulta a implementação de recursos adicionais que possam ser providos pela interface de gerenciamento.

Embora as propostas referenciadas nesta seção apresentem seus respectivos benefícios, elas não se preocupam com a possível convergência entre diversos tipos de controladores SDN atualmente disponíveis. Algumas das propostas disponibilizam apenas funcionalidades de visualização gráfica da topologia em execução e são restritas a um controlador específico, não apresentando em suas arquiteturas métodos ou APIs que possibilitem escalabilidade relacionada ao número de controladores gerenciados. Outro grupo de propostas apresentam dependência de softwares que necessitam de licença de uso e dependência de protocolos proprietários, desencorajando seu uso em ambientes de larga escala.

A análise da literatura relacionada revelou que as soluções propostas não fornecem funcionalidades mínimas de orquestração ao operador de rede no que diz respeito ao gerenciamento ubíquo da infraestrutura. Tarefas fundamentais como gerenciamento de fluxos e de políticas de QoS devem ser disponibilizadas no mais alto nível de abstração. Diante disso, espera-se que as próximas gerações de soluções de orquestração sejam capazes de prover, no mínimo: *(i)* visualização em tempo real da topologia de rede; *(ii)* mecanismos de interação com a topologia em execução; e *(iii)* independência de softwares proprietários. A Tabela 1 apresenta uma comparação dos trabalhos anteriores discutidos nesta seção em relação com a proposta apresentada neste trabalho (sistema ATOM), levando em consideração os aspectos básicos de funcionalidades mínimas esperadas de uma solução de orquestração de controladores SDN.

A relação abaixo apresenta uma descrição de cada item utilizado como parâmetro de comparação:

- **T1 – Visualização de Topologia**: prover mecanismo de visualização de topologia para o operador de rede;
- **T2 – Interação com a Topologia**: prover, com interatividade, mecanismos básicos de gerenciamento de topologia e de QoS;
- **T3 – Independência de Softwares Proprietário**: independência de mecanismos de terceiros, de controladores SDN e protocolos proprietários.

Conforme a Tabela 1, que agrupa o estado da arte atualmente disponível na literatura, é possível constatar que apenas 01 (OrchFlow [Frate et al. 2018]) dos 10 estudos realizados em trabalhos anteriores apresentou evidências de ter se comprometido na concepção de um sistema para gerenciamento de controladores SDN. Todavia, a proposta OrchFlow não menciona a possibilidade da interoperabilidade entre os controladores da

[3]https://neo4j.com/

rede para fornecer recursos como meio de possibilitar a adaptação de funcionalidades inovadoras por meio de interfaces em alto nível.

Tabela 1. Comparativo entre os trabalhos relacionados e a proposta do sistema ATOM

Proposta	T1	T2	T3
HyperFlow, [Ganjali and Tootoonchian 2010]	✓	X	X
Onix, [Koponen et al. 2010]	✓	✓	X
Kandoo, [Hassas Yeganeh and Ganjali 2012]	✓	X	X
DISCO, [Phemius et al. 2013]	✓	✓	X
[Huang et al. 2014] Mantoo, [Salsano et al. 2015] OpenGUFI, [Schultz et al. 2015]	✓	X	X
IRIS-CoMan, [Choi et al. 2015]	✓	X	X
RUNOS, [Shalimov et al. 2015]	✓	X	X
OrchFlow, [Frate et al. 2018]	✓	✓	X
ATOM	✓	✓	✓

A próxima seção introduz o sistema ATOM, cuja proposta baseada em uma arquitetura modular que viabiliza uma melhor organização dos componentes, se apresenta como uma alternativa promissora para lidar com as lacunas não preenchidas pela literatura relacionada.

3. Proposta do Sistema ATOM

Esta seção apresenta o sistema ATOM (An **A**rchitecture for Ubiqui**TO**us **M**anagement of SDN Controllers). ATOM atua no gerenciamento de controladores SDN e é capaz de prover interfaces de comunicação padronizadas e abstrair a complexidade de implementação e utilização das funcionalidades dos controladores. As seções seguintes fornecem uma visão geral da arquitetura e apresentam em detalhes as funcionalidades de cada componente.

3.1. Visão Geral

O sistema ATOM tem como principal objetivo abstrair a dificuldade encontrada atualmente para gerenciamento das funções fornecidas pelos controladores SDN e, em paralelo, fornecer ao operador da rede uma visão em alto nível de toda a infraestrutura que está sob o domínio do controlador. Para isso, é necessário o provisionamento de interfaces de comunicação e módulos que proporcionem interoperabilidade entre o controlador e os componentes da ferramenta. A fim de prover tais funcionalidades, faz-se necessário a adição de uma nova camada que agrega os componentes responsáveis pelo gerenciamento dos recursos fornecidos pelos controladores SDN. A Figura 1 apresenta a arquitetura do sistema ATOM posicionada sob uma típica infraestrutura SDN, composta pelo controlador, demais elementos do plano de dados, interfaces de comunicação e protocolos utilizados.

ATOM é formado por um conjunto de componentes que convergem a fim de obter e agregar as informações de toda a topologia, incluindo os eventos de rede. Estas

informações são obtidas dos controladores SDN e, após abstração da complexidade relacionada à leitura e processamento, são disponibilizadas para as aplicações. Numa visão de alto nível, os componentes da arquitetura são descritos da seguinte maneira: *(i) Interface Management*; *(ii) Resource Manager*; *(iii) Server Provider*; *(iiii) Broker Server Interface*; e *(iv) Broker Client Interface*.

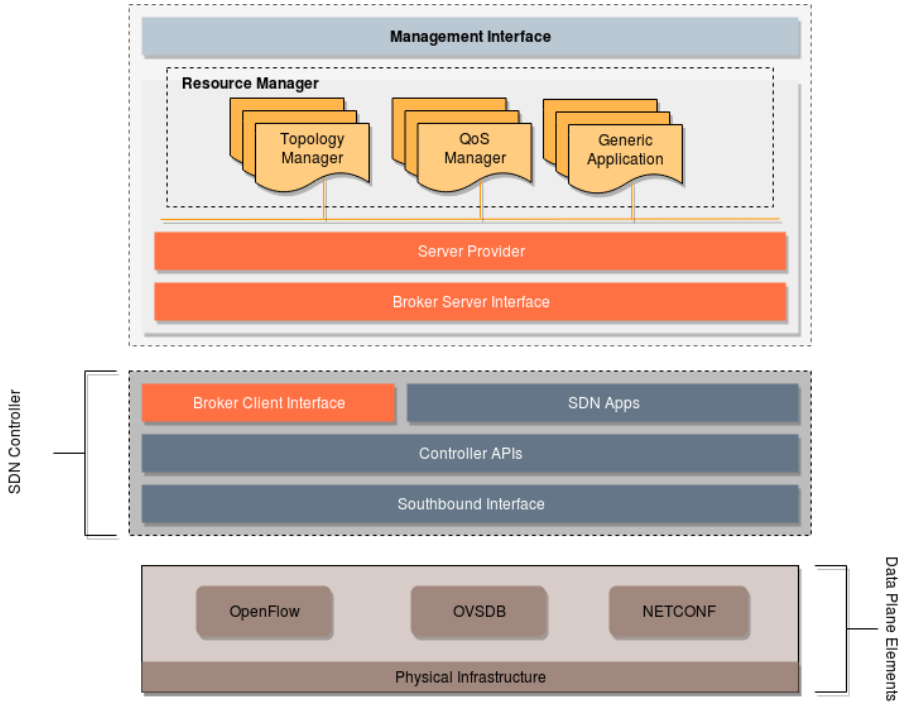

Figura 1. Arquitetura do sistema ATOM

3.1.1. Resource Manager

O *Resource Manager* é o módulo responsável por agregar os componentes de *Topology Manager* e *Qos Manager* para fornecer uma interface de comunicação com o intuito de disponibilizar as funcionalidades providas por tais componentes.

3.1.2. Topology Manager

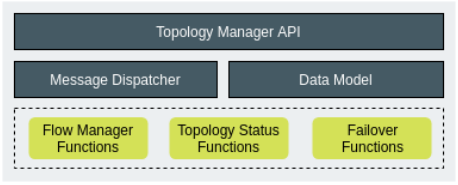

Figura 2. Arquitetura interna do *Topology Manager*

Componente responsável por três principais tarefas: *(i)* obtenção do *status* da topologia de rede em tempo real; *(ii)* provimento de funções básicas para gestão da rede

(ex. gerenciamento de fluxos); *(iii)* provimento de mecanismos para detecção de falha de comunicação entre os enlaces (*failover*). De acordo com a Figura 2, o *Topology Manager* é estrutura da seguinte maneira:

- ***Topology Manager REST API***: interface de comunicação utilizada pela interface de gerenciamento em alto nível (Management Interface);
- ***Message Dispatcher***: funções responsáveis pelo encaminhamento das requisições geradas pelas operações de gerenciamento para o Server Provider;
- ***Data Model***: modelo responsável pela definição da estruturação das requisições que são recebidas pelo *Topology Manager* e enviadas para o *Server Provider*;
- ***Flow Manager Functions***: disponibiliza as funções para gerenciamento das regras de fluxo;
- ***Topology Status Functions***: disponibiliza as funções para obtenção e gerenciamento do estado atual da topologia;
- ***Failover Functions***: funções responsáveis pela identificação (e aplicação de contramedidas) de falha em enlaces de comunicação.

3.1.3. QoS Manager

O *QoS Manager* viabiliza a criação e manutenção de políticas de QoS, possibilitando ao operador de rede a opção de configuração de reservas de recursos em caminhos selecionados. Por meio de uma abordagem simplificada, que modela os dados necessários para a operação desta funcionalidade, o operador de rede deverá apenas informar as especificações de QoS desejadas, conforme a Tabela 2.

Tabela 2. Especificação das requisições QoS Manager

Campo	Tipo	Descrição
portName	text	Porta do dispositivo a qual a política será aplicada.
type	linux-htb	Escalonador de filas.
maxRate	bits	Limite máximo de *bits* aplicados na política.
minRate	*bits*	Limite mínimo de bits aplicados na política.

3.1.4. Server Provider

A fim de disponibilizar todas as funcionalidades e informações coletadas pelos componentes previamente descritos, um sistema de *backend* apropriado é necessário para realizar todo o processamento de informações. Desta forma, o *Server Provider*, que é o ponto principal de comunicação entre o controlador SDN e os demais elementos da arquitetura, é o componente responsável por suportar toda a estruturação de componentes do sistema ATOM.

O *Server Provider* é capaz de padronizar as requisições do *Topology Manager* e do *QoS Manager*. O processo de padronização é conduzido através de um mapeamento, que converte as requisições para uma linguagem de texto com sintaxe de fácil leitura baseada no formato *JavaScript Object Notation* (JSON). Sendo assim, quando algum componente

necessita se comunicar com o controlador SDN, o *Server Provider* executa o procedimento de estruturação das mensagens. Após isso, realiza a verificação de consistência de tais informações e, em caso positivo, encaminha a requisição para o controlador SDN.

Neste contexto, o *Server Provider* pode ser visto como um intermediador entre todas as informações trafegadas entre o ATOM e o controlador SDN, uma vez que não necessita tomar conhecimento de qual o tipo de controlador ele está recebendo e solicitando informações. Por realizar o tratamento de requisições de forma assíncrona, ele é capaz de atender a requisições de controladores de forma indistinta.

3.1.5. *Broker Interfaces*

Para cumprir o objetivo da proposta, é necessário a implementação de uma camada de adaptadores para prover a intercomunicação entre o *Server Provider* e o controlador SDN. O componente responsável por tal adaptação recebe o nome de *Broker*. A arquitetura do sistema ATOM é composta por dois tipos de *Broker*:

1. ***Broker Server*** – responsável por receber as informações padronizadas pelo *Server Provider* e enviá-las para o controlador SDN.
2. ***Broker Client*** – responsável por executar, no lado do controlador SDN, as instruções recebidas pelo *Broker Server*.

3.1.6. Management Interface

Este componente é definido por uma GUI, projetada para prover ao operador de rede a visão holística de toda a infraestrutura que está sob o domínio administrativo de um determinado controlador. Através do *Management Interface* o operador da rede não necessitará realizar intervenções manuais por meio de operações em interfaces de linhas de comando (CLI, do inglês *Command-Line Interface*) dos controladores ou dos elementos de rede. Por meio desta abordagem as funcionalidades providas pelo *Resource Manager* podem ser aplicadas diretamente aos dispositivos de rede.

O *Management Interface* também é capaz de abstrair a complexidade de leitura de informações providas pelos controladores SDN, exigindo pouca intervenção manual para que seja possível a visualização de informações, a exemplo: *(i)* regras de fluxos ativas nos dispositivos de rede; *(ii)* estatísticas de tráfego dos enlaces; e *(iii)* políticas de QoS.

4. Operações e Avaliação

ATOM fornece ao operador de rede a possibilidade de gerenciamento de fluxos customizados (criação e remoção) para atender demanda específicas das aplicações. A Figura 3 detalha a sequência de operações e mensagens trocadas entre os componentes envolvidos no procedimento de criação de fluxos.

Conforme descrito na Figura 3, o operador de rede, no primeiro momento, requisita a instalação de um novo fluxo com parâmetros customizados. A requisição é então enviada para o *Topology Manager* (passo 2.1) e, por meio de seus componentes internos (*Topology Manager API* e *Message Dispatcher*), realiza a tradução e o encaminhamento da mensagem para o componente responsável pela estruturação da requisição (*Topology*

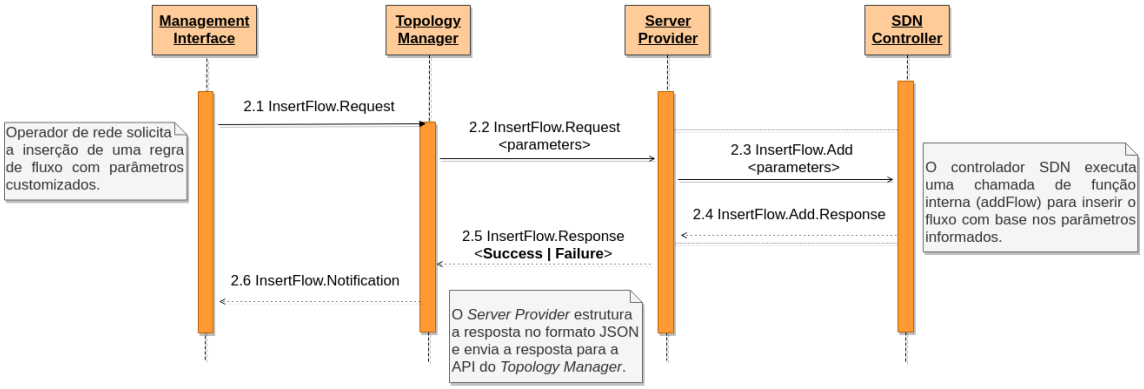

Figura 3. Diagrama de sequência do procedimento de inserção de fluxos

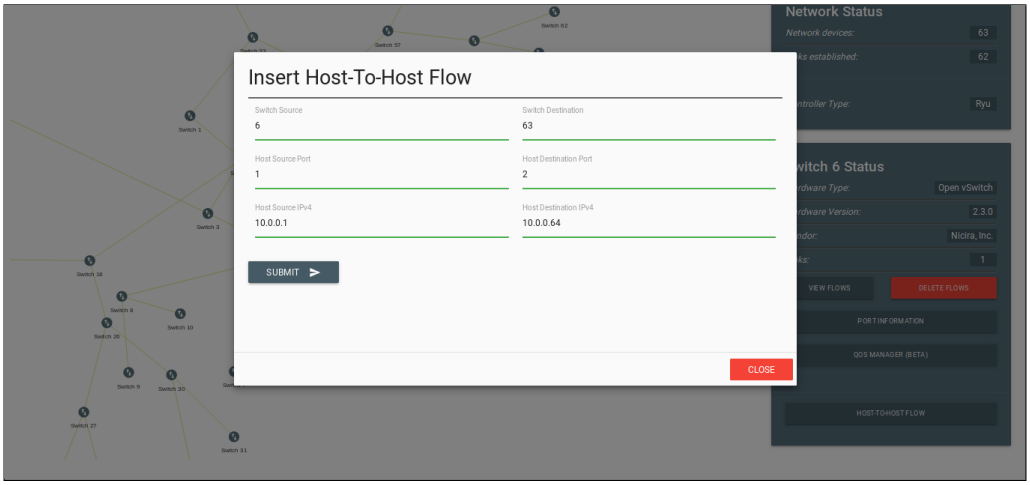

Figura 4. Visualização da interface de inserção de fluxos

Status Functions). Em seguida, a requisição é enviada para o controlador SDN, por meio do*Server Provider* (passo 2.2). O controlador SDN traduz a requisição em mensagens `FlowMod`) e executa uma chamada interna para adicionar uma nova entrada na tabela de fluxos (passo 2.3) do dispositivo indicado durante a solicitação de criação do fluxo. Após isso, o controlador envia uma mensagem ao *Server Provider* (passo 2.4) contendo o resultado do procedimento (*Success* ou *Failure*). Em ambos os casos, o operador sera notificado por meio do *Management Interface* (passo 2.6). As Figuras 4 e 5 demonstram as funcionalidades providas pelo *Management Interface* para os procedimentos de inserção e visualização das tabelas de fluxo.

Uma série de experimentos foram conduzidos com o objetivo de validar a eficiência do sistema ATOM no que diz respeito ao tempo necessário para realizar os procedimentos de orquestração por meio do *Management Interface*. Foram utilizadas 04 infraestruturas de rede com topologias configuradas de forma diferente, conforme a Tabela 3.

As infraestruturas foram logicamente virtualizadas através do emulador Mininet[4]. As avaliações foram conduzidas em um *host laptop* Dell Inspiron, Intel® Core™ i7-

[4]http://mininet.org

Tabela 3. Configuração dos cenários de avaliação

Topologia	Switches	Nº de Saltos	Nº de Fluxos**
T1	15	14	30
T2	25	24	50
T3	35	34	70
T4	45	44	90

** Representa o número de fluxos a serem instalados no experimento.

5500U, 8GB RAM, 500GB de disco rígido e sistema operacional Linux Debian 8.

Os experimentos levaram em consideração o procedimento de inserção de fluxos fim a fim entre *hosts* localizados em extremidades opostas da rede. Para isso, todas as topologias descritas na Tabela 3 foram utilizadas, e o mesmo processo foi executado 20 vezes para cada uma delas. Tomando como exemplo a topologia **T1**, que é composta por 15 switches e 2 *hosts* (separados por 14 saltos de distância), 30 regras de fluxos são necessárias para prover a comunicação entre os *hosts* (fluxos de ida e fluxos de volta). A Figura 6 revela os resultados obtidos para cada topologia a partir dos tempos médio gastos para a instalação das respectivas regras de fluxos.

De acordo com os resultados mostrados na Figura 6, a inserção de fluxos provida pelas funcionalidades do sistema ATOM foi realizada em um tempo menor do que na abordagem tradicional (fazendo uso de scripts baseados em Python [5]). A Tabela 4 apresenta um comparativo entre os resultados obtidos (a partir da média aritmética simples dos resultados obtidos em cada uma das 20 execuções para cada topologia) na avaliação e o percentual de otimização alcançado pela abordagem utilizada no sistema ATOM.

Embora a organização da arquitetura do sistema ATOM sugira a perspectiva de maior latência nas atividades de orquestração, os resultados das avaliações mostraram que os benefícios vão além das funcionalidades descritas nas seções anteriores. No caso deste experimento, ao executar apenas uma requisição que insere todos os fluxos em um

[5]Scripts disponível em: https://github.com/latarc/expsltrc

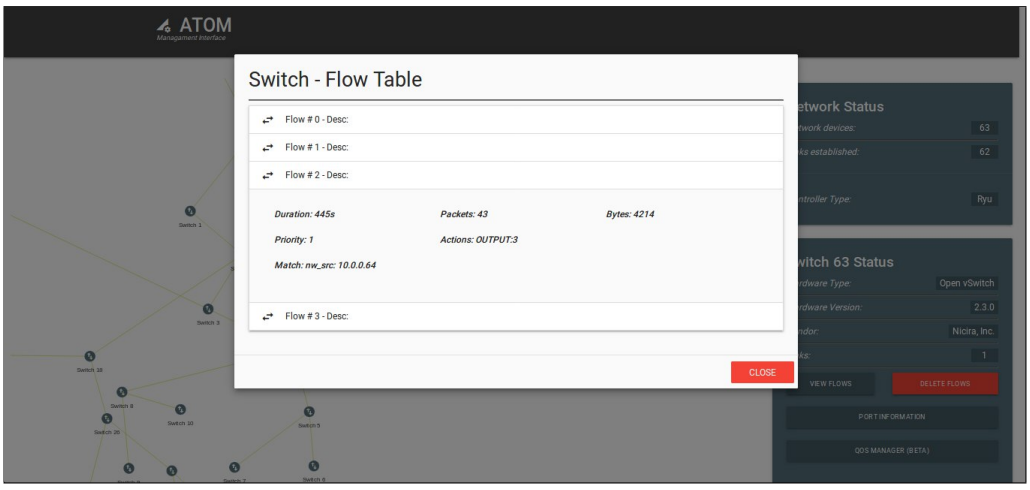

Figura 5. Visualização de informações das tabelas de fluxo dos dispositivos

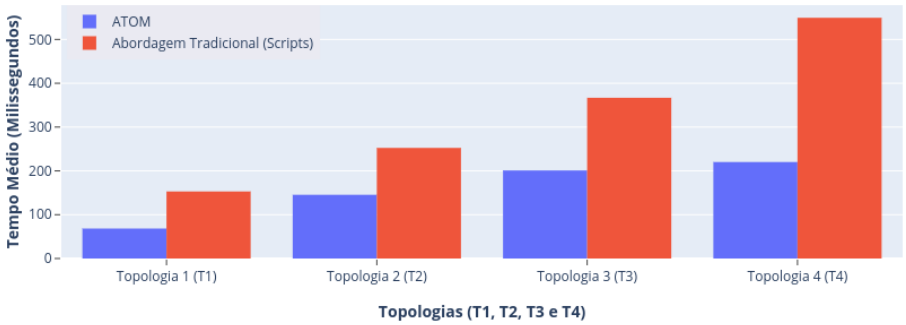

Figura 6. Experimento para avaliação da inserção de fluxos

Tabela 4. Resultados do experimento de inserção de fluxos

Topologia	ATOM	Inserção manual	Otimização provida por ATOM
T1	68.4ms	153.0ms	55.3%
T2	145.4ms	252.4ms	42.4%
T3	201.0ms	366.8ms	45.21%
T4	220.2ms	549.4ms	59.92%

determinado caminho (ATOM), o tempo de processamento das mensagens e da inserção de fluxos acaba sendo inferior ao tempo gasto pela inserção manual devido ao processamento único da requisição (necessário para prosseguir com a configuração dos fluxos em um determinado caminho). Em contraste, a inserção de fluxos manuais necessitará de um requisição para cada fluxo a ser instalado, o que pode representar um problema de escalabilidade devido ao número de dispositivos existentes na topologia.

5. Considerações Finais

Neste trabalho apresentamos o ATOM, uma arquitetura modular capaz de maximizar a capacidade de orquestração de controladores SDN, permitindo uma melhor utilização dos recursos de rede e na redução de execução de operações manuais. Por meio da separação das funcionalidades em componentes especializados, ATOM é capaz de prover a interação com os dispositivos da rede de forma direta. Através das especificações da arquitetura é possível disponibilizar uma interface de gerenciamento para viabilizar o gerenciamento holístico da infraestrutura que está associada a um determinado controlador.

ATOM foi projetado para implementar, de forma padronizada, o fluxo de mensagens relacionados às requisições necessárias para configuração das aplicações. Desta forma, é possível implementar módulos que utilizam as *Northbound Interfaces* disponibilizadas pelos controladores SDN para proporcionar o gerenciamento da topologia em tempo real.

Para os trabalhos futuros pretende-se aprimorar o gerenciamento das interações entre o ATOM e os controladores a fim de proporcionar melhor experiência para adaptação de novas funcionalidades. Dentre as principais, destaca-se a otimização dos

mecanismos disponibilizados pelo *Topology Manager*, em específico para o mecanismo de *failover*, para realizar a seleção do caminho ótimo baseado na disponibilidade de recursos dos enlaces. Além disso, tenciona-se estender o conceito de gerenciamento ubíquo provido por ATOM para o gerenciamento agnóstico de controladores. Desta maneira, esperamos fornecer a capacidade de orquestração de múltiplos controladores diferentes a partir de uma mesma *Management Interface* através da adoção de templates previamente configurados que irão definir a forma de comunicação com os controladores SDN, disponibilizando interfaces agnósticas para tal comunicação. Futuras avaliações consistirão de analises das demais operações fornecidas pelo ATOM, a fim de mensurar o tempo de gerenciamento em diferentes cenários para servir como prova de conceito para utilização do trabalho proposto.

6. Agradecimentos

Esta pesquisa foi parcialmente apoiada pela 4ª chamada colaborativa EU-BR (H2020, *grant agreement* no. 777067, NECOS – *Novel Enablers for Cloud Slicing*), financiada pela Comissão Européia e pelo Ministério da Ciência, Tecnologia, Inovação e Comunicação (MCTIC), através da RNP e CTIC. Os autores também agradecem ao Conselho Nacional de Desenvolvimento Científico e Tecnológico (CNPq) e ao IFRN pelo apoio.

Referências

Choi, T., Lee, B., Kang, S., Song, S., Park, H., Yoon, S., and Yang, S. (2015). Iris-coman: Scalable and reliable control and management architecture for sdn-enabled large-scale networks. *J. Netw. Syst. Manage.*, 23(2):252–279.

Frate, M., Marczuk, M. K., and Verdi, F. L. (2018). OrchFlow: An Architecture for Orchestration of Multiple Controllers in OpenFlow Networks. *Journal of Network and Systems Management*, (0123456789).

Ganjali, Y. and Tootoonchian, A. (2010). Hyperflow: A distributed control plane for openflow. In Akella, A., Feamster, N., and Rao, S. G., editors, *INM/WREN*. USENIX Association.

Gomes, C. S., Silva, F. S. D., Neto, E. P., Costa, K. B., and da Silva, J. B. (2016). Towards a modular interactive management approach for sdn infrastructure orchestration. In *2016 IEEE Conference on Network Function Virtualization and Software Defined Networks (NFV-SDN)*, pages 1–6.

Guck, J. W., Bemten, A. V., and Kellerer, W. (2017). Detserv: Network models for real-time qos provisioning in sdn-based industrial environments. *IEEE Transactions on Network and Service Management*, 14(4):1003–1017.

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.

Haleplidis, E., Pentikousis, K., Denazis, S., Salim, J. H., Meyer, D., and Koufopavlou, O. (2015). Software-Defined Networking (SDN): Layers and Architecture Terminology. RFC 7426.

Hassas Yeganeh, S. and Ganjali, Y. (2012). Kandoo: A framework for efficient and scalable offloading of control applications. In *Proceedings of the First Workshop on Hot Topics in Software Defined Networks*, HotSDN '12, pages 19–24, New York, NY, USA. ACM.

Huang, W.-Y., Chou, T.-Y., Hu, J.-W., and Liu, T.-L. (2014). Automatical end to end topology discovery and flow viewer on sdn. In *Proceedings of the 2014 28th International Conference on Advanced Information Networking and Applications Workshops*, WAINA '14, pages 910–915, Washington, DC, USA. IEEE Computer Society.

Koponen, T., Casado, M., Gude, N., Stribling, J., Poutievski, L., Zhu, M., Ramanathan, R., Iwata, Y., Inoue, H., Hama, T., and Shenker, S. (2010). Onix: a distributed control platform for large-scale production networks. In *Proceedings of the 9th USENIX conference on Operating systems design and implementation*, OSDI'10, pages 1–6, Berkeley, CA, USA. USENIX Association.

Phemius, K., Bouet, M., and Leguay, J. (2013). Disco: Distributed multi-domain sdn controllers. *CoRR*, abs/1308.6138.

Salsano, S., Ventre, P. L., Lombardo, F., Siracusano, G., Gerola, M., Salvadori, E., Santuari, M., Campanella, M., and Prete, L. (2015). Mantoo - a set of management tools for controlling sdn experiments. In *Proceedings of the 2015 Fourth European Workshop on Software Defined Networks*, EWSDN '15, pages 123–124, Washington, DC, USA. IEEE Computer Society.

Schultz, J., Szczepanski, R., Haensge, K., Maruschke, M., Bayer, N., and Einsiedler, H. (2015). Opengufi: An extensible graphical user flow interface for an sdn-enabled wireless testbed. In *Computer and Information Technology; Ubiquitous Computing and Communications; Dependable, Autonomic and Secure Computing; Pervasive Intelligence and Computing (CIT/IUCC/DASC/PICOM), 2015 IEEE International Conference on*, pages 770–776.

Sezer, S., Scott-Hayward, S., Chouhan, P. K., Fraser, B., Lake, D., Finnegan, J., Viljoen, N., Miller, M., and Rao, N. (2013). Are we ready for sdn? implementation challenges for software-defined networks. *IEEE Communications Magazine*, 51(7):36–43.

Shalimov, A., Nizovtsev, S., Morkovnik, D., and Smeliansky, R. L. (2015). The runos openflow controller. In *EWSDN*, pages 103–104. IEEE Computer Society.

Sharma, P., Banerjee, S., Tandel, S., Aguiar, R., Amorim, R., and Pinheiro, D. (2013). Enhancing network management frameworks with sdn-like control. In *Integrated Network Management (IM 2013), 2013 IFIP/IEEE International Symposium on*, pages 688–691.

The Apache Software Foundation (2016). Apache ZooKeeper. [Online] Accessed 4 Mar 2016. Available at: `https://zookeeper.apache.org/`.

Voellmy, A., Wang, J., Yang, Y. R., Ford, B., and Hudak, P. (2013). Maple: Simplifying sdn programming using algorithmic policies. In *Proceedings of the ACM SIGCOMM 2013 Conference on SIGCOMM*, SIGCOMM '13, pages 87–98, New York, NY, USA. ACM.