
UNIVERSIDADE FEDERAL DO PARÁ
INSTITUTO DE CIÊNCIAS EXATAS E NATURAIS
PROGRAMA DE PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

RÔMULO SILVA PINHEIRO

# REPOSDN: UM MÉTODO DE ORGANIZAÇÃO E COORDENAÇÃO DE APLICAÇÕES EM REPOSITÓRIO PARA REDES DEFINIDAS POR SOFTWARE

BELÉM-PA

02 / 2014

**RÔMULO SILVA PINHEIRO**

# REPOSDN: UM MÉTODO DE ORGANIZAÇÃO E COORDENAÇÃO DE APLICAÇÕES EM REPOSITÓRIO PARA REDES DEFINIDAS POR SOFTWARE

Dissertação submetida à banca julgadora na Universidade Federal do Pará como parte dos requisitos para obtenção do grau de Mestre em Ciência da Computação

Orientador: Dr. Antônio J. Gomes Abelém

**BELÉM-PA**

**02 / 2014**

**RÔMULO SILVA PINHEIRO**

# REPOSDN: UM MÉTODO DE ORGANIZAÇÃO E COORDENAÇÃO DE APLICAÇÕES EM REPOSITÓRIO PARA REDES DEFINIDAS POR SOFTWARE

Dissertação submetida à banca julgadora na Universidade Federal do Pará como parte dos requisitos para obtenção do grau de Mestre em Ciência da Computação

Aprovada em: __/__/____

**BANCA EXAMINADORA**

---

Prof. Dr. Antônio J. Gomes Abelém
Universidade Federal do Pará
Orientador

---

Prof. Dr. Eduardo Coelho Cerqueira
Universidade Federal do Pará

---

Prof. Dr. Josivaldo de Souza Araújo
Universidade Federal do Pará

Prof. Dr. Rodrigo Quites Reis
Universidade Federal do Pará

*A meus familiares.*

# Agradecimentos

Agradeço a Deus por ser um apoio muito importante em minha vida.

Serei eternamente grato a minha família que é e sempre será o meu alicerce. Meu pai maravilhoso que sempre mostrou para seus filhos que a educação é o bem mais valioso que um homem pode ter. A minha mãe bondosa que tem um coração gigante e um amor infinito, obrigado pelos sacrifícios que a senhora fez pelos seus filhos, já disse que te amo hoje?

Ao meu orientador Prof. Dr. Antonio J. G. Abelém e Co-Orientador Msc. Billy A. Pinheiro que me deram um voto de confiança e acreditaram no meu trabalho, serei grato eternamente pela oportunidade e levarei todo o aprendizado que tive em prol daqueles que também precisam de uma pequena chance, obrigado.

A todos os membros do GERCOM que sempre me ajudaram nas horas de dúvidas e tiveram paciência e compreensão para me aturar.

A todos os meus amigos que compreenderam os momentos que tive que dizer "não"quando me chamavam para beber.

A CAPES e FAPESPA pelo apoio financeiro em forma de bolsa.

# Resumo

Resumo da Dissertação apresentada à UFPA como parte dos requisitos necessários para obtenção do grau de Mestre em Ciência da Computação.

## RepoSDN: Um Método de Organização e Coordenação de Aplicações em Repositório para Redes Definidas por Software

Orientador: Dr. Antônio J. Gomes Abelém
Palavras-chave: Software Defined Network; Repositório de Aplicações; Aplicações SDN; Gerenciamento de Aplicações SDN; Controladores

A adoção do paradigma de Redes Definidas por Software por instituições acadêmicas e grandes empresas como Google, Cisco, HP, entre outras, já é uma realidade. Este paradigma é considerado uma solução para o novo modelo de Internet do Futuro devido a sua viabilidade técnica e econômica. O paradigma SDN permite novas perspectivas em termos de abstrações, ambientes de controle e de desenvolvimento de aplicações, livre das limitações de tecnologias atuais. Contudo ainda há carência na forma como ocorre o gerenciamento e controle de diversas aplicações existentes na Internet.

A Programabilidade é uma das chaves para o sucesso do paradigma de Redes Definidas por Software, pois permite a separação do plano de dados com o plano de controle. No entanto, no contexto deste paradigma, gerenciar a entrada e saída de diversas aplicações para os controladores é um problema pouco explorado, visto que as aplicações SDN, como as aplicações de roteamento, por exemplo, não ficam em um ambiente controlado e organizado.

Nesta dissertação é proposto um método de organização e coordenação de repositório de aplicações para Redes Definidas por Software, chamado de RepoSDN, que especifica, modela e automatiza os processos necessários para que as aplicações cheguem até o administrador e este possa gerenciar suas utilizações na rede, tornando-as simples, seguras e ágeis.

O RepoSDN apresenta uma estrutura onde cada aplicação SDN é gerenciada dentro da rede até chegar aos controladores. Além disso, ele contém um repositório central, onde qualquer usuário pode enviar suas aplicações a fim de compartilhar com os administradores da Rede Definida por Software, os quais utilizam aplicações específicas para a sua necessidade.

Foi desenvolvido um protótipo do RepoSDN, o qual implementa as funcionalidades do repositório de aplicações e dos componentes criados para auxiliar na gerência das aplicações em uma rede programável. O método de organização e coordenação de aplicações em repositório para SDN auxilia o Administrador a manipular cada aplicação.

A proposta foi validada por meio do desenvolvimento do repositório e realização de testes de desempenho no mesmo, que constituiu em fazer testes de carga nos componentes desenvolvidos. O objetivo dos componentes é tornar mais simples o método com que as aplicações são inseridas nos diversos tipos de controladores das Redes Definidas por Software, tendo como resultado a comprovação da viabilidade, escalabilidade e flexibilidade do RepoSDN.

# Abstract

Abstract of Dissertation presented to UFPA as a partial fulfillment of the requirements for the degree of Master in Computer Science.

## RepoSDN: A Method for Organization and Coordination Applications Repository for Software Defined Networks

Advisor: Dr. Antônio J. Gomes Abelém
Key words: Software Defined Network; Application Repository; SDN Applications; Application Management SDN; Controllers

The adoption of the paradigm of Software Defined Networks - SDN by academic institutions and large companies like Google, Cisco, HP, among others, is already a reality. This paradigm is considered a solution to the new model of Future Internet due to its technical and economic viability. The SDN paradigm enables new perspectives in terms of abstractions, control and development applications, free from the limitations of current technologies environments. However there is still lack in the way the management and control of several existing applications in the Internet occurs.

The programmability is key to successful paradigm for Software Defined Networks as it allows the separation of the data with the control plane level. However, in the context of this paradigm, manage input and output controllers for many applications is a relatively unexplored issue, since the SDN applications, routing applications such as, for example, are not organized in a controlled environment.

This thesis proposes a method of organization and coordination applications in repository for software defined networks, called RepoSDN, which specifies, models and automates the processes needed for applications reaching the administrator can manage this and their uses in the network, making it simple, safe and agile.

The RepoSDN presents a structure where each SDN application is managed

within the network to reach out to controllers. Additionally, it contains a central repository, where any user can send their applications to share with management Software Defined Network, which use specific applications for your need.

A prototype of RepoSDN, which implements the functionality of the applications repository and components designed to help in the management of applications on a programmable network was developed. The method of organization and coordination of applications repository for SDN helps the administrator to handle each application.

The proposal has been validated through the development of the repository of application, which consisted in doing load testing in developed component testing. The objective of the components is becoming simpler method that applications are entered in the various types of controllers of Software Defined Networking, resulting in the proof of feasibility, scalability and flexibility of RepoSDN.

# Sumário

# Lista de Abreviaturas

| | |
|---|---|
| SDN | Redes Definidas por Software |
| IF | Internet do Futuro |
| QoS | Qualidade de Serviço |
| QoE | Qualidade de Experiência |
| PC | Plano de Controle |
| PD | Plano de Dados |
| API | Application Programming Interface |
| SDH | Synchronous Digital Hierarchy |
| SSL | Secure Socket Layer |
| TCP | Transmission Control Protocol |
| SNMP | Simple Network Management Protocol |
| CLI | command-line interface |
| SA | Sistemas Autônomos |
| ISO | International Standards Organization |
| SDK | Software Development Kit |
| ADT | Android Developer Tools |
| SO | Sistema Operacional |
| HIG | Human Interface Guidelines |
| APT | Advanced Packaging Tool |
| HP | Hewlett-Packard |
| GPL | General Public License |
| RIPng | Routing Information Protocol Next Generation |
| OSPF | Open Shortest Path First |
| UML | Unified Modeling Language |
| ONF | Open Network Foundation |
| NBIs | Northbound API Interfaces. |

| | |
|---|---|
| FSF | Free Software Foundation |
| GPL | General Public License |
| OSI | Open Source Initiative |
| IP | Internet Protocol |

# Lista de Figuras

# Lista de Tabelas

# CAPÍTULO 1

# Introdução

Este capítulo discute como a gerência e distribuição de aplicações de rede passaram a ser muito importantes para o paradigma de Redes Definidas por Software - SDN. A Seção 1.1 traz a visão geral de como a Internet tem um papel fundamental na organização das aplicações de rede. As Seções 1.2 e 1.3 apresentam respectivamente a motivação e objetivos do trabalho e a Seção 1.4 apresenta a organização do trabalho.

## 1.1 Visão geral

A Internet atualmente é usada para uma grande variedade de propósitos comerciais e não comerciais. Para muitas pessoas é uma ferramenta de trabalho importante e também um eficiente meio de comunicação, entretenimento, entre outros. A Internet é um enorme sucesso mundial e vem mudando a forma como interagimos, trabalhamos e nos divertimos [Farias et al. 2011].

A Internet vem sofrendo muitas extensões ao longo dos anos para incluir novas funcionalidades, as quais não foram previstas no projeto inicial. Muitos especialistas de rede agora consideram que é necessário conduzir um estudo de arquiteturas alternativas para Internet do Futuro - IF, como uma maneira realmente eficiente de resolver muitos dos problemas prementes que atualmente afligem a Internet [Farias et al. 2011].

Segundo Paul e Jain [Paul et al. 2011], a nova arquitetura para Internet do Futuro deve oferecer suporte às novas tecnologias de rede, segurança e robustez, qualidade de serviço - QoS, qualidade de experiência - QoE, separação dos planos de controle - PC e o plano de dados - PD, mobilidade e escalabilidade.

Atualmente existem duas abordagens sendo pesquisadas. A primeira é intitulada de *Clean Slate*, cujo objetivo é substituir a arquitetura atual por uma nova totalmente

reconstruída, como se não tivesse havido arquitetura alguma. A outra, chamada de *Evolutionary*, que pretende evoluir a arquitetura atual sem perder a compatibilidade com a arquitetura anterior.

A adoção de qualquer uma das arquiteturas alternativas, *Clean Slate* ou *Evolutionary*, pode alterar a situação em que a Internet atual se encontra. Entretanto, um sério obstáculo para adoção efetiva de tais inovações tem sido a inabilidade de validá-las de maneira convincente. A redução no impacto do mundo real de qualquer inovação se deve, em grande parte, à enorme base instalada de equipamentos e protocolos e à relutância em experimentar com tráfego de produção, o que tem criado uma barreira extremamente alta para a entrada de novas ideias. O resultado é que a maior parte das novas ideias da comunidade de pesquisa de rede não é testada, levando à crença comumente mantida de que a infraestrutura da Internet "ossificou"ou enrijeceu [Paul et al. 2011].

As redes de computadores são dinâmicas e complexas, como resultado, a sua configuração e o seu gerenciamento continuam sendo um desafio. Estas redes normalmente compreendem um grande número de comutadores com muitos tipos de eventos que ocorrem simultaneamente, sendo os administradores de rede os responsáveis pela sua configuração, a fim de aplicar várias políticas de alto nível, para responder à ampla gama de eventos de rede [Kim and Feamster 2013].

Uma das formas de se prover autonomia às redes de computadores é através da implementação das Redes Definidas por Software, que são redes cujo substrato físico é composto por equipamentos onde o plano de controle é realizado por um componente externo chamado de controlador, e a redução no funcionamento de cada equipamento, ou conjunto de equipamentos, é definido por um conjunto de softwares especializados armazenados dentro deste componente. SDN torna as redes mais flexíveis, com baixo custo [Greenhalgh et al. 2009].

Neste contexto, as Redes Definidas por Software surgem como um meio para a alternativa do paradigma de redes *Clean Slate* [Kind et al. 2012]. As instituições irão possuir um controle maior sobre a sua rede, oferecendo uma maior segurança para a integridade dos dados e paralelo a isso, dispositivos de hardware especializados serão substituídos por hardwares preparados para esse novo paradigma, apoiados por aplicativos especialmente desenvolvidos para funcionar sobre as SDN.

Para as corporações e provedores de serviços, SDN é uma realidade que vem revolucionando as redes de computadores. É importante saber que este novo paradigma amplia todo o serviço virtual de redes e as aplicações que nela já existem. As novas implementações e conceitos de gerenciamento e serviços estão baseadas em software, e não mais no hardware [de Oliveira Silva et al. 2012].

O gerenciamento de aplicações em Redes definidas por Software, ainda é uma solução pouco explorada. As aplicações de rede se encontram espalhadas em diversos repositórios de códigos na Internet, ou ainda são embutidas de fábrica nos comutadores. Esta dissertação desenvolve todo um ecossistema entre API, aplicações SDN, repositórios e controladores, o que possibilita um melhor gerenciamento e controle das aplicações SDN,

definindo toda a sua trajetória, iniciada no desenvolvedor até o seu uso pelos controladores.

## 1.2 Motivação e desafios

Há alguns anos, poucas empresas tinham o domínio sobre a informática. As grandes empresas possuíam o domínio tanto no ramo de servidores quanto nos de computadores pessoais, e as pessoas e empresas eram obrigadas a comprar seus softwares proprietários para usar em hardwares específicos.

Atualmente as empresas fabricam componentes e permitem que os próprios usuários possam desenvolver e inserir seus aplicativos no mesmo, tendo em vista esse fato, se tornou comum existirem diversas aplicações de rede espalhadas na Internet, sem existir um local onde centralizar essas aplicações e muito menos uma forma de como gerenciá-las até chegar em seu destino final.

O paradigma SDN permite que cada aplicação SDN criada possa ser inserida em qualquer controlador dentro da SDN, aumentando a eficiência de distribuição e gerenciamento das aplicações na rede, levando em consideração que o administrador desta rede terá acesso a um repositório central que conterá todas as aplicações enviadas.

## 1.3 Objetivos

O Objetivo principal desta dissertação é desenvolver um método de organização e coordenação de aplicativos armazenados em repositórios para Redes Definidas por Software, chamado de RepoSDN. Esse método permite especificar e modelar todo o processo necessário para que uma aplicação chegue até o administrador da rede programável e este possa gerenciar a sua utilização nos controladores da rede. O RepoSDN oferece uma maior organização e agilidade no que diz respeito a gerência de aplicativos na rede programável, pois permite que o administrador determine o destino de cada aplicação.

Desta forma, para alcançar o objetivo principal exposto, os seguintes objetivos específicos precisam ser superados:

- Possibilitar que o administrador da Rede Definida por Software use os aplicativos nos controladores de sua rede;
- Desenvolver modelos de organização para as aplicações;
- Definir os métodos de implantação do RepoSDN;
- Definir os métodos de administração das aplicações;
- Especificar e modelar os requisitos do RepoSDN;
- Definir a arquitetura do RepoSDN;

- Desenvolver um protótipo;
- Validação do método de organização e coordenação das aplicações SDN.

## 1.4 Organização do texto

Esta dissertação está estruturada da seguinte forma: O Capítulo 2 aborda as questões referentes a SDN, incluindo um breve histórico, os principais conceitos sobre SDN, a arquitetura, bem como uma explanação sobre a separação do plano de dados e de controle.

O Capítulo 3 apresenta os trabalhos relacionados, e as dificuldades encontradas, a comparação dos modelos baseados para o desenvolvimento da dissertação, seguido de uma breve análise sobre os trabalhos relacionados.

O Capítulo 4 apresenta o RepoSDN: Um Método de Organização e Coordenação de Repositório de Aplicações para Redes Definidas por Software, sendo descrito a sua arquitetura, além de mostrar o modelo de gerenciamento e distribuição de aplicações SDN.

O Capítulo 5 apresenta as políticas de uso propostas para o repositório de aplicações. Neste capítulo é explicado como as aplicações devem ser empacotadas, assim como o conteúdo de cada uma e os usuários que têm acesso.

O Capítulo 6 apresenta uma explanação sobre os testes de carga e de desempenho feitos para a validação da proposta, assim como o cenário utilizado e os resultados gerados.

Enfim, o Capítulo no 7 são apresentadas as contribuições, uma avaliação geral da proposta e sugestões de trabalhos futuros.

# CAPÍTULO 2

# Redes Definidas por Software

Este capítulo disserta sobre as principais questões referentes as Redes Definidas por Software. A Seção 2.1 apresenta um breve histórico e a evolução da SDN. Na Seção 2.2 é descrito o conceito da SDN. Na Seção 2.3 é apresentada a arquitetura e, por fim, a Seção 2.4 mostra o conceito, os tipos de controladores SDN. A Seção 2.5 apresenta a conclusão geral do capítulo.

## 2.1 Histórico e Evolução SDN

As atuais tecnologias de redes de computadores têm uma certa dificuldade para suportar a alta demanda dos usuários e das empresas, devido a uma grande quantidade de recursos exigidos pelos mesmos. Sendo assim, são desenvolvidos novos protocolos que de alguma forma melhoram ambientes de redes proprietários ou desenvolvidos de forma isoladas por instituições.

Dessa forma as redes de computadores se tornaram enormes e ao mesmo tempo estão restritas pela sua arquitetura distribuída e fechada. Contrapondo a abordagem da arquitetura atual das redes de computadores, surgiram ideias e pesquisas sobre a Internet do Futuro, pesquisas que visam resolver os problemas até então enfrentados pela sua arquitetura, tais como problemas associados ao endereçamento, a mobilidade, a escalabilidade, o gerenciamento e a qualidade de serviço [Costa 2013].

Quando é preciso dar escalabilidade a rede e acrescentar mais dispositivos, essa tarefa vai ficando cada vez mais complexa, pois como muitos dispositivos são de fabricantes diferentes e as vezes cada um possui seus próprios sistemas, pode não haver interoperabilidade entre esses comutadores, e quando isso acontece, novos aplicativos e protocolos são desenvolvidos, o que torna o processo lento inviabilizando a implantação de novas

tecnologias em uma rede existente.

A Rede Definida por Software é uma tecnologia que surgiu em 2008, com a parceria entre a universidade de Berkley e a universidade de Stanford. A ideia veio com base no conceito de software livre, ou seja, ter o código aberto para que toda a comunidade de pesquisadores e grandes empresas possam usufruir e se beneficiar desse novo paradigma. Esse novo tipo de redes de computadores logo chamou a atenção da comunidade, pois permitem com que os softwares dos comutadores fiquem dissociados dos hardwares, provendo independência e aumento da qualidade de gerencia e segurança.

As redes programáveis possibilitam a implementação de diversas aplicações de rede que dão autonomia para os elementos pertencentes a ela, realizam monitoramento e acompanhamento de tráfego dos pacotes e controle de tabelas de rotas dos comutadores de uma maneira bem mais eficiente [Kim and Feamster 2013].

## 2.2 Conceito

Com SDN os parâmetros do sistema operacional dos comutadores são transferidos para um controlador central, onde este terá como função o controle e gerência da rede, fazendo com que os comutadores que antes vinham com software de fábrica, percam a sua autonomia e seu sistema fazendo com que os comutadores sejam comprados "limpos", passando a serem e controlados em um servidor chamado controlador onde o plano de dados e de controle são dissociados, o que é uma das principais características das redes programáveis [Sezer et al. 2013].

Com a existência dos controladores na rede, as novas políticas de roteamento e otimização de comutação poderão ser implementadas e utilizadas nessas redes dinâmicas, reduzindo os investimentos iniciais de uma empresa, ou seja, utilizando equipamentos já existentes.

Os planos de controle e os de dados são separados, a inteligência de rede é logicamente centralizada, com isso, empresas e operadoras ganham maior controle de rede, permitindo a construção de redes altamente escaláveis e flexíveis, que prontamente podem se adaptar às necessidades empresariais em constante mudança.

Uma vantagem desta separação de planos é a adoção de novas tecnologias de plano de dados pela arquitetura de rede como: um comprimento de onda; quadro SDH; ou uma linha de transmissão de energia (Power Line Comunication - PLC). Logo a separação entre os planos é parte integrante desta próxima geração da arquitetura da Internet [Farias et al. 2011].

As Redes Definidas por Software são acessíveis, pois seu conceito básico possui desde hardware comum, como os de computadores pessoais, até softwares especializados de redes. As empresas irão possuir um controle maior sobre a sua rede, possibilitando uma maior segurança e paralelo a isso, dispositivos de hardware especializados serão substituídos por hardwares comuns, apoiados por softwares especialmente desenvolvidos

para funcionar sobre essas plataformas.

## 2.3 Arquitetura SDN

A Figura 1 mostra de uma forma lógica a arquitetura SDN. Fica clara a proposta de inteligência centralizada da SDN, como resultado a camada de controle aparece para as demais como um interruptor lógico, o que simplifica o modelo e a funcionalidade da rede. As SDN também simplificam os próprios dispositivos de rede, uma vez que já não precisam compreender e processar vários protocolos, mas apenas aceitar instruções dos controladores.

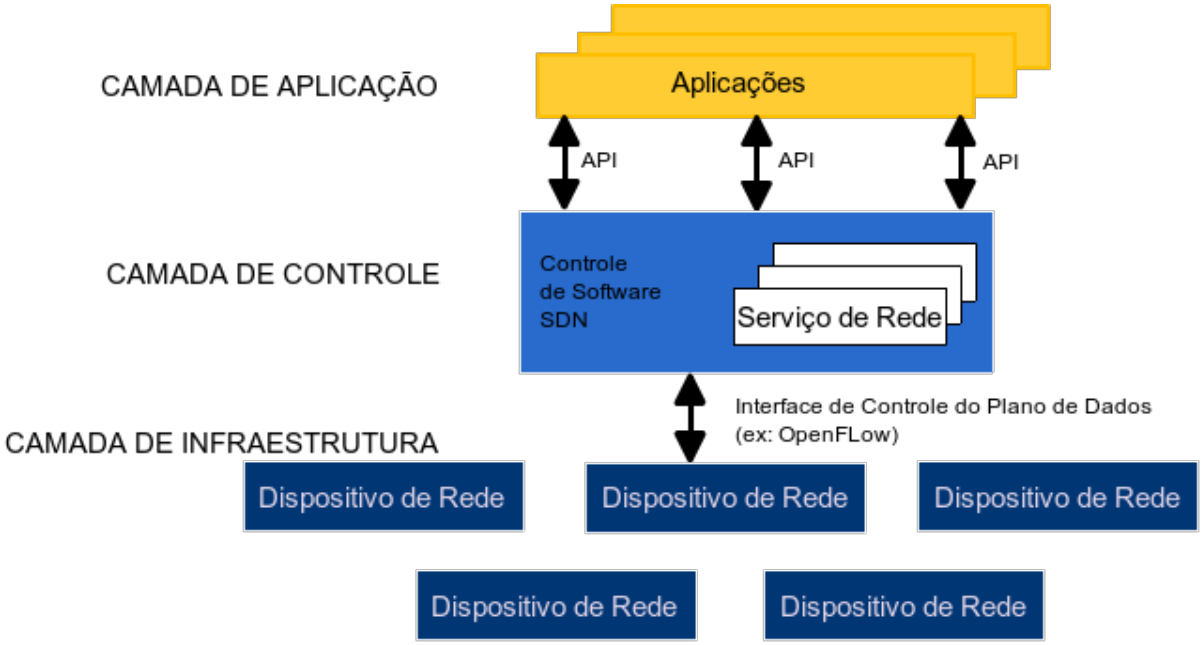


Figura 1: Arquitetura SDN [Foundation 2014].

Uma rede definida por software contém elementos de rede habilitados para que o estado das tabelas de encaminhamento possam ser instalados através de um canal seguro, conforme as decisões de um elemento central da rede. Os componentes da arquitetura destas redes são descritos a seguir:

- Tabelas de Fluxos: Cada entrada na tabela de fluxos do hardware consiste em regras, ações e contadores. A regra a ser aplicada a um determinado fluxo que chega na rede é formada com referência na definição de um ou mais campos do cabeçalho do pacote. Associa-se a ela um conjunto de ações que definem o modo com que os pacotes devem ser processados e para onde eles devem ser encaminhados. Os contadores são usados com a finalidade de manter estatísticas de utilização e também servem para remover fluxos inativos. As entradas nas tabelas de fluxos podem ser interpretadas como decisões em hardware do plano de controle (software), sendo, portanto, a mínima unidade de informação presente no plano de dados da rede.

- O Canal Seguro: Como a rede é toda desenvolvida em cima de protocolos públicos, ou seja, abertos, é necessário que se tenha um canal para que possa trocar de forma segura informações entre o comutador e o controlador, sem que sofra ataque de elementos mal-intencionados. A interface de acesso recomendada é o protocolo *Secure Socket Layer* - SSL, no entanto, interfaces alternativas (passivas ou ativas) incluindo-se o TCP são essenciais em ambientes virtuais e experimentais pela facilidade de utilização, pois não necessitam de chaves criptográficas.

- OpenFlow [Openflow 2011]: É um protocolo aberto utilizado para a comunicação, fazendo uso de uma interface de acesso, para a troca de mensagens entre os equipamentos de rede e os controladores. É importante lembrar que o protocolo Openflow não é o único protocolo disponível, pois o laboratório *Open Networking* (ON.LAB) vai lançar em breve um sistema operacional de rede de código aberto, chamado ONOS [ON.LAB 2014]. O Openflow é, basicamente, um protocolo de comunicação entre um Controlador e Switches openflow-enabled (termo usado para definir um dispositivo com suporte ao Openflow), por meio de formatos de mensagens e eventos [dos Reis et al. 2013].

Os administradores de redes poderão configurar apenas um dispositivo e economizar milhares de linhas de código em diversos dispositivos de redes, quando ele terá concentrado apenas no controlador. Além disso, aproveitando a inteligência centralizada do controlador SDN, ele pode alterar o comportamento da rede em tempo real, e implantar novos aplicativos e serviços de rede em questão de horas ou dias, em vez das semanas ou meses necessários hoje. Como essa rede está centralizada na camada de controle, isso dá aos administradores de rede a flexibilidade para configurar, gerenciar, proteger e aperfeiçoar os recursos de rede através de programas automatizados.

Além de abstrair a rede, a arquiteturas SDN tem um conjunto de APIs que tornam possível a implementação de serviços de rede já conhecido, como roteamento, *multicast*, segurança, controle de acesso, gerenciamento de banda, engenharia de tráfego, qualidade de serviço, processador e otimização de armazenamento, o uso de energia, e todas as formas de gestão política, customizado para atender os objetivos de negócios.

Assim, com as APIs entre a camada de controle e camadas de aplicações, a camada de aplicação pode operar em uma abstração da rede, aproveitando os serviços de rede e capacidades sem ficar presa aos detalhes de sua implementação. As Redes Definidas por Software tornam as redes mais personalizadas ou conscientes e as aplicações que rodam sobre ela podem ser de diversos tipos ou funções, como resultado disso, o uso dos recursos de armazenamento, de computação e de rede podem ser otimizadas.

Na Figura 2 observa-se uma análise da arquitetura atual dos roteadores e de um modelo de hardware de rede programável. O modelo atual dos roteadores trata das camadas do plano de controle e plano de dados que ficam juntas nos comutadores. O modelo de hardware para SDN na Figura mostra a separação dos dois planos.

O modelo atual dos roteadores é encarregado de tomar as decisões de rotea-

mento, transfere essas decisões para o plano de encaminhamento através de uma API proprietária. A única interação da gerência com o dispositivo ocorre através de interfaces de configuração (Web, SNMP, CLI, por exemplo), limitando o uso dos dispositivos às funcionalidades programadas pelo fabricante [Rothenberg et al. 2011].

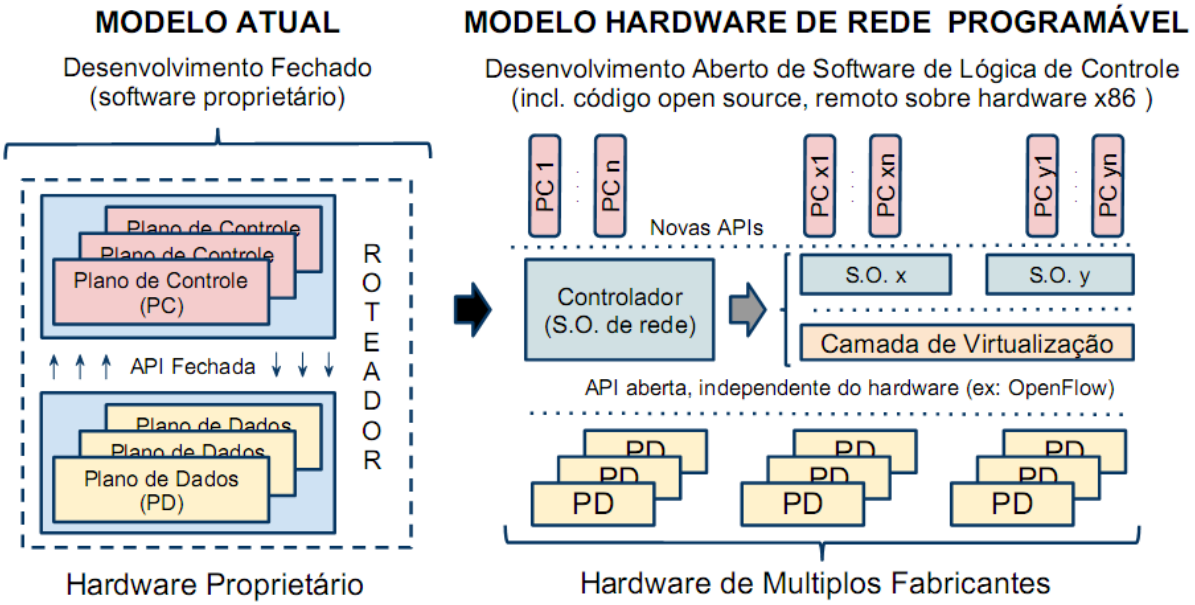


Figura 2: Arquteturas de Roteadores: Modelo Atual (Mainframe) e Modelo Programável (Openflow) [Rothenberg et al. 2011].

O modelo de hardware de rede programável é definido por duas camadas controladas, não sendo necessário que elas estejam contidas em um mesmo equipamento físico, ou seja, o software pode estar em outro componente da rede e controlando outros comutadores. A arquitetura da SDN subdivide essas camadas de forma que seja possível programar remotamente o dispositivo de rede, permitindo que a camada de controle possa ser movida para um servidor dedicado e com alta capacidade de processamento.

Sendo assim, mantém-se um alto desempenho no encaminhamento de pacotes na rede, aliado à maleabilidade de inserir, remover fluxo de dados por meio das aplicações de rede utilizando protocolos ou APIs abertos que podem ser customizados pelo administrador da rede.

## 2.4 Controlador SDN

Em Redes Definidas por Software o objetivo geral é fazer com que cada comutador que antes possuía um sistema operacional fechado de fábrica, desmembre o software do hardware, passando todo o controle e gerencia para um único dispositivo. Tendo em mente esse objetivo, faz se necessário ter um elemento central que vai funcionar como o sistema operacional da rede, controlando os comutadores nela pertencente, esse elemento é chamado de controlador.

Controladores são elementos centrais em uma rede definida por software, onde a sua tabela de fluxos é quem vai definir todo o seu funcionamento, adicionando e removendo entradas, no caso do modelo de administração e gerenciamento esses controladores serão os clientes que usarão as aplicações de redes.

Dessa forma, o controlador atua como um sistema operacional para gerenciamento e controle das redes, e oferece uma plataforma com base na reutilização de componentes e na definição de níveis de abstração. Contudo, novas aplicações de rede podem ser desenvolvidas rapidamente [Gude et al. 2008].

O controlador é responsável por tomar decisões e adicionar ou remover as entradas na tabela de fluxos, de acordo com que o administrador de rede SDN desejar. Esse componente, chamado de Controlador SDN, pode concentrar a comunicação com todos os elementos programáveis da rede oferecendo uma visão unificada. Assim é possível acrescentar aplicativos de rede que além de ter uma visão centralizada da rede, com análises detalhadas, é possível também implementar novas funcionalidades para chegar a decisões operacionais de como o sistema deve se comportar.

O controlador não gerencia a rede, as aplicações implementadas no controlador são responsáveis por esta gestão. O controlador apenas aplica as ações decididas pelas aplicações [Vasco 2013].

Essa visão unificada da rede não necessariamente precisa ser centralizada. A analogia que fazemos a sistemas operacionais distribuídos podem ser implementado perfeitamente para os controladores. A implementação pode der desenvolvida de forma distribuída, seja pela divisão dos elementos de diferentes sistemas autônomos - SA ou por um controlador realmente desenvolvido de forma distribuída com algoritmos que sejam capazes de manter uma visão consistente entre suas partes [Costa 2013].

Ao longo da definição do paradigma SDN foram criados diversos controladores para sustentar as pesquisas sobre o tema. Alguns desses controladores possuem interface gráfica para implementação na rede, outros funcionam como interface de texto, em diversas linguagens de programação e sistemas operacionais. Nesta dissertação falaremos um pouco mais sobre os principais controladores encontrados.

- Nox [Nox 2013]: É o controlador que permite o desenvolvimento na linguagem C++, e funciona sobre a plataforma Linux. O Nox trabalha sobre o conceito de fluxo de dados e verifica o primeiro pacote de cada fluxo e também determina a política a ser aplicada em cada aplicação. É um sistema operacional de rede relativamente simples e provê primitivas para o gerenciamento de eventos e funções para a comunicação com os switches.

- Pox [Pox 2013]: Segue as mesmas premissas do controlador NOX, a diferença é que o POX é escrito inteiramente em Python. Este controlador está em constante desenvolvimento e têm o objetivo de substituir o futuramente o NOX.

- Beacon [Beacon 2013]: O Beacon é um controlador baseado em Java que suporta

operações de controle baseadas em eventos e operações baseadas em *threads*. Desenvolvido pela Universidade de Stanford, tem como principais características a estabilidade, é *open source*, é capaz de construir *frameworks* em Java e pode ser executado em diversas plataformas.

- Floodlight [Floodlight 2014]: O controlador *open source* Floodlight é baseado em Java e também é multiplataforma. O Floodlight oferece um sistema de carregamento de módulos que facilitam a exportação de serviços. Sua interface permite identificar e descobrir o status e a topologia de uma rede automaticamente.

- Maestro [Maestro 2010]: O controlador Maestro foi desenvolvido em Java e tem como objetivo orquestrar aplicações de controle por meio de interfaces que acessam e modificam o estado da rede coordenando suas interações. O controlador Maestro é uma plataforma para alcançar funções de controle de rede automáticas e programáticas usando esses aplicativos modularizados.

## 2.5 Conclusões do capítulo

Este capítulo apresentou o conceito e histórico das redes definidas por software, bem como sua arquitetura e alguns dos controladores que fazem parte deste novo paradigma, além dos principais benefícios da utilização de SDN.

As redes definidas por software são apoiadas por grandes empresas como: Cisco, HP, Google, Facebook, VMware, Dell e entre outras [ONF 2014]. É um assunto que promete muitas mudanças nas redes atuais. Algumas tecnologias já são utilizadas e pesquisas sobre o novo paradigma estão sendo feitas em universidades a fim de unir o estudo acadêmico e o comercial.

# CAPÍTULO 3

# Trabalhos Relacionados

Este capítulo apresenta os principais modelos encontrados relacionados à proposta desta dissertação. As propostas listadas estão associadas a um ou mais aspectos considerados neste trabalho. Desta forma a Seção 3.1 mostra os desafios encontrados para a elaboração da pesquisa e a Seção 3.2 apresenta os modelos já existentes que foram comparados com esta proposta. Por fim a Seção 3.4 apresenta a conclusão dos trabalhos relacionados e a sua análise.

## 3.1 Desafios

O sucesso que as redes programáveis vem fazendo em ambientes de produção faz com que várias empresas e instituições criem aplicações para solucionar seus problemas e criar novas soluções a fim de melhorar o desempenho e controle de sua rede, entretanto a criação de tantas aplicações diferentes faz com que não haja um lugar para centralizar as mesmas.

Para facilitar o entendimento sobre gerenciamento, podemos dividi-lo em 5 tipos principais de gerências [Nanda and Kotz 2008]. Estes são definidas pela Organização Internacional de Padronização International Standards Organization - ISO da seguinte maneira:

- Gerência de Falhas: É responsável por detectar, registrar e responder a condições de falhas na rede;
- Gerência de Configuração: Controla a modificação das configurações de hardware e de software em uma rede;

- Gerência de Administração: Provê os meios necessários para gerenciar os usuários do sistema, suas senhas e permissões. Além disso, administra algumas operações nos equipamentos como *backups*, atualizações e sincronismos;
- Gerência de Desempenho: Durante todo o tempo que a rede esteja em funcionamento esta gerência irá monitorar, analisar, reportar e controlar o desempenho dos diferentes componentes da rede;
- Gerência de Segurança: Controla o acesso aos recursos disponibilizados pela rede;

Uma das ferramentas mais importantes para o gerenciamento de redes, são os controladores. Esses dispositivos de redes funcionam como um sistema operacional que aplica todas características de gerência na rede através das aplicações, ou seja, é o administrador da rede que vai configurar as regras e configurações nesse dispositivo.

## 3.2 Modelos Comparados

Foram utilizados como trabalhos relacionados modelos consagrados no sentido de gerência de repositório de aplicações, visto que existe uma grande dificuldade em encontrar trabalhos científicos sobre o tema proposto, cada tecnologia abaixo foi analisada e potencializada da melhor forma para o desenvolvimento do nosso método de organização e coordenação de aplicações em repositório para redes definidas por software a fim de tornar viável a sua implementação.

### 3.2.1 Google Play

O *Google Play* [Google 2014] é uma loja virtual da Google que tem como objetivo distribuir aplicativos, jogos, livros e filmes, que podem ser criados por qualquer desenvolvedor que esteja interessado. Essas aplicações estão disponíveis de gratuitamente ou a um determinado custo, o que pode alternar de acordo com cada desenvolvedor.

O Google fornece o Android SDK, onde este possui bibliotecas de API e as ferramentas necessárias para desenvolver as aplicações. O Google também fornece um pacote chamado *Android Developer Tools* - ADT que contém um conjunto de ferramentas que auxiliam o *Android SDK* na criação das aplicações.

Os passos básicos para desenvolver as aplicações para o *Google Play* são:

- Instalação: Instalar e configurar o Ambiente de desenvolvimento, é permitido criar ambientes virtuais no *Android*.
- Desenvolvimento: Nesta fase o desenvolvedor cria e desenvolver todo o projeto, contém todo o código fonte e arquivos da aplicação.

- Depuração e Teste: Durante essa fase o desenvolvedor cria e debuga o projeto, criando um arquivo *.apk*, que pode ser instalado e aplicado em um emulador do *Android*.
- Publicação: Para publicar um aplicativo no *Google Play* deve se preparar o pedido de liberação e se aprovado, liberar o aplicativo para os usuários.

A política de segurança do Google prevê que apenas aplicações em perfeito estado de funcionamento podem estar em sua loja virtual, entretanto essas aplicações não podem conter código malicioso ou infringir as leis. O controle das políticas é feito através de testes que nas aplicações prontas pelo Google e a resposta do aceite das aplicações demora em torno de um ou dois dias.

### 3.2.2 App Store

A *App Online Store* [Apple 2014] é a loja a onde estão disponíveis todas as aplicações desenvolvidas exclusivamente para o Sistema Operacional - SO fechado da Apple.

Para desenvolver aplicativos para a plataforma do IOS, a Apple disponibiliza uma SDK para os desenvolvedores criarem seus aplicativos e mandarem para a loja virtual, entretanto a empresa cobra uma licença para os usuários se tornarem desenvolvedores, uma taxa anual. Cada aplicação desenvolvida passa por um rigoroso processo de controle de qualidade no software e no hardware onde apenas aplicações com o Human Interface Guidelines - HIG que é a norma de design da Apple, são aprovadas e publicadas no site da empresa, e todo esse processo pode levar de 15 a 30 dias.

### 3.2.3 Advanced Packaging Tool

O *Advanced Packaging Tool* - APT [APT 2007] é uma ferramenta de pacotes utilizada nas distribuições Debian do linux baseadas em comandos dpkg, enquanto outros sistemas usam o rpm por exemplo. O dpkg permite que os usuários cuidem do seu próprio arquivo de configuração e escolham de quais repositórios querem receber atualizações.

Em cada sistema operacional linux baseado no Debian, há um arquivo de configuração onde nele ficam os endereços dos repositórios no qual o sistema busca a instalação e atualização dos pacotes, sendo que o usuário precisa entrar com o comando apt-get install nomedopacote e automaticamente o APT verifica se esse pacote consta na lista de repositórios.

Todos os pacotes utilizados nesse sistema são do tipo DEB. A gerência de cada pacote se da por meio de comandos utilizados por meio do APT. Os principais comandos são:

- apt-get update: Atualiza todas as listas de repositórios do arquivo de configuração.

- apt-get install nome-do-pacote: Instala um novo pacote.
- apt-get remove nome-do-pacote: Remove um pacote.
- apt-get - -purge nome-do-pacote: Remove um pacote e seus arquivos de configuração.
- apt-get upgrade: Atualiza todos os pacotes que já instalados no sistema.
- apt-cache search nome-do-pacote: Procura novos pacotes disponíveis.

### 3.2.4 Hewlett-Packard

A *Hewlett-Packard* - HP [HP 2013] está desenvolvendo uma loja de aplicativos SDN, e disponibilizando uma SDN Developer Kit, permitindo que os usuários desenvolvam aplicativos apenas para seu controlador de hardware fechado, ou seja, os aplicativos contidos na loja virtual só funcionarão no controlador da HP.

Na Figura 3 pode-se observar o controlador de redes de aplicativos virtuais da HP que oferece uma solução com o controlador da HP. O controlador da HP utiliza o OpenFlow para programar as camadas de infraestrutura. A HP ainda tem uma rede de aplicativos virtuais [HP 2014b] para as redes definidas por softwares que permite que as empresas desenvolvam aplicativos para seu controlador.

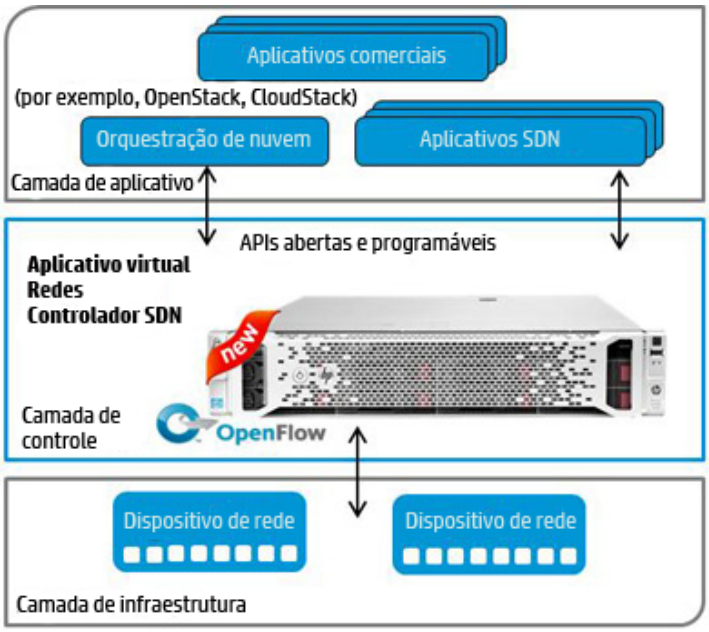


Figura 3: Controlador SDN de Redes de Aplicativos Virtuais HP[HP 2014a]

## 3.3 Análise dos Trabalhos Relacionados

Esses exemplos de tecnologias foram usados como modelo e base para desenvolver cada componente desta proposta. Na Tabela 1 apresentada a seguir, comparamos cada modelo e suas respectivas características com o nosso modelo de administração e gerenciamento de repositório de aplicações para SDN.

O Primeiro item a ser avaliado relaciona quais das tecnologias que usam o paradigma de Redes Definidas por Software atualmente. Apenas duas delas integram SDN em seu modelo, são: Hewlett-Packard e o RepoSDN. O segundo item é a utilização de aplicações estáveis, onde todos as tecnologias avaliadas possuem apenas aplicações funcionais e não em versão de testes. O terceiro item avaliado foi a utilização de vários controladores diferentes em uma mesma rede, e apenas o RepoSDN neste artigo permite essa utilização. E o quarto e último item mostra quais os tipos de licenças que cada um possui: apenas o APT e o modelo SDN proposto possuem licenças de software livre, no caso a *General Public License - GPL*, permitindo assim ser utilizado por qualquer pessoa ou entidade.

Tabela 1: Modelos Comparados no Trabalho Relacionado

| | App Store | Google Play | HP | APT | RepoSDN |
|---|---|---|---|---|---|
| Utiliza SDN | Não | Não | Sim | Não | Sim |
| Aplicações Estáveis | Sim | Sim | Sim | Sim | Sim |
| Múltiplos Controladores | Não | Não | Não | Não | Sim |
| GPL | Não | Não | Não | Sim | Sim |

## 3.4 Conclusões do Capítulo

O Capítulo de Trabalhos Relacionados mostrou que apesar da grande deficiência de trabalhos acadêmicos para a proposta, foi possível utiliza-se de modelos consagrados na Internet. Para a elaboração da proposta foram tomados em consideração os pontos mais importantes de cada modelo, aplicando assim o conceito de Redes Definidas por Software para o método de organização e coordenação de aplicações em repositório para Redes Definidas por Software.

# CAPÍTULO 4

# RepoSDN: Um Método de Organização e Coordenação de Repositório de Aplicações para Redes Definidas por Software

Este capítulo disserta sobre a proposta de um método de organização e coordenação de repositório de aplicações para redes definidas por software, chamado de RepoSDN. Esse método visa melhorar a gerência e controle de aplicações criadas para a utilização em redes programáveis. Tais aplicações SDN estão reunidas em um ambiente preparado para armazenar diversas aplicações de diferentes tipos, chamado de Repositório de Aplicações. Toda aplicação disponível nesse repositório tem como objetivo final ser executada em controladores SDN, sendo que é o administrador quem vai decidir quando e como as aplicações serão executadas.

A Seção 4.1 apresenta a introdução sobra a proposta, a Seção 4.2 disserta sobre a arquitetura do RepoSDN, a Seção 4.3 apresenta os diagramas elaborados do método RepoSDN e as Sub-seções 4.3.1, 4.3.2 e 4.3.3 apresentam o diagrama de caso de uso, diagrama de classe e diagrama de sequencia respectivamente, e por fim a seção 4.4 a conclusão da proposta.

## 4.1 Introdução

Modelos de gerenciamento utilizados nas redes comuns não são adequados às redes programáveis, uma vez que não tratam da implantação dinâmica de novos serviços. Além disso, dependendo das políticas de acesso às infraestruturas programáveis, um grande número de usuários poderão propor e implantar suas próprias aplicações de rede. Nesse cenário, harmonizar as aplicações, usuários e controladores, mantendo a confiabilidade e

escalabilidade dos serviços implantados é um problema em aberto [De Jesus et al. 2013].

O RepoSDN tem como objetivo implantar um método de organização e coordenação para que as aplicações SDN cheguem nos clientes da rede, no caso os controladores. Desta forma, o RepoSDN visa prover, para os administradores de rede uma forma de encontrar essas aplicações a fim de que existirá apenas uma fonte para as mesmas (Repositório de Aplicações), cuja função é dar para a comunidade todo um suporte e segurança no que se trata de distribuição de aplicações.

Atualmente existem muitas aplicações de rede desenvolvidas para serem utilizadas em controladores, como por exemplo o *Routing Information Protocol Next Generation* - RIPng[RFC-2080 1997], *Open Shortest Path First* - OSPF[RFC-2328 1998], entre outras. Entretanto, não há uma forma de como gerenciá-las, esta dissertação propõe uma solução para este problema, um método de organização e coordenação de repositório de aplicações para Redes Definidas por Software, RepoSDN, onde todos os tipos de aplicações de rede ficarão alocadas em um repositório, a disposição de qualquer SDN, institucional ou comercial, onde diversos usuários terão a oportunidade de armazenar e utilizar de aplicações de rede.

É importante pensar na ideia de criar um repositório exclusivo para armazenar as aplicações SDN, pois criar aplicações e armazenar em repositórios já existentes, como o APT, YUM ou OPKG, não permitiria a interoperabilidade entre as aplicações, ou seja, seria necessário criar várias versões da mesma aplicação para cada um dos sistemas operacionais existentes. Atualmente as aplicações criadas são dependentes do sistema operacional, diferentemente das aplicações SDN, que são dependentes do controlador da rede.

## 4.2 Arquitetura do RepoSDN

As redes definidas por software possibilitam a implementação de aplicações de rede que realizam lógicas de monitoração e acompanhamento de tráfego mais sofisticado e mais completo que os atuais [De Araújo 2013].

O Fluxo de aplicações é todo o processo que as aplicações levam para percorrer seu caminho até o seu destino. Este fluxo torna o método de organização eficiente, e alinhando todo o processo e otimizando os resultados através da melhoria do meio com que as aplicações vão seguir até chegar aos controladores nas redes programáveis, diferentemente dos repositórios comuns que por sua vez possuem como função atual apenas armazenar código e aplicações gratuitas ou pagas.

O método proposto reúne todas as aplicações para redes definidas por software, padronizando e definindo quais aplicações o controlador poderá utilizar, permitindo uma maior confiabilidade dessas aplicações, uma vez que haverá apenas uma fonte de aplicações, facilitando a organização, especificação e atualização das mesmas, fazendo com que as aplicações SDN saiam do desenvolvedor e cheguem até os controladores.

Utilizar o método de organização e coordenação de repositório de aplicações para Redes Definidas por Software, RepoSDN, representa ter o que há de mais novo e moderno em termos de aplicações para controladores, aplicações essas que são inteiramente funcionais e não em fases de testes.

Para armazenar e controlar essas aplicações foi desenvolvido um repositório de aplicações, onde qualquer usuário previamente cadastrado pode enviar a sua aplicação e disponibilizá-la para outros usuários. É importante ressaltar que este repositório de aplicações não é um repositório de códigos como o Git Hub [Github 2014].

Todas as aplicações enviadas para o repositório de aplicações, devem ser desenvolvidas para o novo paradigma de Rede Definidas por Softwares. Essas aplicações serão utilizadas por controladores de redes, logo este repositório fornecerá diversos softwares para vários tipos de controladores existentes, seja a arquitetura aberta ou fechada.

Todo esse processo tem início no desenvolvedor que envia a aplicação que criou para o repositório de aplicações, que por sua vez é gerenciada pelo Administrador das Aplicações (ADM APP) onde através das políticas de uso do repositório apresentada no capítulo 5, vai decidir quais aplicações são aceitas ou rejeitadas.

O Administrador da rede definida por software (ADM SDN) tem como função baixar as aplicações armazenadas no repositório e inserir ou não nos controladores da sua rede programável, ou seja, ele tem a opção de baixar e instalar as aplicações diretamente em sua rede ou apenas baixar e só utilizar no momento que achar mais adequado, pode se visualizar melhor essa arquitetura através da Figura 4.

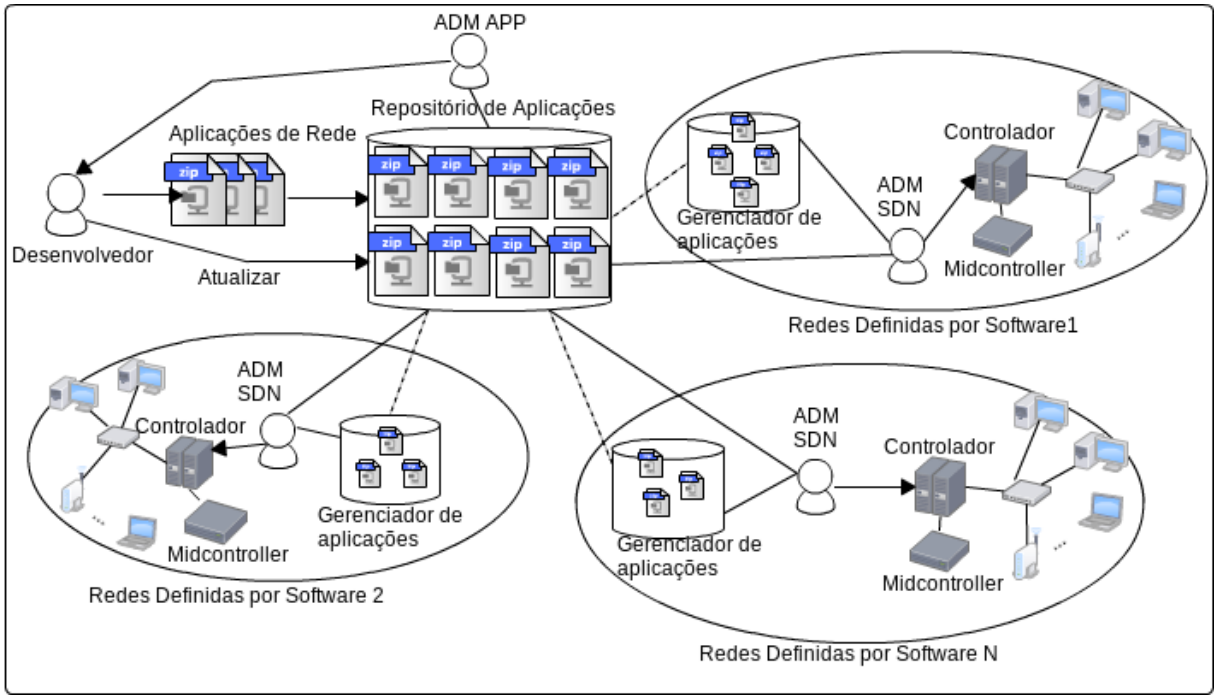


Figura 4: Fluxo de Trabalho do RepoSDN

O fluxo de trabalho das aplicações apresenta todo o caminho que as aplicações devem fazer para chegar nos controladores. Faz se necessário compreender que qualquer

Rede Definida por Software deve ser capaz de utilizar o Repositório de Aplicações como única fonte de aplicações SDN. O Administrador da Rede Definida por Software tem a opção de executar uma ou mais aplicações em seus controladores ou se desejar, apenas baixar as aplicações e armazena-las localmente.

O gerenciador de aplicações é importante nessa arquitetura, pois é ele que armazena localmente as aplicações baixadas do repositório de aplicações. O gerenciador das aplicações solicita as atualizações para as aplicações já armazenadas, deixando como opção para o administrador atualizar. O controlador não tem acesso ao repositório de aplicações, pois por medidas de segurança, o controlador não têm acesso a internet, evitando assim possíveis problemas.

A arquitetura dos componentes do RepoSDN apresentada abaixo, mostra como fluxo de trabalho do RepoSDN foi pensando e projetado, podemos visualizar o repositório de aplicações e todos os seus componentes, interagindo com o gerenciador de aplicações e controlador.

Para viabilizar todo o processo que permite que a aplicação se inicie no desenvolvedor e chegue até o cliente (controlador), foi desenvolvido um Middleware chamado de Midcontroller, que integra e automatiza as funcionalidades do gerenciador de aplicações com o controlador.

Para melhorar a visualização do Midcontroller é necessário expandir a arquitetura como apresentado na Figura 5. A descrição de cada componente é apresentada a seguir:

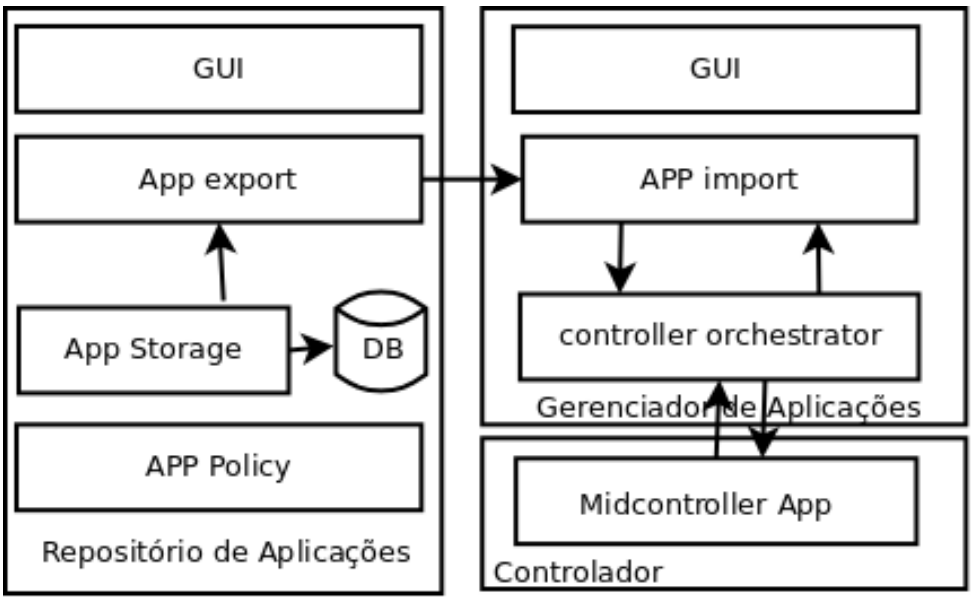


Figura 5: Arquitetura do Componentes do RepoSDN

Repositório de Aplicações: O repositório de aplicações tem como objetivo armazenar todas as aplicações de rede que os desenvolvedores irão criar e enviar para este local, as aplicações de rede podem ser algoritmos de roteamentos (OSPF, RIP), de congestionamentos (Spanning tree), Routeflow [RouteFlow 2013] e tantos outros.

- *GUI*: Interface gráfica para que os desenvolvedores possam interagir com o repositório de aplicações enviando suas aplicações para o armazenar no repositório.
- *App Export*: Responsável por disponibilizar as informações das aplicações, tais como nome, versão, dependências, que se encontram no banco de dados do repositório para que o *app import* faça as requisições via http e esse escolha quais das aplicações são enviadas para o app manager.
- *App Policy*: Responsável por verificar e executar as políticas de segurança do repositório para com as aplicações, tais como verificar se há código malicioso.
- *App Storage*: Responsável pela leitura e escrita das informações no banco de dados MySQL.

Gerenciador de Aplicações: O gerenciador de aplicações controla os softwares que têm origem do repositório de aplicações. Este gerenciador possui módulos que darão suporte para o administrador da rede programável. Cada módulo permite que o administrador tenha mais acesso a todas informações da sua rede. Portanto pode permitir uma maior autonomia da rede e de seus comutadores.

- *GUI*: Interface gráfica que administrador SDN utilizará para interagir com o *App Manager* e o *App Repository*.
- *App Import*: Responsável por consultar e requisitar as aplicações via http, importando-as para o *App Manager*, pois ele utilizará das informações que o *App Export* fornece, podendo dar o devido destino para essas aplicações.
- *Controller Orchestrator*: Responsável por coletar e administrar as informações das aplicações de rede que o *App Manager* coletou através do *APP Import* para os controlares das redes definidas por software utilizarem. Define em que controladores as aplicações são instaladas, atualizadas ou removidas.

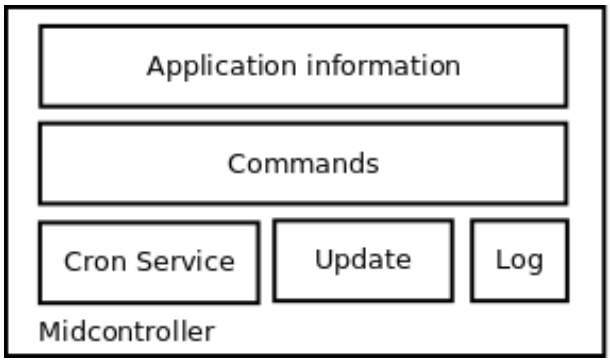


Figura 6: Midcontroller

*Midcontroller*: É apresentado na Figura 6, responsável por fazer a mediação entre o controlador e o *App Manager*. Este componente é dividido em 5 camadas, e suas funcionalidades são descritas a seguir:

- *Application Information*: Responsável por guardar as informações das aplicações que estão no App Manager, ex; nome da aplicação, versão, nome do controlador, versão do controlador, dependências, arquitetura, descrição.
- *Cron Service*: Permite que o administrador SDN configure tarefas agendadas regularmente que operam em horários definidos ou em intervalos regulares.
- *Commands*: Recebe e executa os comandos implementados para a comunicação entre o *Midcontroller* e o *Controller Orchestrator*.
- *Log*: Guarda um registro de todas as operações executadas do Midcontroller para o App Manager. Bem como um histórico de todas as aplicações instaladas e desinstaladas.
- *Update*: Realiza consultas para o Controller Orchestrator a fim de verificar se as aplicações instaladas possuem atualizações a serem feitas.

A distribuição de aplicações se dá por meio da utilização de todos os componentes criados para melhorar o processo de gerência de aplicações SDN. As aplicações são compartilhadas quando o administrador da rede programável escolhe a aplicação que ele já baixou do repositório e vai utilizar em sua rede, podendo selecionar diferentes controladores para a execução da mesma.

## 4.3 Detalhamento da Arquitetura do RepoSDN

Na seção a seguir são apresentados os diagramas UML - Unified Modeling Language. São eles, o diagrama de caso de uso, diagrama de classe e os diagramas de sequência do RepoSDN, e têm como função auxiliar a visualizar o desenho e a comunicação de todo o sistema e cada posição dos usuários e suas respectivas funções.

### 4.3.1 Diagrama de Caso de Uso

A Figura 7 apresenta o diagrama de caso de uso do RepoSDN, onde o desenvolvedor faz o envio das aplicações e também pode atualizar as aplicações criadas, como o desenvolvedor também é um usuário, este pode baixar qualquer aplicação.

O administrador das aplicações pode ou não validar as aplicações, caso o administrador não valide, a aplicação será apagada, ele também tem a opção de baixar após uma análise das políticas de envio dessas aplicações.

O administrador da SDN é responsável por utilizar as aplicações que o administrador das aplicações validou, adquirindo-as com o *baixar_aplicação* e utilizando-as com o *instalar_aplicação*.

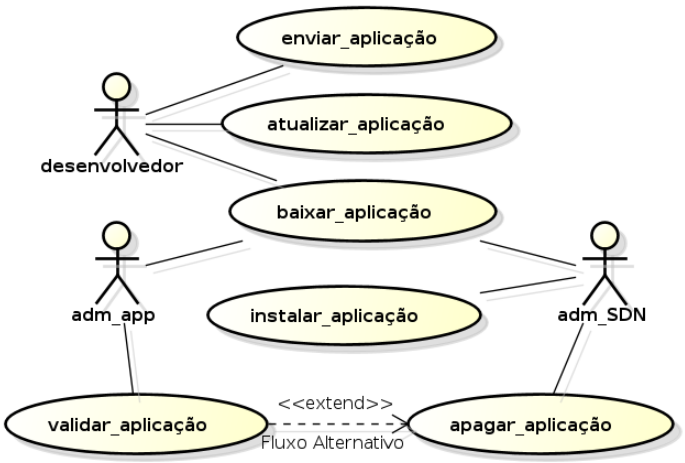


Figura 7: Caso de Uso do RepoSDN

### 4.3.2 Diagrama de Classe

O diagrama de classe do RepoSDN é representado pela Figura 8, onde o conjunto de classes específica de uma forma mais detalhada todo o processo das aplicações na estratégia. A classe Usuário é uma extensão da classe Perfil, ou seja, todos os usuários terão os mesmos atributos, variando o perfil, que tem como atributos id, tipo_perfil e descrição.

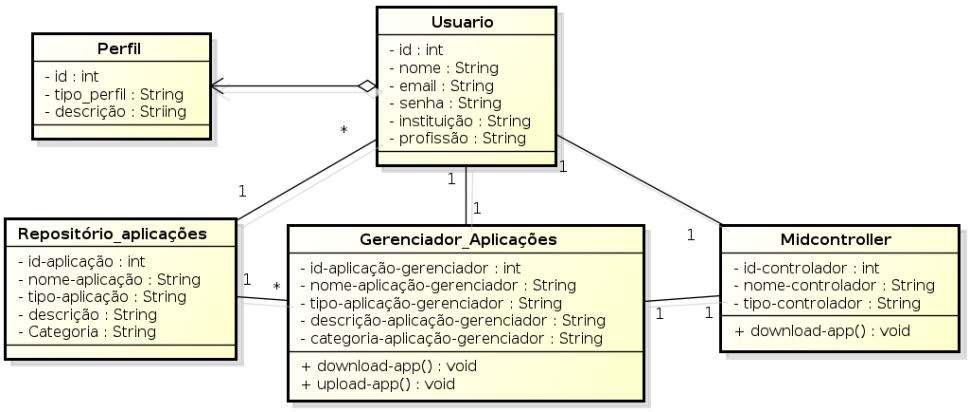


Figura 8: Diagrama de Classe do RepoSDN

A classe *Repositório_Aplicações* tem como função armazenar todos as aplicações enviados pelo desenvolvedor, ela tem como atributos o *id-aplicação* que identifica as aplicações no banco de dados, o atributo *nome-aplicação* que mostra o nome das aplicações, *tipo-aplicação* que mostra o tipo das aplicações, o atributo *descrição* tem como função descrever essas aplicações e o atributo *categoria* separa cada aplicação por categoria.

A classe *Gerenciador_Aplicações* armazena aplicações escolhidas para serem utilizadas na rede programável e tem como atributos o *id-aplicação-gerenciador* que identifica as aplicações no banco de dados, o atributo *nome-aplicação-gerenciador* que mostra o nome das aplicações, *tipo-aplicação-gerenciador* mostra o tipo das aplicações armazenadas no gerenciador, *descrição-gerenciador-aplicação* onde tem como função descrever essas

aplicações e a *categoria-aplicação-gerenciador* que separa cada aplicação por categoria.

A classe *Midcontroller* define como será o controlador do sistema, identificando com um *id-controlador*, *nome-controlador* e *tipo-controlador*, permitindo que o administrador das redes definidas por software utilize as aplicações nesse controlador da melhor forma possível.

### 4.3.3 Diagramas de Sequência

A Figura 9 é representada pelo diagrama de sequência do desenvolvedor que mostra a sequencia de processos da estratégia de distribuição, iniciada no desenvolvedor que tem como função criar e enviar a aplicação para o Repositório de Aplicações, permitindo a atualização da aplicação caso ele altere a mesma, e por fim a exclusão da aplicação caso o desenvolvedor decida por essa opção.

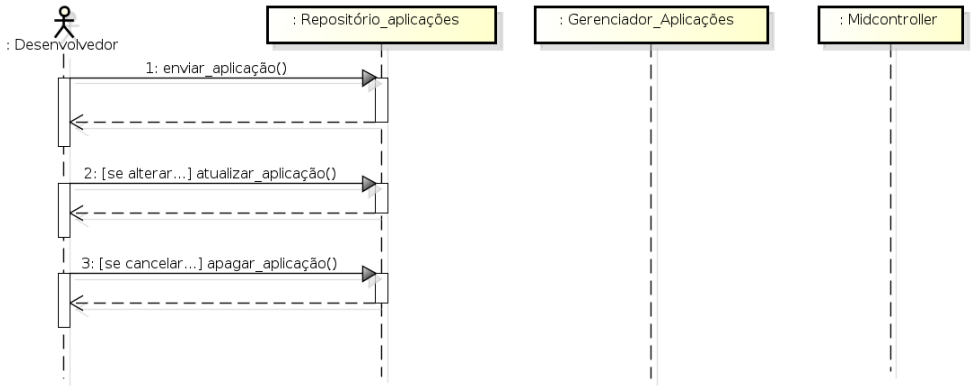


Figura 9: Diagrama de Sequência do Desenvolvedor do RepoSDN

Na Figura 10 administrador de aplicações (adm_app) tem como função analisar cada aplicação enviada para o Repositório de aplicações podendo validar ou recusar (apagar_aplicação) a aplicação seguindo as políticas de envio de aplicações que os desenvolvedores terão acesso. O adm_app não terá acesso ao gerenciador de aplicações e ao Midcontroller.

Na Figura 11 do diagrama de sequência, o administrador SDN (*adm_sdn*) faz o download das aplicações do Repositório de Aplicações e faz o *enviar_aplicação* para o gerenciador de aplicações na sua rede programável permitindo com que o administrador escolha se quer só armazenar ou instalar a aplicação diretamente no controlador armazenando também no gerenciador de aplicações, pois ele pode utilizar diferentes aplicações em controladores distintos com o *instalar_aplicação*. É um elemento importante de todo o processo, pois ele tem contato com todos os elementos da estratégia de distribuição.

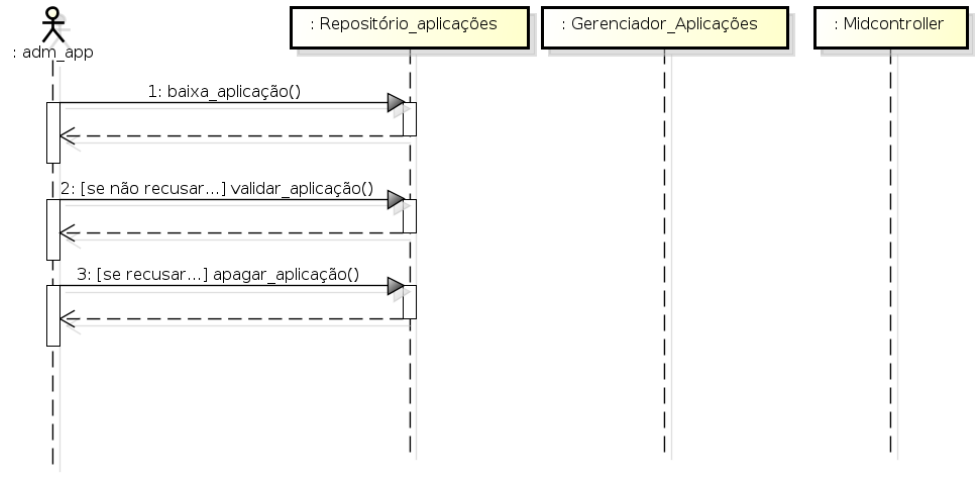


Figura 10: Diagrama de Sequência do Administrador das Aplicações do RepoSDN

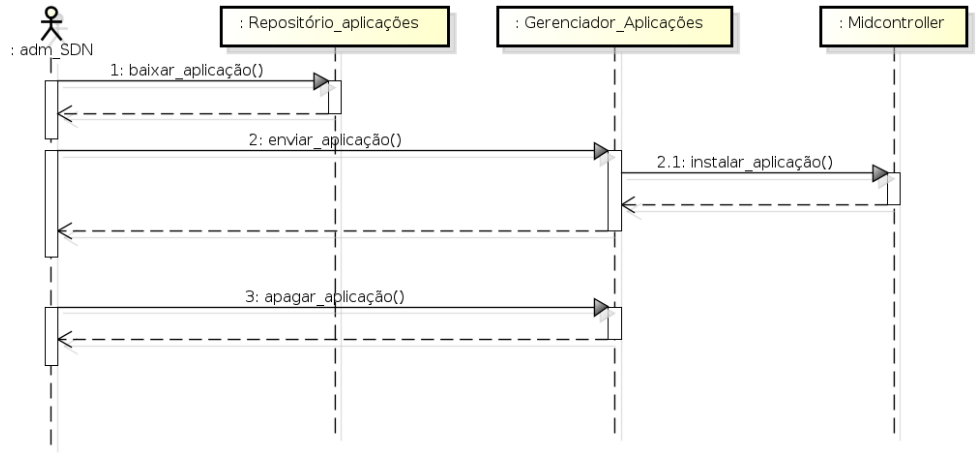


Figura 11: Diagrama de Sequência do Administrador da SDN do RepoSDN

## 4.4 Conclusões do Capítulo

Este capítulo apresentou toda a arquitetura do RepoSDN, a arquitetura dos componentes e todos os diagramas do método proposto. É importante ressaltar a ideia central do repositório de aplicações, que é a de evitar a proliferação de aplicações SDN em outros repositórios de códigos. Faz necessário reunir as aplicações em um lugar para que torne a fonte de aplicações confiável e oficial, incentivando o desenvolvimento de cada vez mais aplicações SDN.

# CAPÍTULO 5

# Políticas de Uso do Repositório de Aplicações

Neste capítulo são apresentadas as políticas de uso do Repositório de Aplicações. Na Seção 5.1 apresentamos uma introdução das políticas de uso. Na seção 5.2 é explanado os tipos de usuários que podem utilizar o repositório de aplicações. Na Seção 5.3 é mostrado como devem ser as aplicações que serão armazenadas no repositório. Na Seção 5.4 é apresentado um modelo de como devem ser o conteúdo das aplicações e as regras para que elas sejam armazenadas no repositório. Na Seção 5.5 é apresentada o critério de qualidade das aplicações, e o método que os desenvolvedores devem seguir para criar as aplicações. Na Seção 5.6 é apresentado os tipos de licenças que os desenvolvedores devem seguir para o desenvolvimento de suas aplicações. Na Seção 5.7 mostramos a conclusão do capítulo.

## 5.1 Introdução

O repositório de aplicações é um site disponível ao público no qual os desenvolvedores de aplicações para redes de computadores podem ofertar aplicações para diversos controladores, *switchs* entre outros comutadores. Para ofertar aplicações no RepoSDN, é necessário adquirir e manter uma conta de desenvolvedor válida.

A *Open Network Foundation* - ONF [ONF 2014] propõe a criação de um grupo de desenvolvimento em colaboração com membros de diferentes organizações, as quais detém grande parte do mercado de aplicações para redes e telecomunicações, dentre estas organizações estão as americanas HP e Microsoft, e também a chinesa Huawei.

A ONF teve como ideia inicial a criação de um conselho que escolhe como será o desenvolvimento de APIs objetivando a padronização e uniformização para o desenvolvimento de aplicações de código aberto ou proprietárias, para os diversos tipos de contro-

ladores SDN que serão escolhidos pelo grupo, e a estas APIs o conselho as denomina de Northbound API Interfaces - NBIs

Enquanto a ONF e o conselho não padronizam como as aplicações devem ser desenvolvidas para o novo paradigma de software, o RepoSDN cita como devem ser as aplicações que serão enviadas para o repositório de aplicações, faz se necessário criar regras para que haja uma homogeneidade no envio de cada uma. As seções abaixo é descrito como cada parte deve se comportar para obter o máximo de sucesso na utilização do repositório.

## 5.2 Dos Usuários

Usuários Desenvolvedores: São aqueles usuários predefinidos na hora do cadastro como pessoas que irão enviar seus programas para o repositório de aplicações, a fim de que os mesmos sejam compartilhados com os demais usuários. Esses usuários receberão uma confirmação para que possam hospedar as suas aplicações no RepoSDN. Se enquadram também na categoria de usuários comuns.

Usuários Comuns: São os usuários cadastrados que podem apenas fazer download de qualquer aplicativo no RepoSDN, sem alterar ou excluir as aplicações do repositório.

Administrador das Aplicações: É o usuário com privilégios para excluir ou aceitar as aplicações no repositório, este usuário tem total acesso a qualquer aplicação armazenada no repositório, podendo excluir sem aviso prévio qualquer aplicação que interfira nas políticas de uso mencionadas no Capítulo 5, Seção 5.3 e 5.4.

## 5.3 Das Aplicações

Todas as aplicações enviadas para o repositório, devem ser desenvolvidas para o novo paradigma de Rede Definida por Software. Essas aplicações serão utilizadas por controladores de redes, logo este repositório de aplicações fornecerá diversos tipos de softwares para qualquer controlador existente, mesmo sendo pago ou gratuito.

Todas as aplicações enviadas para o repositório passarão por uma avaliação e sessão de testes antes de serem aprovadas e inseridas no repositório.

O tempo de avaliação e testes é indeterminado, pois como apresentado na Seção 5 desta política de uso, o conteúdo de cada um das aplicações é avaliado para que não possam prejudicar nenhum dos controladores da redes definidas por software.

O repositório apenas aceitará aplicações que tenham uma versão estável, ou seja, aplicações funcionais e que não sejam beta ou em fase de testes.

O repositório de aplicações não se importa com a linguagem do aplicativo, deste que este seja plenamente funcional e siga as regras da Seção 5.5.

Toda e qualquer aplicação deverá seguir os critérios de qualidade das aplicações descrito na seção 5.4.

## 5.4 Conteúdo

As Políticas de conteúdo se aplicam a qualquer conteúdo que o aplicativo mostre ou ao qual esteja vinculado, incluindo qualquer tipo de anúncio exibido aos usuários e qualquer conteúdo gerado por usuários que o aplicativo hospede ou aos quais se vincule. Da mesma forma, essas políticas se aplicam a qualquer conteúdo da conta escolhida pelo usuário que for exibida publicamente no repositório de aplicações. Em seguida é apresentada as políticas de conteúdo das aplicações:

- Falsidade Ideológica ou Comportamento Enganoso: Não finja ser outra pessoa nem que o seu aplicativo é produzido ou distribuído por outra empresa ou entidade. Os arquivos contidos neles também não podem imitar a funcionalidade nem os avisos ou outras aplicações.

  As aplicações não podem conter informações falsas ou enganosas em seu conteúdo ou componentes. Os Desenvolvedores não podem mentir para os usuários nem fornecer informações para sites que simulem ou se façam passar por outro aplicativo, organização ou pessoa.

- Propriedade intelectual: Os direitos de propriedade intelectual são guardados pela política de uso deste repositório de aplicações, é proibido a violação de direitos de pessoas, patente, marca registrada, segredo comercial, direitos autorais e outros direitos de propriedade, muito menos encorajar ou induzir a violação de direitos de propriedade intelectual por terceiros.

- Informações pessoais e confidenciais: É proibida a publicação ou divulgação não autorizada de dados pessoais e confidenciais dos usuários e ou aplicações de empresas ou organizações, como código fonte, arquivos de configuração, documentos pessoais.

- Atividades ilegais: As aplicações devem ser criadas dentro da lei. Em caso de desobediência das políticas de uso ou leis, como a venda de aplicações de terceiros ou apropriação indevida dos mesmos, caso isso aconteça, as autoridades legais serão comunicadas.

- Produtos maliciosos: É proibida a propagação de conteúdo que prejudique, interfira na operação ou acesse, de maneira não autorizada, redes, servidores ou outra infraestrutura sem o consentimento dos usuários.

  É proibido a transmissão de vírus, worms, defeitos, cavalos de Troia, malware ou qualquer outro item que possa criar vulnerabilidades de segurança ou danificar os controladores e comutadores, aplicações ou dados pessoais dos usuários, bem como scripts maliciosos e golpes de phishing de senha também são proibidos nas aplicações

no repositório, assim como aplicações que levam o usuário a fazer o download ou a instalação involuntários de aplicações de fontes fora do repositório involuntariamente.

- Interferência do sistema: Um aplicativo transferido do repositório de aplicações (ou seus componentes ou elementos derivados) não podem fazer alterações ao dispositivo do usuário fora do aplicativo sem o conhecimento e o consentimento do desenvolvedor.

  Isso inclui comportamentos como a substituição ou a reorganização da apresentação padrão de aplicações. Se um aplicativo fizer essas alterações com o conhecimento e o consentimento do usuário, será necessário que a informação sobre qual aplicativo fez a modificação seja clara para o usuário. Além disso, será preciso que essa alteração possa ser facilmente revertida ou desinstalada do aplicativo pelo usuário.

  As aplicações não podem estimular, incentivar ou enganar os usuários para que removam ou desativem aplicações de terceiros, exceto se isso fizer parte de um requisito indispensável de um serviço de segurança fornecido pelo aplicativo.

## 5.5 Critérios de Qualidade das Aplicações

A Qualidade das aplicações influencia diretamente no sucesso a longo prazo de seu aplicativo em termos de instalações, avaliação do usuário e comentários. Os Administradores das redes definidas por software esperam aplicações de alta qualidade para utilizarem nos controladores de sua rede.

Esta seção ajuda a avaliar os aspectos básicos de qualidade nas aplicações através de um conjunto compacto de critérios de qualidade das aplicações básicas e os testes associados. Todos as aplicações para redes definidas por software publicados no repositório de aplicações devem atender a esses critérios.

Antes de enviar o aplicativo, faz se necessário um teste contra estes critérios para garantir que ele funcione bem em muitos controladores e comutadores presentes nas redes definidas por software, atenda aos padrões do repositório de aplicações, o teste vai bem além do que é descrito neste capítulo, o objetivo desta seção é especificar as características essenciais de qualidade básica para que seja incluso nos planos de testes.

- Não importa a linguagem de programação que o desenvolvedor escolha. O importante é que essas aplicações estejam compactadas no formato .zip, haja vista que esses arquivos .zip serão incluídos no repositório de aplicações para que todos os Administradores das aplicações interessados baixem para o seu Gerenciador de Aplicações local.

- Para aplicações de rede de computadores que usem linguagem de programação interpretada, é preciso que quando o usuário execute o comando para instalar, au-

tomaticamente compile e instale o aplicativo, da mesma forma para atualizar e desinstalar.

- Todo aplicativo .zip possui uma estrutura padrão. Logo cada aplicativo deve obedecer essa estrutura e consequentemente obedecer a criação de scripts para a execução dos mesmos.

- Criar um aplicativo para ser utilizado no RepoSDN é um procedimento simples que requer a criação de alguns diretórios e arquivos especiais. O primeiro passo a ser tomado é criar ou separar um diretório totalmente vazio para abrigar os conteúdos das aplicações. É importante considerar este diretório como sendo o diretório raiz (/) do sistema.

- Caso o conteúdo do aplicativo seja alguma documentação ou script não-compilado, todos os arquivos devem estar dentro do diretório raiz imitando o lugar onde ele deveria ocupar no sistema, pois se o usuário quiser empacotar um script que, em um sistema de verdade, deva estar em /usr/bin/script.sh, basta criar dentro de /home/usuario/meuaplicativosdn/, um diretório chamado usr e, dentro desta, outro chamado bin. Após, portanto, é só copiar o script para este último diretório, sendo que o caminho dos diretórios deverá ficar assim: /home/usuario/meuaplicativosdn/usr/bin/meuscript.

- Se o aplicativo criado pelo usuário seja um binário, a melhor alternativa é compilá-lo e instalá-lo no diretório raiz. A maioria dos programas para GNU/Linux pode ser compiladas com um conjunto de instruções ./configure, make e make install, entretanto o make install geralmente instala o programa no diretório raiz (/) do sistema, sendo assim, faz se necessário dizer que o aplicativo é instalado no diretório criado para o mesmo.

```
./configure --prefix=/home/usuario/meuaplicativosdn
make
make install
```

- Dependendo do programa, compilar o aplicativo diretamente em um diretório pode ser feito de outra forma, como:

```
./configure
make
make install DESTDIR=/home/usuario/meuaplicativosdn
```

- O Nome do aplicativo deverá ser escrito assim:

  <nome do aplicativo>-<versão do pacote>-<arquitetura>.zip

  por exemplo: aplicativosdn-1.0-i386.zip

- Todas as aplicações enviadas para o repositório devem estar compactadas no formato ZIP, e junto com os arquivos do software, porém no diretório raiz, deve existir um

arquivo de configuração que conterá as informações das aplicações criadas por cada desenvolvedor, o nome do arquivo de texto deve ser *app.inf*, as linhas contidas nos arquivos devem seguir essa ordem:

- NAME_APP: Essa linha deve conter o nome da aplicação.
- VERSION_APP: Essa linha deve informar a versão da aplicação.
- APP_TYPE: O tipo da aplicação.
- MAINTAINER: Nome e e-mail do criador da aplicação.
- DEPENDENCES_APP: Se essa aplicação contém dependências necessárias para a instalação dos pacotes.
- NAME_CONTROLLER: O nome do controlador a qual a aplicação deve ser instalada.
- VERSION_CONTROLLER: A versão do controlador que a aplicação foi testada.
- ARCHITECTURE: Específica qual tipo arquitetura essa aplicação foi projetada, ex: i386, amd64 e caso o aplicativo seja para todas as arquiteturas, basta colocar a opção *all*.
- COPYRIGHT: Qual o tipo de licença tem a aplicação.
- DESCRIPTION: Descrição da aplicação.

Esse arquivo é muito importante pois é através dele que o administrador SDN vai saber para qual controlador a aplicação deve ser executada, e utilizando os componentes criados, o administrador vai facilitar a gestão das diversas aplicações sendo executadas na rede.

## 5.6 Direitos Autorais

Esta seção apresenta os tipos de licenças e suas diferenças para as diversas aplicações que são armazenadas no repositório de aplicações, é importante ressaltar que toda aplicação enviada para o repositório será compartilhada com todos os usuários.

Código Aberto: Um software que é dito como código aberto ou *open source*, é um software cujo código fonte é visível a todos, ou seja, publicamente. O software de código aberto deve respeitar as quatro liberdades definidas pela *Free Software Foundation* - FSF[1]. Porém, não estabelece certas restrições como as contidas na *General Public License* - GPL. É protegido pela Iniciativa do Código Aberto a *Open Source Initiative* - OSI[2].

A Licença Pública Geral, GNU ou GPL, é a definição da licença para software livre criada por Richard Stallman[3] no final da década de 1980, e é utilizada pela maioria

[1]https://www.fsf.org/pt-br
[2]http://opensource.org/
[3]http://stallman.org/

dos programas do GNU, assim como muitos outros programas open source que não fazem parte do projeto GNU. A GPL é a licença com maior utilização por parte dos projetos de software livre, em grande parte devido à sua adoção para o Linux.

Software comercial é um software desenvolvido por uma empresa ou instituição com o objetivo de obter lucros com sua utilização. É importante ressaltar que comercial e proprietário não são o mesma coisa no âmbito de softwares. A maioria dos softwares comerciais são proprietários, mas também existem softwares livres que são comerciais, e existem softwares não-livres não-comerciais.

Software proprietário é aquele na qual a cópia, redistribuição ou modificação são de alguma forma proibidas pelo seu desenvolvedor, dono ou distribuidor. A expressão vai em oposição à ideia de software livre. Quando se quer redistribuir ou utilizar o código fonte do software, faz se necessário solicitar ou adquirir uma licença.

Software Livre: Muitas pessoas confundem o conceito de software livre com software grátis, é uma questão de liberdade, não de preço. Para entender o conceito, deve-se pensar em "liberdade de expressão". Pode ser relacionado à liberdade com que os usuários executam, copiam, distribuem, estudam, modificam e aperfeiçoam o software. Detalhando mais esse conceito, se refere a quatro tipos de liberdade, esses são descritos sucintamente a seguir:

- Liberdade Nº 0: A liberdade de executar o programa, para qualquer propósito.
- Liberdade Nº 1: A liberdade de estudar como o programa funciona, e adapta-lo para as suas necessidades. Aceso ao código-fonte é um pré-requisito para esta liberdade.
- Liberdade Nº 2: A liberdade de redistribuir cópias de modo que possa ajudar a outros usuários.
- Liberdade Nº 3: A liberdade de aperfeiçoar o programa, e liberar os seus aperfeiçoamentos, de modo que toda a comunidade se beneficie. Acesso ao código-fonte é um pré-requisito para esta liberdade.

Um aplicativo é dito software livre se os usuários possuí as quatro liberdades descritas. Portanto, os usuários devem ser livres para redistribuir cópias, com ou sem alterações, seja gratuitamente ou arrecadando uma taxa pela redistribuição, para qualquer usuário em qualquer lugar. Ter liberdade para fazer essas coisas significa que não é necessário ter que pedir ou pagar pela permissão ou utilização das aplicações.

Um dos conceitos de liberdade também é poder fazer modificações nas aplicações e usa-las no trabalho ou de forma particular, sem nem mesmo mencionar que elas existem. Se modificações são publicadas, não há a necessidade de comunicar pessoas ou empresas. De modo que a liberdade de fazer modificações, e de publicar versões aperfeiçoadas, seja significativa, tendo acesso ao código-fonte do programa. Portanto, acesso ao código-fonte é uma condição necessária ao software livre.

Entretanto, existem certos tipos de regras sobre a maneira de distribuir software livre que são aceitáveis, quando elas não entram em conflito com as liberdades principais. Por exemplo, a chamada copyleft, termo usado para sugerir que é permitido efetuar a cópia de um software já alterado. É a regra de que quando redistribuindo um programa, não será possível adicionar restrições para negar à outras pessoas as liberdades principais.

A copyleft não entra em conflito com as quatro liberdades, na verdade, ela as protege. Regras sobre como unificar uma versão modificada são aceitáveis, se elas não acabam bloqueando a sua liberdade de liberar versões modificadas.

O repositório de aplicações não coleta informações que não sejam voluntárias, Faz-se necessário o compartilhamento das informações para que possamos investigar, prevenir ou tomar medidas cabíveis quanto a atividades ilegais, fraude suspeita e situações que envolvam tanto o termo de serviço quanto a política de conteúdo das aplicações.

## 5.7 Conclusões do Capítulo

Este capítulo apresentou as políticas de uso do repositório de aplicações SDN, tais políticas são muito importantes pois definem como funciona todo o processo para que as aplicações possam ser utilizadas pelos clientes.

É válido ressaltar que essas políticas de uso definem uma forma segura para que todos os usuários possam interagir com o repositório e baixar as aplicações para as suas redes programáveis.

# CAPÍTULO 6

# Aplicabilidade do RepoSDN

Este Capítulo apresenta todos os meios utilizados para a validação do método de organização e coordenação de aplicações em repositório para Redes Definidas por Software, RepoSDN. Na Seção 6.1 é apresentada uma introdução dos testes realizados, e a forma com que se deu o desenvolvimento das aplicações para o auxílio do RepoSDN, bem como uma explanação da utilização dos componentes. Na Seção 6.2 exibimos uma introdução dos experimentos e testes realizados. Na Seção 6.2.1 apresentamos o primeiro teste realizado, chamado de teste de carga. Na Seção 6.2.2 mostramos como foram realizados o segundo teste, chamado de teste de desempenho, este teste se deu no repositório de aplicações. Na Seção 6.3 é apresentado a conclusão do capítulo.

## 6.1 Introdução

A utilização do método proposto automatiza o uso das aplicações para o novo paradigma de redes programáveis, onde qualquer aplicação desenvolvida poderá atender aos requisitos estipulados por esse modelo, onde as SDN que se utilizarem desse modelo, apenas precisarão configurar os parâmetros do Midcontroller e de seus coletores de acordo com a sua necessidade.

O método proposto visa o uso de qualquer tipo de controlador previamente configurado para a instalação e adaptação das aplicações SDN na rede, nesta dissertação foram executadas e instaladas as aplicações nos controladores Floodlight[1] e o Nox[2], onde as APIs auxiliam o método de instalação, atualização e remoção das aplicações nos controladores.

O ambiente visualizado na Figura 12 demonstra o Midcontroller instalado em um

[1]Floodlight: http://www.projectfloodlight.org/floodlight/
[2]Nox: http://www.noxrepo.org/

servidor e a utilização de controladores Nox, foram implementados também algumas APIs para automatizar a instalação dos aplicativos de rede nos controladores.

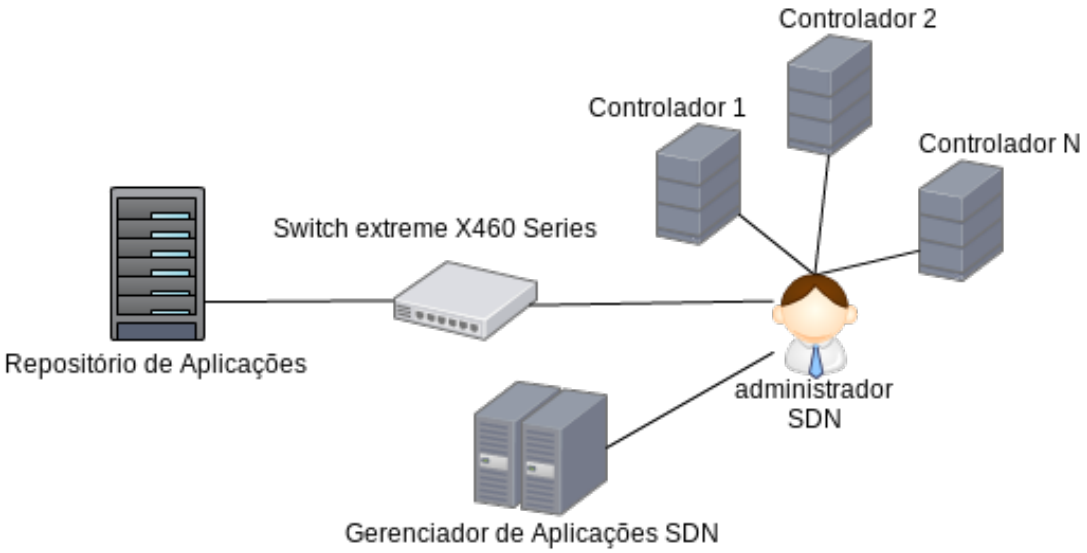


Figura 12: Ambiente de Aplicabilidade do Modelo

O Repositório de Aplicações está instalado e configurado num servidor linux Ubuntu 12.10 lts-x86, onde os desenvolvedores cadastrados podem fazer o upload das aplicações via web, preenchendo um formulário solicitando a aprovação para o Administrador das Aplicações. Todas as aplicações de rede enviadas para o Repositório de Aplicações deverão seguir as especificações apresentadas no capítulo 5, que visa a padronização das aplicações.

Para avaliar o RepoSDN foram desenvolvidos testes contendo todas as funcionalidades do Midcontroller bem como seus componentes. Os componentes do Midcontroller foram desenvolvidos na linguagem Python[3] empregando também a tecnologia XML-RPC[4] para a conexão com o servidor.

A linguagem de programação Python foi escolhida pois além de ser indentada, o que facilita no entendimento e organização da linguagem, também é uma linguagem orientada a produção, Python compila automaticamente quando o programa é executado.

Para supervisionar o manejo das aplicações armazenadas no repositório, foi desenvolvido uma API que automatiza, controla e trata as aplicações de rede, permitindo a comunicação com os coletores do Midcontroller, que por sua vez inicia-os automaticamente no controlador Nox.

O ambiente de testes se deu no laboratório de pesquisa GERCOM, localizado na UFPA, onde os equipamentos que foram utilizados para a instalação e configuração do repositório de aplicações, e para a configuração da API, do Midcontroller, e do controlador Nox respectivamente foram:

[3]Python: http://www.python.org
[4]XML-RPC: http://www.xmlrpc.com

- Servidor Linux: 2.6.32-5-xen-amd64.
- Servidor Linux: Ubuntu 12.10 lts x86, 6Gb de memória ram, 1024Gb de disco.
- Switch Extreme 460 series.

## 6.2 Experimentos e Resultados

Foram realizados dois experimentos para a validação do RepoSDN, o primeiro foi o teste de carga, que teve o objetivo de medir o tempo que os componentes criados levariam para executar uma determinada quantidade de controladores na rede.

O segundo experimento realizado foi o teste de desempenho no servidor onde o RepoSDN está armazenado, afim de testar a quantidade de usuários que conseguiam se conectar e baixar uma determinada aplicação simultaneamente.

Pode-se explanar que o RepoSDN e seus componentes foram submetidos a um tipo de teste chamado de benchmarking, que tem por objetivo a melhoria do processo e gestão do RepoSDN, tornando como referência suas práticas e conceitos.

### 6.2.1 Teste de Carga

Os testes de carga, diferentemente do teste de desempenho, tem como objetivo a verificação do comportamento dos componentes com uma determinada quantidade de usuários acessando simultaneamente os mesmos. Dessa forma, o sistema é analisado sobre a utilização de um número estimado de usuários próximo das condições reais.

O primeiro experimento realizado foi o teste de carga, que analisou como o Controller Orchestrator e o Midcontroller se comportariam em um sistema real. Afim de validar o tempo de execução, foram realizadas sucessivas requisições HTTP utilizando a tecnologia XML-RPC para funcionar como cliente/servidor entre os dois componentes. O Controller Orchestrator está localizado no servidor e o Midcontroller funciona como o cliente, requisitando as funcionalidades no servidor do Controller Orchestrator.

Para analisar a quantidade de tempo de resposta que o sistema levaria para executar tal tarefa. Foram realizadas 10 execuções num intervalo de confiança de 95% para cada numero de requisições. No caso do teste, as seguintes requisições foram 10, 20, 30, 40, 50, 100, 500 e 1000 controladores. Pode-se observar os dados de cada tempo de execução do Midcontroller na Tabela 2 e do Controller Orchestrator na Tabela 3.

A Figura 13 mostra a média de tempo de processamento que o Controller Orchestrator levou para receber as 1000 solicitações do cliente que foi de 4,835 segundos e o desvio padrão foi de 0,565. Na Figura 14 pode se analisar que o cliente Midcontroller responde as 1000 requisições simultâneas em uma média de tempo de 1,170 segundos e o desvio padrão de 1,130. A barra de erro dos gráficos representa o desvio padrão.

Tabela 2: Tempo de Execução do Midcontroller.

| Amostras | Maior Tempo (s) | Menor Tempo (s) | Média (s) | Desvio Padrão |
|---|---|---|---|---|
| 10 | 0,090 | 0,040 | 0,065 | 0,025 |
| 20 | 0,110 | 0,040 | 0,075 | 0,035 |
| 30 | 0,130 | 0,040 | 0,085 | 0,045 |
| 40 | 0,150 | 0,040 | 0,095 | 0,055 |
| 50 | 0,180 | 0,040 | 0,110 | 0,070 |
| 100 | 0,290 | 0,040 | 0,165 | 0,125 |
| 500 | 0,810 | 0,040 | 0,425 | 0,385 |
| 1000 | 2,300 | 0,040 | 1,170 | 1,130 |

Tabela 3: Tempo de Execução do Controller Orchestrator.

| Amostras | Maior Tempo (s) | Menor Tempo (s) | Média (s) | Desvio Padrão |
|---|---|---|---|---|
| 10 | 0,100 | 0,040 | 0,070 | 0,030 |
| 20 | 0,140 | 0,080 | 0,110 | 0,030 |
| 30 | 0,160 | 0,120 | 0,140 | 0,020 |
| 40 | 0,210 | 0,160 | 0,185 | 0,025 |
| 50 | 0,260 | 0,230 | 0,245 | 0,015 |
| 100 | 0,550 | 0,410 | 0,480 | 0,070 |
| 500 | 2,410 | 2,260 | 2,335 | 0,075 |
| 1000 | 5,400 | 4,270 | 4,835 | 0,565 |

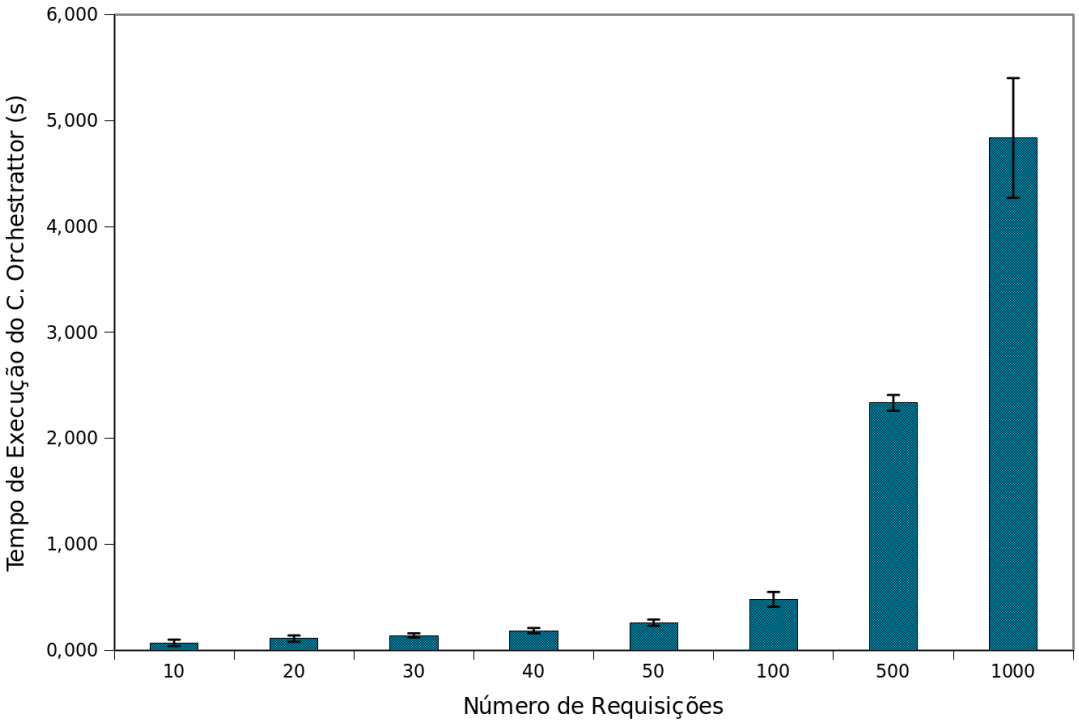


Figura 13: Tempo de Resposta do Controller Orchestrator

Portanto pode-se atentar que o Controller Orchestrator demorou mais tempo de processamento para tratar todas as solicitações do que o tempo do Midcontroller, pois ele

sempre fica em um estado de execução aguardando as solicitações do Midcontroller.

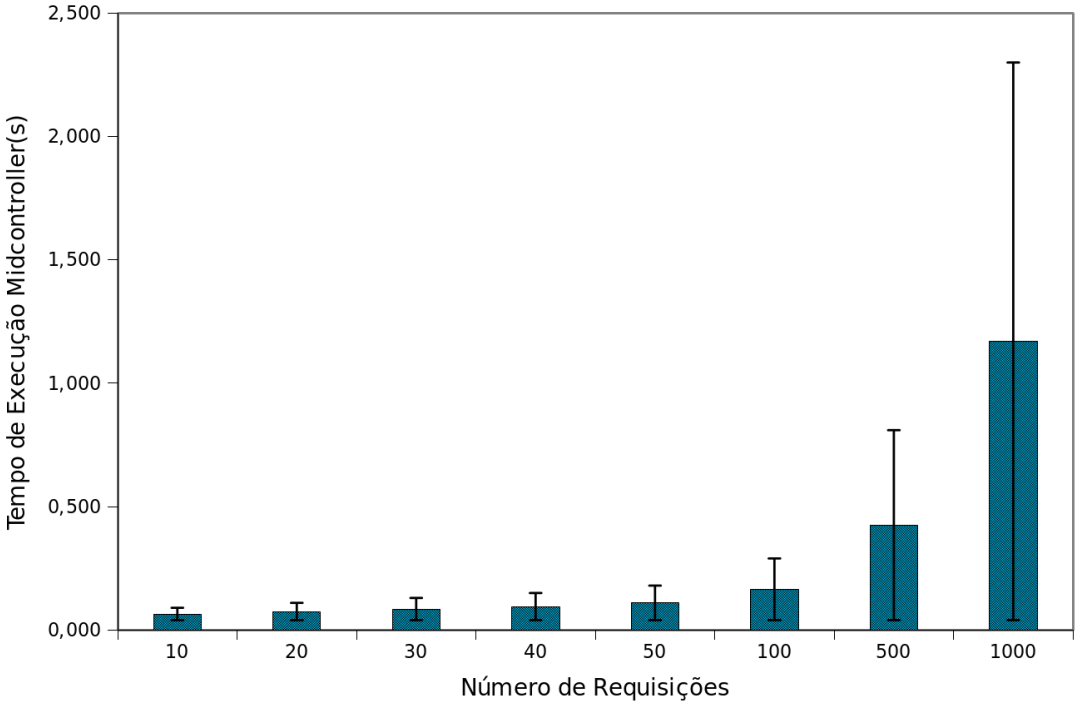


Figura 14: Tempo de Respota do Midcontroller

### 6.2.2 Teste de Desempenho no Repositório de Aplicações SDN

O teste de desempenho feito no repositório de aplicações é importante por se tratar de um ambiente onde todas as aplicações se encontram, e um fluxo muito grande de usuários pode acessar e baixar as aplicações simultaneamente, afinal este repositório se encontra na internet.

O desempenho medido no teste foi dado por uma ferramenta de código aberta desenvolvida para este propósito, chamada de JMeter[5] apresentado na Figura 15. Este tipo de teste permite saber quantos downloads simultâneos o repositório de aplicações suporta sem retornar a erros.

Uma das características mais importantes do Repositório de Aplicações SDN é a existência APIs, que tornam possível a tomada de decisão automática de acordo com a configuração que o administrador SDN implementou. Para validar a utilização do Repositório de Aplicações SDN foram feitos testes de desempenho no Repositório.

A ferramenta Apache JMeter é um software livre desenvolvido em java, projetado para medir capacidade e desempenho de aplicações web, consultas em servidores, simular uma determinada carga em um servidor ou grupo de servidores e analisar o desempenho global em diferentes tipos de carga.

[5]jmeter: http://jmeter.apache.org/

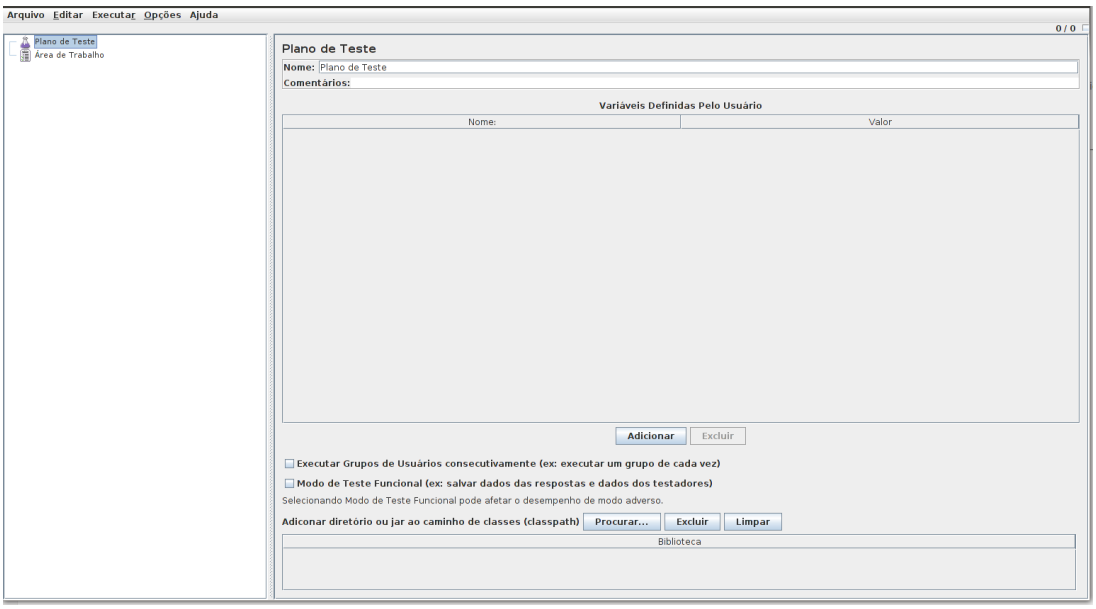


Figura 15: Interface Inicial do Apache Jmeter

Foram feitos testes no servidor onde o repositório está armazenado e configurado, foram criados 1.000, 2.000, 5.000, 10.000, 20.000 e 50.000 usuários virtuais (threads), onde o tempo de inicialização foi de 1 segundo e o contador de iteração foi de 1, ou seja, cada usuário irá requisitar uma vez a cada um segundo para o Repositório de Aplicações SDN.

Para ajudar nos testes de desempenho, o Repositório de Aplicações SDN tem armazenado um arquivo chamado ospfviz.zip de tamanho 70KB, onde todos os usuários virtuais fizeram o download através do protocolo HTTP com o método GET, da mesma forma que deve ser feito em um ambiente real.

A Tabela 4 a seguir apresenta os dados estatísticos coletados neste teste. Com estes dados, a exemplo do primeiro experimento, também foram contruídos gráficos para melhor visualização dos resultados obtidos.

Tabela 4: Teste de Desempenho no Repositório de Aplicações

| Amostras | 1.000 | 2.000 | 5.000 | 10.000 | 20.000 | 50.000 |
|---|---|---|---|---|---|---|
| Máximo(ms) | 257 | 605 | 2785 | 5759 | 15433 | 35098 |
| Mínimo(ms) | 3 | 3 | 3 | 3 | 3 | 3 |
| Média(ms) | 25 | 69 | 683 | 1300 | 1962 | 2327 |
| Desvio Padrão | 17,46 | 194,09 | 1293,18 | 2563,49 | 4415,44 | 6918,53 |
| % de Erro | 0 | 0 | 0 | 0 | 0 | 0 |
| Vazão(s) | 90,7 | 152,7 | 218,2 | 301,3 | 429,2 | 513,8 |
| KB/s | 10774,61 | 15127,81 | 20591,03 | 31189,62 | 37672,83 | 45668,21 |
| Média de Bytes | 121585,1 | 121665,2 | 122083,3 | 124005,4 | 127090,2 | 131009,6 |

Analisando as requisições HTTP pode-se observar que quanto mais usuários virtuais foram acrescentados, maior o tempo máximo que um usuário levou para fazer a requisição no repositório. Foram testados até 50.000 usuários virtuais sem que o JMeter

retornasse com pacotes de erros.

Com a utilização de 1.000 usuários virtuais o tempo máximo é de 257 milissegundos e o tempo médio é de 25 milissegundos. Com o teste de 50.000 usuários virtuais, o resultado para o tempo máximo é de 35.098 milissegundos e o do tempo médio é de 2.327 milissegundos.

O Gráfico da Figura 16 apresentada exibe média de tempo que os usuários virtuais levaram para acessar e fazer o download do arquivo no Repositório, onde nas 50.000 requisições houve um tempo médio de 2327 milisegundos e o desvio padrão de 2245,53. A barra de erro do gráfico representa o desvio padrão.

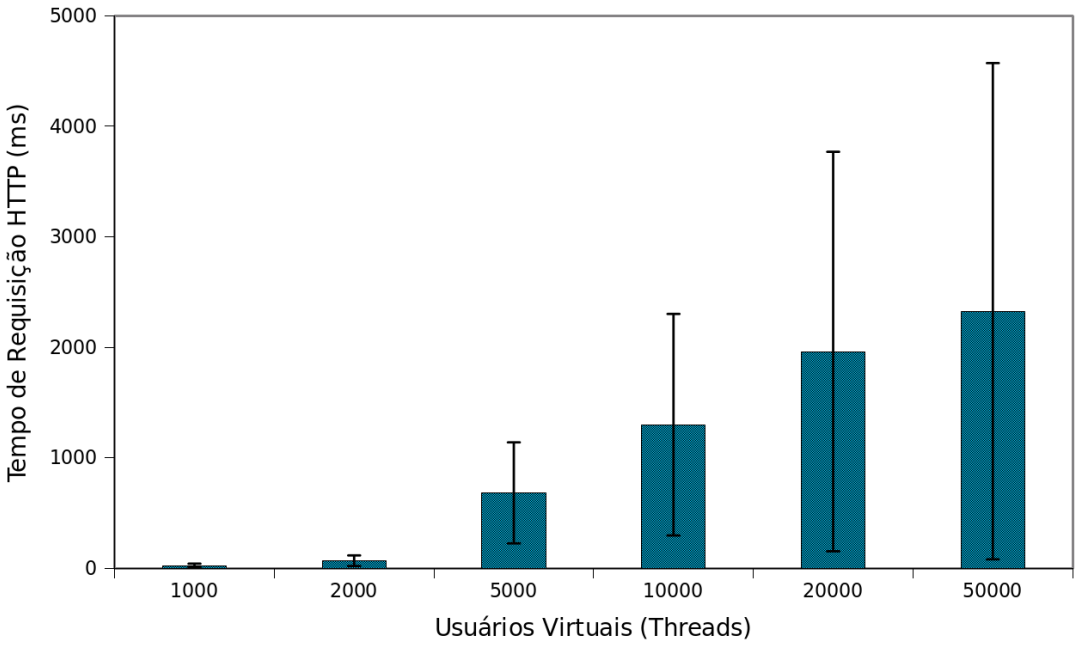


Figura 16: Média de Tempo dos Usuários Virtuais

O Gráfico da Figura 17 apresenta a medida da quantidade de requisições (Throughput) em KB/s, onde observar-se o crescimento linear de acordo com o aumento do número de usuários virtuais fazendo requisições para download do arquivo em questão no repositório, onde para 50.000 usuários virtuais, a medida da quantidade de requisições foi de 45668,21 kB/s.

O Gráfico da Figura 18 mostra o tamanho médio das respostas de todas as requisições do teste de desempenho em Bytes, onde para as 1.000 requisições iniciais a média das respostas em bytes foi de 121585,1 Bytes, e para a média de respostas das 50.000 requisições foi de 121585,6 Bytes.

Pode-se perceber um leve aumento na médias das respostas com o passar do aumento do número de requisições, o que é explicado pelo fato de que o servidor mantém um padrão de resposta para cada cliente.

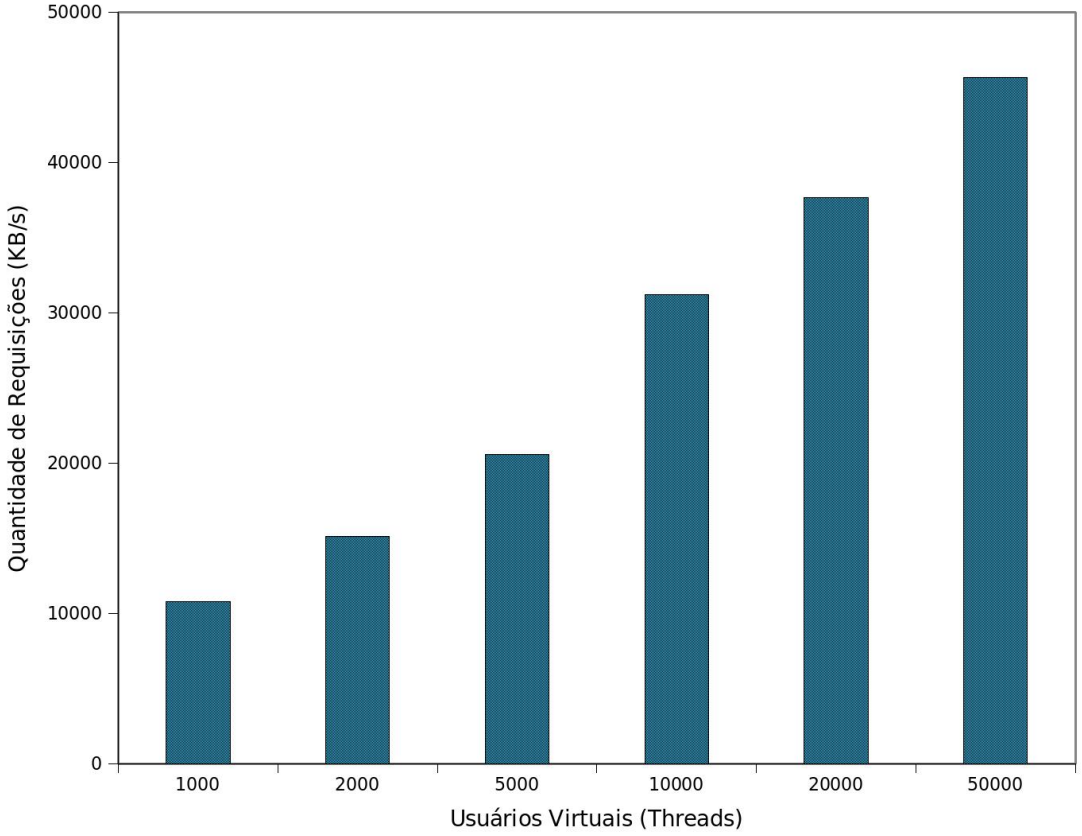


Figura 17: Quantidade de Dados Tranferidos

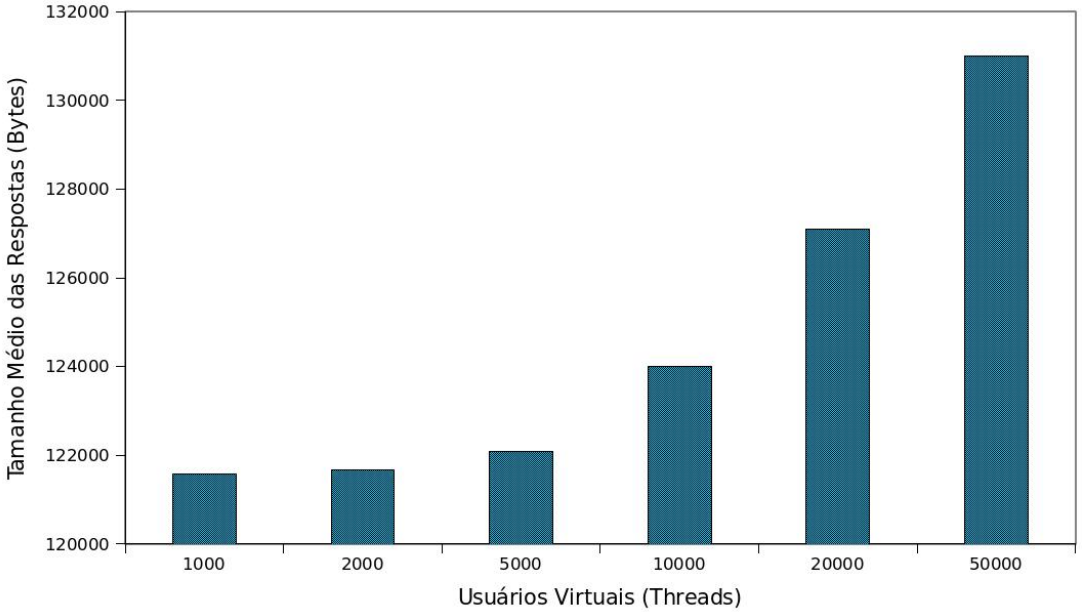


Figura 18: Média de Respostas das Amostras

## 6.3 Conclusões do Capítulo

Este capítulo apresentou os experimentos feitos para a validação da proposta do método de organização e coordenação de aplicações em repositório para redes defini-

das por software, o RepoSDN. Os experimentos realizados foram usados para avaliar a aceitabilidade dos limites operacionais do RepoSDN.

Pode-se concluir com esse capítulo que todos os testes realizados foram satisfatórios no sentido de provar o uso dos componentes numa rede programável, haja vista que os testes se deram em duas partes, um teste de carga nos componentes do gerenciador de aplicações e outro teste de desempenho no repositório de aplicações, duas peças fundamentais para todo o método de organização proposto.

# CAPÍTULO 7

# Conclusões

A implantação e gerenciamento de aplicações para Redes Definidas por Software em repositórios tem sido um grande desafio. Apesar de sua importância, poucos estudos tem sido feitos na área. Nesta dissertação, apresentamos um novo método de organização e coordenação de aplicações e gerenciamento do processo que as aplicações levam para sair do desenvolvedor até chegar nos controladores, levando em conta os aspectos de implantação e gerência, bem como propiciando características autonômicas aos controladores.

Os testes de carga com o Controller Orchestrator e o Midcontroller mostraram que podem suportar um numero de 1.000 requisições sem gerar sobrecarga no sistema. Já os testes de desempenho e estresse no arquivo escolhido do Repositório de Aplicações SDN comprovaram a viabilidade do mesmo para prover as 50.000 requisições.

### 7.0.1 Contribuições

A principal contribuição desta dissertação foi propor o RepoSDN, um método para ajudar e facilitar a implantação e gerenciamento de aplicações para redes definidas por software, além de automatizar uma série de tarefas para o administrador de aplicações, permitindo com que ele tenha um maior controle de cada aplicação sendo executada nos controladores.

Desta forma as principais contribuições desta dissertação com base nos objetivos definidos na introdução são:

- Definição dos métodos para a implantação do RepoSDN.
- Definição das políticas de uso do repositório de aplicações.

- Definição de todo o fluxo de trabalho e arquitetura do RepoSDN.
- Implementação do repositório de aplicações.
- Implementação do protótipo do gerenciador de aplicações.
- Realização dos experimentos com o repositório de aplicações e o gerenciador de aplicações.

### 7.0.2 Conclusões Gerais

Este método de organização e coordenação visa incentivar o desenvolvimento de aplicações SDN para armazenamento em repositórios, uma vez que este repositório fornece a base teórica necessária para a sua criação, por especificar e padronizar os elementos que compõem as aplicações, dado que um dos maiores obstáculos para implantação desse modelo é não existir um lugar para reunir todas as aplicações existentes ou as que estão por vir. A flexibilidade do componentes desenvolvidos permite que a inclusão de serviços através do Midcontroller.

### 7.0.3 Trabalhos Futuros

Como trabalhos futuros, pretendemos ampliar os componentes já propostos e incluir um modelo de interface gráfica para aplicativos SDN capaz de comunicar usuários por meio de um conjunto de protocolos e tecnologias padrão, a interface será responsável por proporcionar ao usuário uma maneira mais fácil e prática de realizar o gerenciamento de uma determinada SDN.

As políticas de segurança das aplicações implementadas devem ser tratadas como trabalho futuro, pois deve haver uma verificação em cada aplicação no repositório de aplicações antes que as mesmas sejam executadas nos controladores, evitando assim possíveis invasões na rede.

As atualizações das aplicações devem ser executadas de forma com que não prejudique a execução de outras aplicações, e ainda sim verificar a integridade da aplicação atual, para que não afete o desempenho da SDN. Os desenvolvedores devem mandar a nova versão das aplicações para o repositório de aplicações, como um novo processo de envio. Os administradores da SDN devem baixar a nova versão e atualizar na rede local.

# Referências

[Apple 2014] Apple (2014). App Store. Disponível em: http://www.apple.com/itunes/charts/paid-apps/. [Online; acessado em 02, Fevereiro de 2014].

[APT 2007] APT (2007). Gerenciamento de pacotes com o APT. Disponível em: http://wiki.ubuntu-br.org/AptGet. [Online; acessado em 02, Fevereiro de 2014].

[Beacon 2013] Beacon (2013). What is Beacon? Disponível em: https://openflow.stanford.edu/display/Beacon. [Online; acessado em 02, Fevereiro de 2014].

[Costa 2013] Costa, L. R. (2013). Openflow e o paradigma de redes definidas por software.

[De Araújo 2013] De Araújo, M. R. A. G. (2013). Uma abordagem para aprovisionamento de qos em redes definidas por software baseadas em openflow.

[De Jesus et al. 2013] De Jesus, W. P., Dos Santos, R. L., Rendón, O. M. C., and Granville, L. Z. (2013). Provinet: Uma plataforma para gerenciamento de redes virtuais programáveis.

[de Oliveira Silva et al. 2012] de Oliveira Silva, F., de Souza Pereira, J. H., Rosa, P. F., and Kofuji, S. T. (2012). Enabling future internet architecture research and experimentation by using software defined networking. In *Software Defined Networking (EWSDN), 2012 European Workshop on*, pages 73–78. IEEE.

[dos Reis et al. 2013] dos Reis, V. R., Coletti, C., and Senne, E. L. F. (2013). Implementando routeflow em uma rede em produção.

[Farias et al. 2011] Farias, F. N., Júnior, J. M. D., Salvatti, J. J., Silva, S., Abelém, A. J., Salvador, M. R., and Stanton, M. A. (2011). Pesquisa Experimental para a Internet do Futuro: Uma Proposta Utilizando Virtualização e o Frame-work Openflow. *Minicurso da SBRC*.

[Floodlight 2014] Floodlight (2014). Floodlight Is an Open SDN Controller. Disponível em: http://www.projectfloodlight.org/floodlight/. [Online; acessado em 02, Fevereiro de 2014].

[Foundation 2014] Foundation, O. N. (2014). OpenFlow-enabled SDN and Network Functions Virtualization. Disponível em: https://www.opennetworking.org/images/stories/downloads/sdn-resources/solution-briefs/sb-sdn-nvf-solution.pdf . [Online; acessado em 02, Fevereiro de 2014].

[Github 2014] Github (2014). Github. Disponível em: https://github.com/. [Online; acessado em 02, Fevereiro de 2014].

[Google 2014] Google (2014). Google Play. Disponível em: https://play.google.com/store. [Online; acessado em 02, Fevereiro de 2014].

[Greenhalgh et al. 2009] Greenhalgh, A., Huici, F., Hoerdt, M., Papadimitriou, P., Handley, M., and Mathy, L. (2009). Flow Processing and the Rise of Commodity Network Hardware. *ACM SIGCOMM Computer Communication Review*, 39(2):20–26.

[Gude et al. 2008] Gude, N., Koponen, T., Pettit, J., Pfaff, B., Casado, M., McKeown, N., and Shenker, S. (2008). Nox: towards an operating system for networks. *ACM SIGCOMM Computer Communication Review*, 38(3):105–110.

[HP 2013] HP (2013). HP Networking. Disponível em: http://h17007.www1.hp.com/. [Online; acessado em 02, Fevereiro de 2014].

[HP 2014a] HP (2014a). Controlador SDN de Redes de Aplicativos Virtuais HP. Disponível em: http://pro-networking-h17007.external.hp.com/br/pt/solutions/technology/openflow/index.aspx. [Online; acessado em 02, Fevereiro de 2014].

[HP 2014b] HP (2014b). Redes de Aplicativos Virtuais HP. Disponível em: http://pro-networking-h17007.external.hp.com/br/pt/solutions/technology/van/index.aspx. [Online; acessado em 02, Fevereiro de 2014].

[Kim and Feamster 2013] Kim, H. and Feamster, N. (2013). Improving network management with software defined networking. *IEEE Communications Magazine*, 51(2):114–119.

[Kind et al. 2012] Kind, M., Westphal, F., Gladisch, A., and Topp, S. (2012). Splitarchitecture: Applying the software defined networking concept to carrier networks. In *World Telecommunications Congress (WTC), 2012*, pages 1–6. IEEE.

[Maestro 2010] Maestro (2010). Maestro-Platform. Disponível em: https://code.google.com/p/maestro-platform/. [Online; acessado em 02, Fevereiro de 2014].

[Nanda and Kotz 2008] Nanda, S. and Kotz, D. (2008). Mesh-mon: A multi-radio mesh monitoring and management system. *Computer Communications*, 31(8):1588–1601.

[Nox 2013] Nox (2013). About NOX. Disponível em: http://www.noxrepo.org/nox/about-nox/. [Online; acessado em 02, Fevereiro de 2014].

[ONF 2014] ONF (2014). Open Network Foundation. Disponível em: https://www.opennetworking.org/. [Online; acessado em 02, Fevereiro de 2014].

[ON.LAB 2014] ON.LAB (2014). OPEN-SOURCE SDN STACK. Disponível em: http://onlab.us/tools.html. [Online; acessado em 02, Fevereiro de 2014].

[Openflow 2011] Openflow (2011). OpenFlow News. Disponível em: http://archive.openflow.org/. [Online; acessado em 02, Fevereiro de 2014].

[Paul et al. 2011] Paul, S., Pan, J., and Jain, R. (2011). Architectures for the Future Networks and the Next Generation Internet: A Survey. *Computer Communications*, 34(1):2–42.

[Pox 2013] Pox (2013). About POX. Disponível em: http://www.noxrepo.org/pox/about-pox/. [Online; acessado em 02, Fevereiro de 2014].

[RFC-2080 1997] RFC-2080 (1997). RIPng for IPv6. Disponível em: https://tools.ietf.org/html/rfc2080. [Online; acessado em 02, Fevereiro de 2014].

[RFC-2328 1998] RFC-2328 (1998). OSPF Version 2. Disponível em: https://tools.ietf.org/html/rfc2328. [Online; acessado em 02, Fevereiro de 2014].

[Rothenberg et al. 2011] Rothenberg, C. E., Nascimento, M. R., Salvador, M. R., and Magalhães, M. F. (2011). Openflow e Redes Definidas por Software: Um Novo Paradigma de Controle e Inovação em Redes de Pacotes. 7(1).

[RouteFlow 2013] RouteFlow (2013). Welcome to the RouteFlow Project! Disponível em: https://openflow.stanford.edu/display/DOCS/Flowvisor. [Online; acessado em 02, Fevereiro de 2014].

[Sezer et al. 2013] 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. *Communications Magazine, IEEE*, 51(7).

[Vasco 2013] Vasco, M. A. A. (2013). Emulating network applications with a software-defined networking architecture. *Universidade de Lisboa*.

# ANEXO A – Primeiro anexo

Para a execução das APIs desenvolvidas o administrador SDN deve colocar em cada servidor o arquivo Midcontroller.py e editar o mesmo com os referidos IPs fixos dos servidores, em seguida abrir o terminal e através da linha de comando executar a seguinte sintaxe para a execução dos comandos: controlerOrchestrator.py [argumento] [servidor].

Como exemplo de comando, é passado o argumento *-h*, que tem a opção de mostrar todos os argumentos possíveis do Controller Orchestrator para executar em um servidor. É importante atentar que essa é a única opção que não é necessário inserir o servidor.

```
./controlerOrchestrator.py -h
```

Comando que tem como saída:

```
usage: controlerOrchestrator.py
[-h] [-l LIST] [-i INFO] [-I INSTALL]
[-t TASK] [-d DELETE [DELETE ...]]
[-L LOG]

optional arguments:
-h, --help                  Show this Help Message
-l LIST, --list             List Application
-i INFO, --info             Information Application
-I INSTALL, --install       Install Application
-t TASK, --task             Schedule Task
-d DELETE, --delete         Delete Application
-L LOG, --log               View Log
```

A Tabela 5 apresenta a lista completa dos argumentos possíveis e seus significados para a execução em linha de comando pelo Controller Orchestrator.

Outros exemplos de execução dos comandos são mencionados abaixo, e onde *s0* é o nome de um servidor que receberá o comando. Para alterar o IP ou acrescentar novos servidores basta alterar o programa *controlerOrchestrator.py*.

```
./controlerOrchestrator.py -i s0
```

Tabela 5: Sintaxe da Aplicação Controller Orchestrator.

| Argumentos | Significado |
|---|---|
| –help ou -h | Mostra mensagem de ajuda e saí do programa. |
| –list ou -l | Lista todos os aplicativos baixados do Repositório de Aplicações. |
| –info ou -i | Mostra as informações da aplicação. |
| –install ou -I | Instala uma determinada aplicação. |
| –task ou -t | Agenda uma determinada tarefa. |
| –delete ou -d | Deleta uma determinada aplicação. |
| –log ou -L | Mostra o log |

E a saída do comando passando o argumento *-i* e escolhendo a aplicação *nox-verity-1.0-all.zip* é:

```
Entrou no servidor 1
Choose your Application to find Information:

Zip Application Name You Want to Search:
nox-verity-1.0-all.zip

NAME_APP: NOX Network Control Platform
VERSION_APP: 1.0
APP_TYPE: bin
MAINTAINER: https://github.com/noxrepo/nox
DEPENDENCES_APP: no
NAME_CONTROLLER: nox
VERSION_CONTROLLER: 0.5
ARCHITECTURE: all
COPYRIGHT: no.
DESCRIPTION: Welcome to the NOX network control platform.
This distribution includes all the software you need to
build, install, and deploy NOX in your network, as well
as source code and tools to allow you to develop your own
NOX applications.
```

Passando o argumento *-l* teremos como saída uma lista com todas as aplicações já baixadas do Repositório de Aplicações.

```
./controlerOrchestrator.py -l s0
```

Como resultado podemos observar a seguir:

```
Entrou no servidor 1
['nox-verity-1.0-all.zip', 'extracted-files',
'spanning-tree-1.0-all.zip', 'rstplib-1.1-all.zip',
'pox-carp-2.0.all.zip', 'ospfviz07-1.1-all.zip',
```

```
'RIPv1-master-1.0-all.zip', 'myapp.log']
```

O argumento *-I* permite com que o administrador escolha entre as aplicações que ele baixou a que vai instalar no controlador.

```
./controlerOrchestrator.py -I s0
```

O resultado do comando pode ser observado abaixo:

```
Entrou no servidor 1
Installing Application:
Entre com o nome do aplicativo a ser instalado:
```

O Argumento *-t* faz com que o administrador das aplicações possa agendar uma determinada tarefa no servidor.

```
./controlerOrchestrator.py -t s0
```

O argumento solicita o agendamento de uma determinada tarefa no servidor, permitindo que o administrador escolha os minutos, hora, dia, mês, dia da semana, usuário que vai execurtar a tarefa e por fim o comando que será executado.

```
Entrou no servidor 1

Schedule Task with Cron:
Minutes ( 0-59 ): 2

Hour ( 0-23 ): 5

Day ( 1-31 ): *

Month ( 1-12 ): *

Day of Week ( 0-6 )(Sunday=0): *

User: root

Task: echo Agendador de Tarefas
```

Quando o administrador passa como argumento o parametro *-d* para o servidor, ele vai escolher a aplicação que será excluída.

```
./controlerOrchestrator.py -d s0
```

O adminitrador da rede pode escrever o nome da aplicação que será excluída do servidor, como podemos observar abaixo a saída do comando:

```
Entrou no servidor 1
Choose your Application to be Removed

Application name .zip you want to delete:
```

Para o administrador da rede visualizar todos os registros do Controller Orchestrator, basta entrar com o parâmetro *-L* no comando.

```
./controlerOrchestrator.py –L s0
```

O Log do sistema fica armazenado em um arquivo chamado *myapp.log*, podemos ver algumas linhas do arquivo abaixo:

```
2014-02-06 14:57:59,133 DEBUG Option: -i: Information Application
2014-02-06 15:00:46,572 DEBUG Option: -I: Installing Application
2014-02-06 15:01:11,778 DEBUG Option: –L: View Log
2014-02-12 12:25:56,404 DEBUG Option: -t: Schedule Task
2014-02-12 13:50:16,476 DEBUG Option: –d: Delete Application
2014-02-12 14:36:49,772 DEBUG Option: –L: View Log
```