UNIVERSIDADE FEDERAL DO PARÁ
INSTITUTO DE CIÊNCIAS EXATAS E NATURAIS
FACULDADE DE COMPUTAÇÃO
BACHARELADO EM SISTEMAS DE INFORMAÇÃO

DÂNGELO WESLEY OLIVEIRA MENDES

UM GERENCIADOR DE PACOTES DE APLICAÇÕES PARA REDES DEFINIDAS POR SOFTWARE

BELÉM-PA

Dezembro, 2015

DÂNGELO WESLEY OLIVEIRA MENDES

UM GERENCIADOR DE PACOTES DE APLICAÇÕES PARA REDES DEFINIDAS POR SOFTWARE

Trabalho de Conclusão de Curso apresentado no curso de Bacharelado em Sistemas de Informação da Universidade Federal do Pará, como requisito parcial para obtenção do grau de Bacharel em Sistemas de Informação.

Orientador: Prof. Dr. Antonio Jorge Gomes Abelém

Co-Orientadores: Msc. Billy Anderson Pinheiro

BELÉM-PA

Dezembro, 2015

DÂNGELO WESLEY OLIVEIRA MENDES

UM GERENCIADOR DE PACOTES DE APLICAÇÕES PARA REDES DEFINIDAS POR SOFTWARE

Trabalho de Conclusão de Curso apresentado no curso de Bacharelado em Sistemas de Informação da Universidade Federal do Pará, como requisito parcial para obtenção do grau de Bacharel em Sistemas de Informação.

Aprovada em: __/__/____

BANCA EXAMINADORA

Prof. Dr. Antonio Jorge Gomes Abelém
Universidade Federal do Pará
Orientador

Msc. Billy Anderson Pinheiro
Universidade Federal do Pará
Co-orientador

Prof. Dr. Raimundo Viégas Júnior
Universidade Federal do Pará

Prof. Msc. Rômulo Silva Pinheiro
Universidade da Amazônia

Ao meus eternos e saudosos avós: Albertina e José, ao meu eterno, saudoso e amado pai Agostinho Castro Mendes pela criação, pelas ótimas lembranças e pelos ensinamentos deixados, à minha amada mãe Eunice Oliveira Mendes pelo exemplo de mulher guerreira, pela criação, ensinamentos e por ter me proporcionado toda a base que precisei para galgar o nível superior, aos meus amigos e colegas.

Agradecimentos

Dedico meus sinceros agradecimentos primeiramente a Deus que tem me dado saúde, forças e coragem para continuar buscando a realização de meus sonhos.

Ao meu orientador Professor Doutor Abelém, pela liberdade oferecida de estar junto de todos desta Equipe (assim mesmo com "E"maiúsculo), pela orientação e incentivo. Ao co-orientador deste trabalho, o Mestre Billy, chefe e parceiro, por me apoiar a iniciar, desenvolver e concluir este trabalho. Ao Mestre Rômulo Pinheiro, ser humano incrível e bastante agradável, pelas contribuições pontuais e sempre edificadoras. Aos colegas de turma pela companhia e amizade, em especial ao Samuca, Diogo e ao Léo, os quais sempre estiveram comigo em diversos momentos, ora trabalhando, me dando algum tipo de força ou divindo alegrias e boas conversas. À todos da Equipe do laboratório GERCOM, em especial aos colegas: Airton, Fernando, Vagner, Raphael, André e Júnior, pelos diversos socorros e ajuda. Aos membros desta banca, o que inclui o meu Professor Doutor Raimundo Viégas Júnior, de quem vou levando grandes e boas lembranças da sala de aula. À Tamirys ou melhor, "Tamir". À todos os meus amigos do condomínio Jardim Bela Vida I, em especial ao meu amigo Gilvan (Hulk), com quem é sempre bom curtir um som de violão e apreciar um vinho, esta bebida que o seu pequeno filho Gustavo aprendeu a chamar de "suco de papai". Aos amigos de Irituia. À minha tia Marlene e ao meu tio Josué, os quais por diversas vezes abriram as portas de sua casa para mim. À minha família e a todos os demais que me ajudam a fazer do mundo um lugar mais feliz.

Aos meus pais que me deram todo a base que precisei para chegar até aqui, pois ambos estão, ainda que distantes, sempre presentes em todos os aspectos de minha vida, sendo que sem eles eu julgo que jamais seria alguém.

Resumo

Resumo do Trabalho de Conclusão de Curso apresentado à UFPA como parte dos requisitos necessários para obtenção do grau de Bacharel em Sistemas de Informação.

UM GERENCIADOR DE PACOTES DE APLICAÇÕES PARA REDES DEFINIDAS POR SOFTWARE

Orientador: Prof. Dr. Antonio Jorge Gomes Abelém
Co-orientador: Msc. Billy Anderson Pinheiro
Palavras-chave: SDN; SPM; *control path*; API; POX

O paradigma de Redes Definidas por Software (SDN) tem sido investigado como a solução mais promissora para o atual engessamento da Internet, uma vez que propõe a dissociação entre o plano de dados e o plano de controle, proporcionando maior programabilidade às redes. Contudo, não existem atualmente ferramentas abertas que promovam o gerenciamento de aplicações para controladores SDN, o mesmo ocorre com a padronização no empacotamento destas. Por esse motivo, neste trabalho será apresenta uma solução chamada de Software Package Manager for SDN Applications - SPM. Esta ferramenta define um método de empacotamento para aplicações SDN, a fim de tornar prático o seu gerenciamento no plano de controle de redes programáveis, a partir do uso de APIs que automatizam a instalação, remoção e atualização das mesmas as quais serão distribuídas por meio de um repositório Web, contendo estes pacotes.

Para validação deste trabalho foi selecionado o controlador POX, a partir do qual foi possível a composição de um banco de dados para que a ferramenta SPM possa selecionar as informações que precisa para gerenciar aplicações neste controlador. Ao longo do texto será apresentada a relação deste trabalho com os elementos do plano de controle SDN, bem como demonstrada a semelhança desta proposta com o gerenciamento de pacotes no GNU/Linux. Finalmente será detalhada a solução dada pela ferramenta SPM e apresentados os resultados desta implementação.

Abstract

Abstract of bachelor monograph presented to UFPA as a partial fulfillment of the requirements for the degree of Bachelor of Information Systems.

AN APPLICATIONS PACKAGE MANAGER FOR SOFTWARE-DEFINED NETWORKING

Advisor: Prof. Dr. Antonio Jorge Gomes Abelém
Co-advisor: Msc. Billy Anderson Pinheiro
Key words: SDN; SPM; control path; API; POX

The paradigm of Software-Defined Networking (SDN) has been investigated as the most promising solution to the current ossification of the Internet problem, since it proposes the decoupling of data plane and control plane, providing greater programmability to networks. However, there are currently no open tools that promote the application management for SDN controllers, similarly there are no standardization to packaging it. For this reason, in this work will be presented a solution called Software Package Manager for SDN Applications - SPM. This tool defines a method for packaging applications SDN, in order to make their practical the control path management of programmable networks, from the use of APIs that automates the installation, update and removal of same which are distributed through a Web repository of these packages.

For validation of this work was selected POX controller, from which it was possible the composition of a database for the SPM tool can select the information it need to manage applications in this controller. Throughout the text will be presented the relation of this work with the elements of SDN control plane as well as demonstrated the similarity of this proposal with the package management on GNU/Linux. Finally it will be detailed the solution given by the SPM tool and presented the results of this implementation.

Sumário

Lista de Abreviaturas

SPM	Software Package Manager for SDN Applications
TCP	Transfer Control Protocol
IP	Internet Protocol
SDN	Software-Defined Networking
IETF	Internet Engineering Task Force
NAT	Network Address Translation
QoS	Quality of Service
SSL	Secure Socket Layer
TLS	Transport Layer Security
IANA	Internet Assigned Numbers Authority
CLI	Command Line Interface
TCC	Trabalho de Conclusão de Curso
RepoSDN	Repositório SDN
RPM	Red Hat Packet Manager
LSB	Linux Standard Base
DEB	Debian Binary Packages
APT	Advanced Packaging Tool
GTK+	GIMP Toolkit
YUM	Yellow dog Updater Modified
URL	Uniform Resource Locator
OSPF	Open Shortest Path First
RIP	Routing Information Protocol
UML	Unified Modeling Language
LANOMS	Latin American Network Operations and Management Symposium
UFPB	Universidade Federal da Paraíba

Lista de Figuras

Lista de Tabelas

CAPÍTULO 1

Introdução

1.1 Visão geral

A evolução dos serviços oferecidos na Internet não são acompanhadas pelo já tradicional modelo TCP/IP e isto inclui as camadas que o representam. Em razão de tais limitações na rede tradicional, pesquisadores apontam que a sua evolução se torna cada vez mais crítica devido ao que chamam de "ossificação da Internet", causada dentre outras coisas, pela dificuldade de alteração em seu núcleo [11], uma vez que nas redes tradicionais não há possibilidade de expansão refletida nos hardwares e softwares proprietários os quais possuem arquiteturas e tecnologias fechadas que impedem o seu desenvolvimento para atender novas necessidades.

Neste contexto, surge o paradigma SDN (Software-Defined Networking) como solução a este problema, tendo como principal característica a dissociação entre o plano de controle e o plano de dados.

Assim, a rede tem seu controle centralizado em um novo elemento chamado de controlador, o qual é responsável por coordenar os comutadores. Estes elementos utilizam um protocolo de fluxo, como por exemplo o OpenFlow, que atua fazendo interface entre os planos da rede através de um canal seguro, o que garante não só a proteção contra elementos mal intencionados como também assegura a troca de informações de modo confiável e sem erros [12].

Com a possibilidade de programar a rede, torna-se possível o desenvolvimento de aplicações para os mais diversos propósitos como: roteamento, segurança, controle de acesso e outros, sendo que estas aplicações são criadas para as diferentes arquiteturas de controladores SDN.

Isto implica na adoção de diferentes linguagens de programação para o desenvol-

vimento destas aplicações. Por esse motivo foi desenvolvida a ferramenta SPM, a qual oferece, ao administrador da SDN, um modelo para organizar o plano de controle SDN por intermédio da gestão de seus elementos - controladores e aplicações SDN - definindo a padronização e distribuição centralizada de pacotes de aplicações a partir de um repositório Web, com a finalidade de gerenciá-las nos controladores.

1.2 Motivação e desafios

O paradigma SDN está em franco desenvolvimento. Portanto sua padronização não foi totalmente definida, permitindo que os controladores existentes possam adotar diferentes métodos para tratar aplicações, tal liberdade torna a distribuição de aplicações um ambiente confuso pois não há um padrão a ser seguido para empacotá-las.

Surge então um problema em aberto para se tratar aplicações SDN, uma vez que uma aplicação desenvolvida em C/C++ para o controlador Nox[1] utiliza diferentes recursos se os compararmos com os arquivos Java suportados pelo Floodlight[2], assim ao utilizar controladores de arquiteturas diferentes o usuário se depara com problemas para gerenciar aplicações devido cada controlador implementá-las de acordo com métodos próprios.

Além disso, o plano de controle SDN pode possuir mais de um controlador e assim, em cada controlador podem executar diferentes aplicações. Portanto à medida que os elementos do ambiente aumente este se torna um cenário pouco escalável e com grande complexidade para gerenciá-lo.

Com isto, torna-se necessário dispor de uma ferramenta que defina a padronização referente ao gerenciamento de aplicações SDN e também especifique uma modelagem para o seu empacotamento, centralizando sua distribuição em um repositório Web, garantindo assim facilidade para gerenciar tais aplicações, possibilitando ao administrador da SDN tratá-las de modo transparente para realizar a instalação, atualização e remoção de aplicações em controladores SDN.

1.3 Objetivos

Este trabalho tem como objetivo principal propor uma padronização dos pacotes de software contendo aplicações para Redes Definidas por Software. Para isto é necessário modelar o empacotamento e o conteúdo destes pacotes, a fim de facilitar o seu gerenciamento com a adoção da ferramenta SPM para disponibilizar aplicações no *control path* de redes programáveis.

Para este trabalho atender ao objetivo principal, são destacados os seguintes objetivos secundários a serem solucionados:

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

- Estudar o controlador POX e seus comandos para compreensão dos métodos utilizados para obtenção de uma aplicação;
- Identificar o conteúdo necessário para criação de um modelo de pacote para aplicações SDN;
- Estudar os principais gerenciadores de pacotes para o GNU/Linux;
- Definir um repositório Web onde seja possível hospedar, centralizar e distribuir os pacotes de aplicações SDN padronizados;
- Criar a ferramenta SPM e seu banco de dados utilizado para gerenciar aplicações no POX a partir do repositório Web;
- Descrever por meio de diagramas UML a sequência de eventos decorrentes da utilização da ferramenta SPM;
- Validar a proposta demonstrando a estrutura e organização dos componentes da ferramenta SPM;

1.4 Organização do texto

Os capítulos seguintes se dividem do seguinte modo:

- Capítulo 2: Este capítulo descreverá um resumo do atual cenário da Internet, bem como as limitações atuais na rede global. Apresenta-se as pesquisas que visam solucionar tais limitações, descrevendo o que se espera para a Internet do Futuro, além de fornecer definições sobre o principal arcabouço de investigação a cerca deste tema chamado de Redes Definidas por Software.
- Capítulo 3: Neste capítulo apresenta-se os trabalhos que serviram de base para implementar-se este trabalho de conclusão de curso, a fim de relacioná-los com a atual proposta.
- Capítulo 4: Aqui será detalhada a solução proposta para este trabalho que é chamada de Software Package Manager for SDN Applications (SPM). Neste capítulo descreve-se a sua: arquitetura, modelagem, definições e funcionalidades, destacando a sua importância para as Redes Definidas por Software, e demonstrando por meio de diagramas a sua atuação no contexto de controladores SDN.
- Capítulo 5: Neste capítulo será apresentado o ambiente da ferramenta SPM demonstrando os resultados conseguidos com a sua implementação.
- Capítulo 6: Este capítulo apresentará as considerações finais sobre este trabalho, onde por meio de análise destaca-se as próximas etapas a serem realizadas, de modo a expandir a proposta e descrever o que ainda poderá ser feito utilizando este tema.

CAPÍTULO 2

Redes Definidas por Software

2.1 Breve Histórico das Redes de Computadores

A Internet se tornou presente em diversas tarefas do cotidiano, de modo que relacionamentos, estudos, o exercicío da cidadania e até mesmo grandes e pequenos negócios, são mantidos ou impulsionados pela Rede Global [13].

Contudo, a instituição responsável por padronizar a Internet, o IETF, reconheceu ainda no início da década de 90 que a tecnologia IPv4 em quem se baseia a Internet atual estava ameaçada pelo esgotamento de endereços, sendo este um verdadeiro entrave à sua escalabilidade, tornando imprescindível que se criem novos meios que visem a superação de tais limitações [14], foi então que criou-se o IPv6, solução mais robusta e com capacidade gigantesca para o endereçamento na rede, atendendo ao problema relacionado ao "racionamento de endereços", mas que resolve não apenas este problema em específico como também introduz melhorias no roteamento, a exemplo do cabeçalho de comprimento fixo de 40 bytes, distinguindo-o do IPv4 que possui comprimento de cabeçalho variável, além de outros.

Assim, ao analisar-se a Internet, vemos que ela foi projetada em um contexto bem diferente do atual, toda a capacidade que dela se exige atualmente, dentre as quais podemos citar a crescente demanda por transferência de informação multimídia e recursos de redes móveis, não foram contemplados no projeto inicial.

Após sua consolidação, observou-se o que conhecemos por ossificação da Internet, que é uma característica observada no núcleo da rede e expressa, dentre outras, pelos hardwares e softwares proprietários que o compõem, constituindo verdadeiras "caixas pretas"que impedem as comunidades de desenvolvimento aberto de agregarem conhecimento à essas plataformas fechadas.

2.2 Internet do Futuro

O problema relacionado ao avanço da Internet tem chamado a atenção do meio acadêmico e, neste cenário observa-se a mobilização pela superação de tais dificuldades. Contrapondo a abordagem da arquitetura tradicional das redes de computadores, surgem metodologias de investigações para Internet do Futuro onde diversos grupos de pesquisadores somam forças, tendo o apoio e financiamento da indústria, com o intuito de alavancar a superação dos atuais desafios associados, dentre outros aspectos, a(o): escalabilidade, endereçamento, mobilidade e gerenciamento de redes.

As Redes Definidas por Software (SDN) tem conquistado notoriedade desde seu surgimento em 2008, onde os seus pioneiros defendem a reconstrução da rede, desenvolvendo a superação da infraestrutura atual e criando novas soluções que necessitam de uma nova abordagem [15]. Estes grupos, desejam vencer as limitações da Internet atual que recebeu, ao longo do tempo, diversas adaptações e ajustes que satisfazem apenas parcialmente as necessidades da Internet, mas não conseguem atender a demanda por novos serviços de forma satisfatória.

Devido a isto, criaram-se frentes de pesquisas para tratarem esta abordagem, onde destaca-se a frente denominada *clean slate* - na qual estão inseridas investigações como SDN. Há ainda uma segunda frente chamada *evolutionary*, que pretende a evolução do que já existe com o intuito de não criar incompatibilidades nas redes, quer seja de hardware ou software, o que seria inevitável caso seja criado uma nova infraestrutura, como pretendem os defensores da frente *clean slate* [16].

2.3 Conceitos

O paradigma SDN propõe a separação da rede em planos de dados (*data path*) e plano de controle (*control path*). Surgem então novos elementos, pois nesta arquitetura é definida a existência de um servidor dotado de grande capacidade computacional denominado de Controlador, o qual é o elemento principal no *control path*, sendo responsável pelo controle e configurações dos comutadores.

Já no *datapath*, estão presentes os comutadores, tal como *switches* e roteadores, que recebem comandos do Controlador para determinar os fluxos da rede, com o qual se comunicam por meio do Protocolo *OpenFlow*, que promove a troca confiável de mensagens específicas, utilizando criptografia SSL, entre os planos da rede [17].

2.4 Soluções e Arquitetura SDN

A programabilidade e a virtualização da rede introduzem soluções para muitos problemas, uma destas soluções é a *Cloud Computing* (computação em nuvem) que representa ganhos relacionados a escalabilidade, a qual implementa uma infraestrutura que

abre mão de volumosos servidores locais, os quais são substituídos pelo uso de softwares de rede que se comunicam por meio de APIs com um controlador SDN, como visto na Figura 1, tais servidores apresentam armazenamento robusto, oferecendo recursos e serviços acessados a partir de um *data center* remoto.

Estes serviços são hospedados em servidores capazes de hospedar e virtualizar inúmeras aplicações, permitindo ao usuário contratar serviços de nuvem e contar com a agilidade na montagem da infraestrutura que desejar, pagando apenas pelo serviço que de fato utilizar.

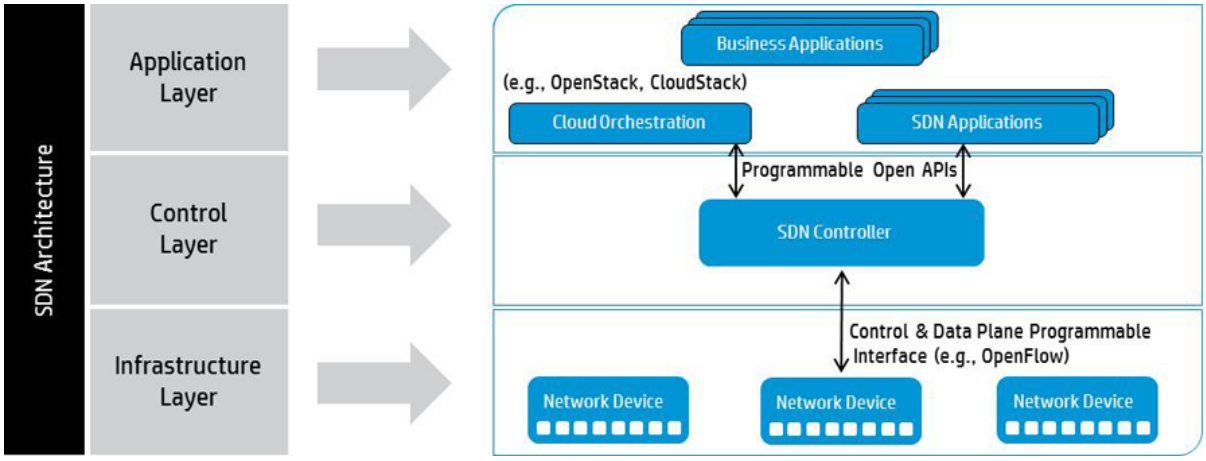

Figura 1: Arquitetura SDN [1]

A principal tecnologia que destaca-se em SDN é a proposta *OpenFlow*. Nela é transferido o controle dos dispositivos comutadores como: *switches* e roteadores, que na atual arquitetura operam sobre o tráfico e fluxo da rede, passando o processamento destas tarefas para os controladores SDN, que por sua vez passam as regras para os dispositivos comutadores. Os comutadores passam a ter menos necessidade de processamento e podem realizar tarefas de encaminhamento com maior velocidade, o que representa ganhos significativos de desempenho pela rede, pois estes passam a lidar apenas com fluxo de dados e não tomam decisões [18].

A proposta de um novo arranjo para a arquitetura da rede pode ser visualizada na Figura 2, onde é visto um comparativo entre o modelo da arquitetura tradicional, o qual mantém os plano de dados e controle operando em um só elemento da rede com desenvolvimento fechado pelo proprietário e portanto de evolução limitada, contrapondo-se ao contexto do paradigma de Redes Definidas por Software, que define a liberdade de desenvolvimento aberto e lógica de controle por intermédio de hardware de múltiplos fabricantes. Desta forma, a SDN propõe que a modelagem da rede deve conter basicamente os seguintes elementos:

- O plano de dados, onde encontra-se o comutador, dissociado do plano de controle;
- O controlador, principal componente do plano de controle, o qual concentra a "inteligência"desta arquitetura, sendo visto como sistema operacional de rede;

- As aplicações, as quais podem implementar, por exemplo: segurança, controle de acesso, algoritmos de roteamento e outras;
- O virtualizador da rede; e as
- APIs que implementam a comunicação entre controlador e as aplicações, e APIs interligando os planos da rede.

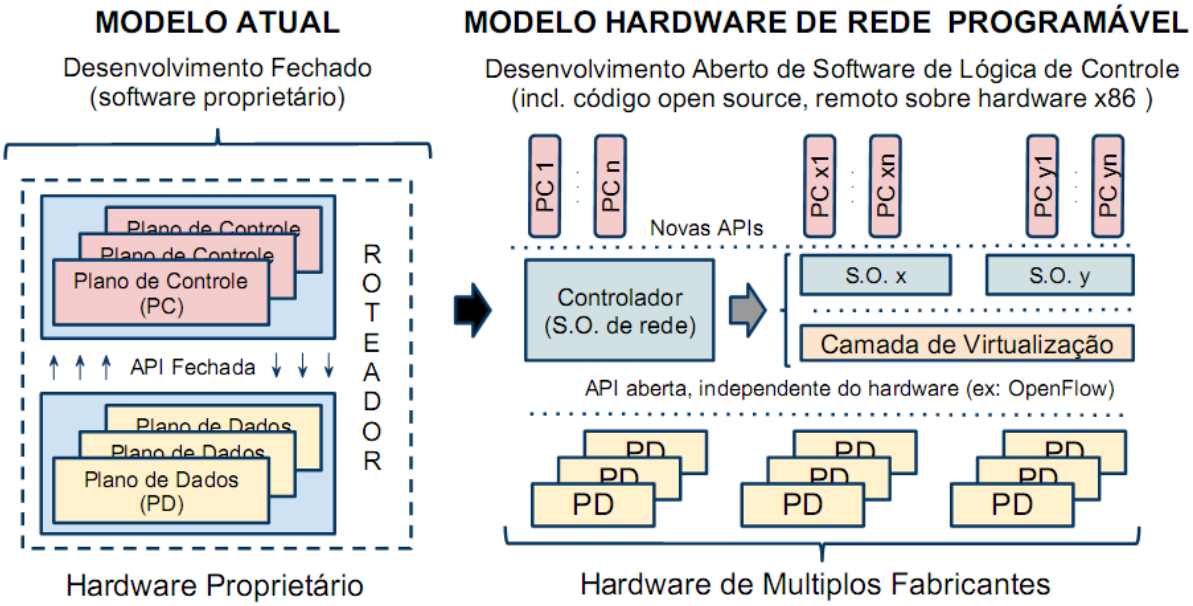

Figura 2: Arquitetura de Roteadores Tradicionais vs. Arquitetura de Modelo Programável [2]

Em SDN as instruções típicas de encaminhamento e rotas, encontram-se em aplicações instaladas no elemento principal - o controlador. Assim, o controlador passa a ser configurado pelo administrador de rede, o que garante maior flexibilidade para operá-las e gerenciar os fluxos de dados por meio de tais aplicações, centralizando no controlador, a capacidade de analisar os pacotes e tomar as decisões previamente configuradas.

Em suma, a tecnologia *OpenFlow* tornam as redes mais seguras e flexíveis, programáveis e com custos menos onerosos, minimizando os problemas de tráfego nos comutadores que apenas executam instruções processadas e passadas por outro elemento principal na rede [19].

2.5 SDN baseada em *OpenFlow*

2.5.1 Controlador

A seguir, na Figura 3, são ilustrados alguns dos principais controladores SDN, as linguagens em que são desenvolvidos, as plataformas em que dão suporte para execução e suas principais características.

Nome	Linguagem	Plataforma	Característica
NOX	C++, Python	Linux	Controlador de referência OpenFlow
POX	C++, Python	Windows, Mac, Linux	Evolução do NOX
Maestro	Java	Windows, Mac, Linux	Explora o paralelismo
Beacon	Java	Windows, Mac, Linux, Android	Multiplataforma, Multithreaded
Floodlight	Java	Windows, Mac, Linux	Pode-se integrar a redes não OpenFlow
Frenetic	Funcional/Declarativa Própria	Linux	Programa a rede como um todo
Onix	C++, Python, Java	Windows, Mac, Linux	Gerência redes de grande porte
SNAC	C++	Linux	Monitoramento de redes OpenFlow; Interface Web
Trema	C, Ruby	Linux	Script; Emulador

Figura 3: Controladores SDN [3]

Neste trabalho, utilizou-se para fins de validação, o controlador POX, uma vez que o tipo do controlador não impacta na proposta, justificando-se esta escolha devido a facilidade para instanciá-lo. Na Seção 2.6 descreve-se o controlador utilizado.

2.5.2 *OpenFlow Switch*

Grande parte dos roteadores e comutadores *Ethernet* possuem em sua implementação as *flow tables* (tabelas de fluxos), as quais são executadas em *line-rate*, que por sua vez implementam funcionalidades como: *Firewalls*, NAT, QoS e coleta de estatísticas [20]. A presença em SDN de um *OpenFlow Switch*, permite a utilização de um protocolo aberto, a partir do qual pode-se promover a programação da tabela de fluxos de diferentes *switches* e roteadores.

2.5.2.1 Canal Seguro

Este elemento da SDN, visa a troca segura de mensagens entre os comutadores e controladores, protegendo assim contra ataques mal intencionados e garantindo que essa comunicação entre os elementos de uma rede *OpenFlow*, ocorra de modo confiável e com baixas taxas de erros.

A ONF em sua especificação *OpenFlow* recomenda que seja utilizada para a interface de acesso do *OpenFlow Switch* ao controlador *OpenFlow*, a tecnologia SSL (*Secure Socket Layer*), no entanto é possível usar o TCP e o TLS (*Transport Layer Security*). Nestes casos, a porta 6633 foi registrada para acesso junto ao departamento IANA(Internet Assigned Numbers Authority), para que seja usada por padrão pela tecnologia *OpenFlow*.

Entretanto, outros serviços recentemente tem sido alocados para usar a porta citada no parágrafo anterior, por isso, devido a possíveis conflitos, não recomenda-se que a usem como padrão, isto porque a IANA tem resolvido este problema com a alocação de uma outra porta, a saber, a porta 6653, esta sim utilizada exclusivamente para as comunicações *OpenFlow* [21].

2.5.2.2 Tabela de Fluxo

Cada entrada na tabela de fluxos da rede é associada por regras, ações e controles de estatística. As regras podem ser definidas a partir da verificação dos valores de um ou mais campos no cabeçalho do pacote, sendo que por meio destes valores é determinado o fluxo.

As ações estão ligadas ao fluxo, sendo assim, um pacote ao ser processado, verifica se o mesmo será encaminhado ou descartado. Já os controles de estatísticas consistem de contadores que visam manter estatísticas de utilização e para remover fluxos inativos.

Portanto, a utilização da tabela de fluxos, permite a verificação do *Match* (Combinação) que contém informações de cabeçalho dos pacotes de rede [22]. Dessa forma, se houver de algum dos campos de cabeçalho, uma determinada entrada de fluxo é acionada e em seguida, os contadores relacionados a essa entrada de fluxo são incrementados e assim são executadas as ações mapeadas para a respectiva entrada de dados.

2.5.3 Protocolo *OpenFlow*

O protocolo *OpenFlow v1.0* é uma especificação de código aberto que possibilita a troca de informações e o gerenciamento entre o controlador remoto e o *swicth OpenFlow* (comutador). O uso desta tecnologia elimina a programação nos comutadores, que no contexto de uma SDN é repassada para o controlador *OpenFlow*, que por sua vez repassará por meio do protocolo OpenFlow todas as regras de encaminhamento ao comutador.

Este protocolo suporta três tipos de mensagens que são: *controller-to-switch*, *asynchronous*, e *symmetric*, cada uma destas mensagens possuem seus múltiplos sub-tipos. Com o uso de mensagens *controller-to-switch* um controlador pode por exemplo iniciar uma sessão com o objetivo de gerenciar ou analisar o estado do *swicth OpenFlow*. Já as mensagens *asynchronous* são iniciadas pelos comutadores sem solicitação do controlador, a fim de atualizá-lo a cerca de eventos na rede ou mudança de estado no *switch*. Por sua vez, as mensagens do tipo *symmetric* são enviadas em ambas as direções e exigem resposta, podendo ser iniciadas por qualquer um dos elementos OpenFlow citados, uma característica deste tipo de mensagem é que ela não precisa de solicitação prévia para ser iniciada [23].

De uma maneira mais prática, o *OpenFlow* determina:

- Como pode ser definido um determinado fluxo.

- As ações a serem realizadas, para cada pacote pertencente a este fluxo.
- Qual o protocolo de comunicação entre o controlador e os comutadores, que por sua vez realizam as definições de fluxos e ações.

2.6 Controlador POX

O Controlador POX, é considerado o irmão mais novo do controlador NOX que foi o primeiro controlador *OpenFlow* [24]. Sua implementação se baseia em seu antecessor, tendo sido desenvolvido com o intuito de servir para fins de pesquisa e ensino. Além de ser utilizado como controlador *OpenFlow*, o POX também pode ser usado como um *switch OpenFlow* [25].

O Pox é desenvolvido sob a linguagem *Python*, sendo este controlador usado como plataforma para o rápido desenvolvimento e prototipagem de software de controle de rede usando o *Python*. Esta proposta utilizou o POX sob a plataforma Debian 8, tendo como requisito que se esteja executando no mínimo a versão *Python* 3.

Esta Seção apresenta as principais características do controlador POX, descrevendo sua arquitetura bem como os elementos principais referentes a programação de seus componentes, de acordo com o que se encontra na página oficial do POX Wiki [10].

2.6.1 Instalação do POX

O código fonte do POX está hospedado no [26] github e para instalá-lo, deve ser criada uma cópia em sua maquina local. Para obter o POX no GNU/Linux é necessário que se crie uma cópia do respectivo git. Para se obter a instalação do git, é necessário usar o comando a seguir via CLI (Interface de linha de comando).

```
$ git clone http://github.com/noxrepo/pox
```

2.6.2 Executando o POX

Para iniciar o controlador POX é preciso que se execute o arquivo pox.py que se encontra na raiz de diretórios deste controlador, e para isso, basta digitar o seguinte comando "./pox.py"em um terminal *Shell*. Uma vez que o POX tenha sido iniciado corretamente, será retornado no terminal a seguinte mensagem:

```
POX 0.2.0 (carp) / Copyright 2011-2013 James McCauley, et al.
INFO:core:POX 0.2.0 (carp) is up.
```

Há ainda a possibilidade de que o usuário passe alguns argumentos ao se iniciar o POX, estes argumentos tem propósitos específicos e visam facilitar a manipulação deste controlador. A seguir na Tabela 1, serão descritas algumas opções úteis ao usuário:

Opções	Descrição
–verbose	Exibe informações extras, tendo papel na depuração onde se objetiva corrigir problemas.
–no-cli	Desativa a interface interativa.
–no-openflow	Não inicia o módulo *OpenFlow*.

Tabela 1: Tabela das Opções de Inicialização do Controlador POX [10]

2.6.3 Componentes do POX

Os componentes no POX são os programas que se executam no controlador fornecendo aplicações para a SDN de acordo com as políticas do componente. Não faz sentido assim que o POX seja executado sem um componente, por conta disso ele acompanha alguns módulos que servem como tutorial para os desenvolvedores que desejem criar seus próprios componentes.

Alguns desses componentes básicos com o intuito de introduzir o usuário a realizar seus próprios testes, e que acompanham o POX são:

- O *"py"*, que executa um depurador *Python* para depuração interativa;
- O *"forwarding.l2_learning"*, que faz com que os *switches OpenFlow* funcionem como *switches* de autoaprendizagem;
- Além do componente *"misc.of_tutorial"*, que é um tutorial *OpenFlow* para iniciantes.

É possível executar vários componentes no POX, para isto, é necessário especificá-los via CLI após qualquer uma das opções encontradas na Tabela 1, da seguinte forma:

```
$ ./pox.py ––no–cli [meu módulo 1]––[opção][meu módulo 2]––[opção]
```

Como visto no comando anterior, os componentes além de poderem ser executados juntos, podem ser passados juntos de opções como argumentos próprios em sua inicialização. Uma outra característica do POX é que seus componentes em geral deverão conter uma função chamada *launch()*, que por sua vez é a responsável por realizar todas as funções do componente, como instanciar classes ou chamar outras funções e métodos.

2.6.4 Núcleo do POX

O POX possui um módulo chamado de *"pox.core"*, cujo finalidade principal é de reunir as APIs do POX a fim de estabelecer conexões entre seus componentes. Com isto, por exemplo, é possível reduzir códigos de inicialização a medida que os componentes registrem suas funcionalidades no núcleo do POX, dessa forma, as funções destes componentes tornam-se disponíveis para outros sem a necessidade do recurso de *"import"* do Python.

Para se registrar um objeto no núcleo do POX, deve-se utilizar o "*core.registrer()*", que recebe como parâmetros dois argumentos, o primeiro é o nome que se deseja dar a esse objeto do componente, após registrado, o outro é o nome do objeto presente em algum componente que se deseje registrar no núcleo.

2.6.5 Eventos do POX

O POX em sua implementação é tido como orientado a eventos e estes tem seu sistema de controle implementados pela biblioteca *pox.lib.revent*, logo, objetos são vistos como eventos e podem disparar seus eventos ou até mesmo esperar por eles através de manipuladores denominados *handlers*. Os eventos do POX são todos os objetos instanciados de subclasses de *revent.Event*, deste modo, para que um objeto gere eventos, este deve herdá-los de *revent.EventMixin*.

2.6.6 OpenFlow no POX

O POX tem como seu principal propósito a prototipação de aplicações de controle OpenFlow, por esse motivo, nesta subseção destacam-se algumas das interfaces de controle OpenFlow do POX. Como descrito anteriormente, o OpenFlow no POX é iniciado automaticamente por meio do componente *openflow.of_01*, mas, como visto na Tabela 1, é possível anular isto ao se declarar em sua inicialização a opção *–no-openflow*.

Deste modo, o POX se comunica diretamente aos comutadores OpenFlow, para tanto, ele registra no núcleo do POX *(pox.core)* um objeto chamado *"openflow"*, assim, é possível utilizar este objeto para enviar e receber mensagens de controle para comutadores com o *OpenFlow* habilitado.

Os eventos são fundamentais ao se trabalhar com o POX pois por meio deles o OpenFlow pode registrá-los pelo componente *of_01*, a partir daí, outros componentes também podem observá-los, desde que hajam manipuladores *handle* devidamente registrados para estes eventos. Com isto, por exemplo, pacotes oriundos de comutadores *OpenFlow* e recebidos pelo objeto *openflow*, geram um evento *PacketIn*, deste modo, qualquer outro componente interessado na chegada de um pacote pode observar tal evento.

2.6.7 Mensagens OpenFlow no POX

As mensagens *OpenFlow* são a ferramenta de comunicação entre comutadores e controladores e estão presentes no POX tal como podem ser encontradas na especificação do OpenFlow, a especificação que trabalha com o POX faz referência a versão 1.0 do *OpenFlow* [10]. No POX são encontradas classes e constantes que correspondem a elementos do protocolo *OpenFlow*, podendo ser encontradas no arquivo *pox.openflow.libopenflow_01.py*.

2.7 Aplicações

Os controladores são vistos como sistemas operacionais de rede, com isto, temos que uma aplicação constitui-se de um software desenvolvido com a capacidade de se coordenar uma infraestrutura de rede física.

Dessa maneira, visualizamos a rede como um sistema unificado e integrado por meio de um sistema operacional gerenciador, portanto a atuação de uma aplicação SDN corresponde as políticas de configuração e interação entre os vários comutadores de rede, permitindo assim, a programabilidade da tabela de fluxos, algoritmos de roteamento, dentre outras finalidades.

A flexibilidade proporcionada por esse paradigma oferece uma estrutura lógica centralizada, abrangendo uma grande diversidade de ambientes de redes. Assim, pode-se considerar que o paradigma SDN, por meio também do conceito de aplicações de redes, proporciona a melhor organização das funcionalidades oferecidas em uma visão lógica completa de toda a rede.

2.8 Conclusões do capítulo

Neste capítulo destacou-se a importância das Redes Definidas por Software para o avanço da rede global de computadores, exibindo suas principais características e ressaltando o controlador POX que se torna referência para este trabalho de conclusão de curso, como validação da presente proposta.

CAPÍTULO 3

Trabalhos Relacionados

3.1 Visão Geral

Este capítulo apresenta uma visão geral sobre os temas relacionados com este trabalho de conclusão de curso (TCC). Aqui serão abordados os referenciais teóricos que serviram de embasamento para a elaboração deste trabalho. Uma breve descrição do RepoSDN será apresentada, bem como os principais sistemas para gerenciamento de pacotes no GNU/Linux, destacando suas vantagens e desvantagens, suas funcionalidades, o fluxo de trabalho, arquiteturas onde são utilizadas, além de fazer distinção entre as suas implementações. Por fim, será apresentada uma breve conclusão sobre este capítulo.

3.2 RepoSDN

Distribuir aplicações ainda é um problema pouco explorado. Para isto, o RepoSDN define um modelo de repositório para aplicações SDN, tal modelo propõe um método que permite especificar e modelar todo o processo para obtenção de uma aplicação, incluindo políticas de segurança a serem seguidas pelos desenvolvedores das aplicações SDN [4].

A grande contribuição do RepoSDN é centralizar a distribuição de aplicações a partir de um repositório e com isto convencionar o uso de aplicações SDN, haja vista que os modelos convencionais de distribuição de software não satisfazem as necessidades do perfil de usuários SDN, pois não tratam da obtenção dinâmica de novos serviços de redes [27].

Além da hospedagem de aplicações SDN, o RepoSDN propõe uma relação de confiança para com os desenvolvedores, uma vez que se torna responsável por dar suporte

e proteção aos direitos autorais, antes de tornar publicas tais aplicações, lhes garantindo que seu trabalho será distribuído respeitando a legislação vigente.

O fluxo de trabalho do RepoSDN pode ser visualizado na Figura 4, onde é possível ver a representação de todo o caminho percorrido pelas aplicações SDN até chegar aos controladores. Para isto é necessário compreender que qualquer SDN deve ser capaz de utilizar o RepoSDN como única fonte de aplicações.

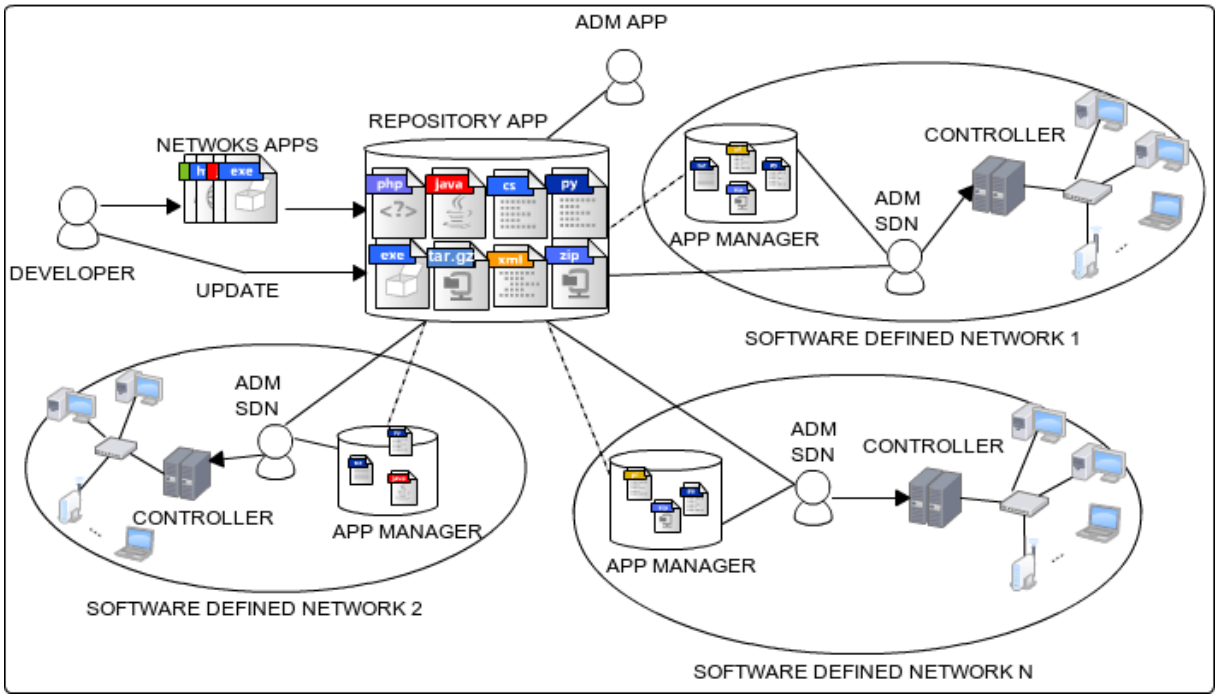

Figura 4: Fluxo de Trabalho do RepoSDN [4]

O elemento ADM SDN, visto na Figura 4, é o administrador da SDN, ele é quem acessa ao repositório em busca de aplicações e assim as salva em sua SDN no local chamado de APP MANAGER, este local guarda uma coleção de aplicações que estarão disponíveis para serem utilizadas de acordo com sua funcionalidade.

Um dos elementos presentes no fluxo de trabalho do RepoSDN é o Midcontroller [4], que trata-se de um elemento definido para guardar as configurações da SDN e nele devem estar presentes as ferramentas de acesso ao repositório para atuar no gerenciamento de informações das aplicações. Com ele é possível automatizar a comunicação dos elementos presentes em APP MANAGER junto à um controlador SDN. A proposta deste TCC é prover uma implementação dos componentes do Midcontroller, onde foi elaborada a ferramenta SPM [9] que encontra-se detalhada no Capítulo 4.

3.3 Sistemas Gerenciadores de Pacotes

Em sistemas Linux observa-se a importância em padronizar o empacotamento de software, como garantia de compatibilidade destes pacotes com uma ferramenta gerenci-

adora, o que permite também a sua distribuição centralizada a partir de repositórios. Os pacotes mais utilizados são aqueles com as extensões .deb ou .rpm, que formam a base de aplicações entre as distribuições mais aceitas pelos adeptos de sistemas livres [28]. Cada uma destas distribuições podem utilizar ferramentas próprias para servir como *front-end* para sistemas gerenciadores de programas Linux.

O acrônimo RPM faz alusão a *Red Hat Packet Manager*. Este é o nome dado ao sistema de gerenciamento de programas nativo em plataformas Red Hat, como o Fedora. O RPM faz parte da LSB (*Linux Standard Base*), é portanto um dos sistemas de pacotes definidos para oferecer um padrão comum de gerenciamento em plataformas Linux.

A Red Hat estabelece como padrão os seus pacotes rpm, cujo conteúdo são softwares prontos para serem instalados e executados e que são disponibilizados a partir de seus repositórios. Além do Fedora, é comum o uso do rpm também em distribuições como: OpenSuse, CentOs, IBM AIX, *Scientific* Linux, e Oracle Linux [29].

Por sua vez, o acrônimo DEB faz referência aos pacotes binários que foram definidos como o formato para a extensão de arquivos originalmente desenvolvidos para versões Linux baseadas no Debian. Um pacote deb contém dois conjuntos de arquivos: um conjunto de arquivos utilizados durante a instalação, e um segundo conjunto que proveem metadados, que serão utilizados tanto em sua instalação como em seu controle, a fim de fornecer informações adicionais a cerca da aplicação [30]. Além do Debian, os pacotes deb estão presentes em distribuições como o Ubuntu e o Mint, dentre outras.

3.3.1 APT

O APT é um projeto amplo composto por ferramentas com a finalidade de gerenciar pacotes, ele inclui o apt-get que foi desenvolvido como um utilitário CLI para distribuições baseadas no Debian, além dos demais programas: apt-cdrom, apt-cache, apt-config.

O apt-get é utilizado para realizar o *downloads* de aplicações para o sistema, desde que esteja conectado à Internet, pois ele procura pelos pacotes em seu arquivo de controle, que é encontrado em /etc/apt/sources.list [31]. Essa lista, contém linhas que são links para os repositórios *online* do Debian, que devem estar indicados no sources.list.

Toda vez que um novo repositório for indicado a partir do sources.list, o usuário deve executar o comando *apt-get update*, com isto, o sistema do APT estará configurado para procurar pacotes neste novo repositório. Além de servir inclusive para atualizar o próprio Linux, o apt-get serve o seu usuário com algumas opções muitas vezes importantes dependendo da situação, com as que seguem:

- -h - ajuda
- -d - baixar arquivos apenas, não instalar
- -f - conserta erros de instalações de pacotes

- -s - não agir, apenas simular operação
- -y - assume 'sim' para todas as perguntas
- -u - mostrar pacotes que serão atualizados também

Há também dentre as funcionalidades do apt-get as opções de armazenar pacotes para que sejam instalados posteriormente, neste caso, os pacotes serão guardados em: /var/cache/apt/archives. Eventualmente o usuário poderá atualizar pacotes e o apt-get faz isso por meio da combinação dos comandos *update* e *upgrade*. Por vezes o usuário precisa remover um pacote instalado em seu sistema, ele poderá fazê-lo por meio do comando *remove*.

3.3.2 dpkg e rpm

O dpgk é o utilitário responsável por gerenciar pacotes DEB em distribuições baseadas no Debian, tendo como finalidade a: instalação de aplicativos, instalação de ferramentas, *codecs*, *plugins* e outros. No entanto, o dpkg atua sobre pacotes já disponíveis no sistema, ou seja, ele não realiza downloads de pacotes e portanto não satisfaz dependências, caso existam.

Os pacotes DEB são gerenciados por comandos dpkg acompanhados de algum argumento válido como (dpkg -i ou –install) que realizam a instalação de pacotes. Como citado anteriormente o dpkg é o comando básico para lidar com pacotes DEB no sistema [32]. É possível escolher dentre algum dos argumentos disponíveis, que podem ser visualizados via terminal Linux através do comando dpkg –help, que exibe todas os demais comandos da ferramenta.

Além disso, o dpkg faz a manipulação de arquivos de configuração os quais servem para exibir informações, como por exemplo, no recurso de consulta ao seu *database* onde é possível que o dpkg retorne ao usuários, todos os arquivos instalados, estado de instalação, cabeçalhos de um determinado pacote instalado, bem como o detalhamento de todas as ações do dpkg dentro do sistema, através de seu arquivo de *log* e etc.

O rpm atua sobre pacotes RPM, sendo uma ferramenta de uso limitado, que assim como o dpkg, também atua somente sobre pacotes que tenham sido conseguidos previamente em alguma mídia ou baixados da Internet, sem sua intervenção. Para consulta aos pacotes no sistema, o rpm reúne em seus bancos de dados as tags associadas a um pacote especificado, assim o usuário pode ter acesso às informações desde que escolha um dos comandos da ferramenta.

Além disto, existem ferramentas gráficas que disponibilizam um front-end para uruários Linux como o Synaptic, este usa os utilitários dpkg e rpm para gerenciar pacotes de software. O synaptic é um projeto desenvolvido pela equipe técnica do Mandriva Linux e tem como base o GTK+ que fornece um toolkit para criação de interfaces gráficas, sendo

um padrão integrado pelo projeto GNU. A seguir na Figura 5, é possível visualizar uma tela do front-end Synaptic.

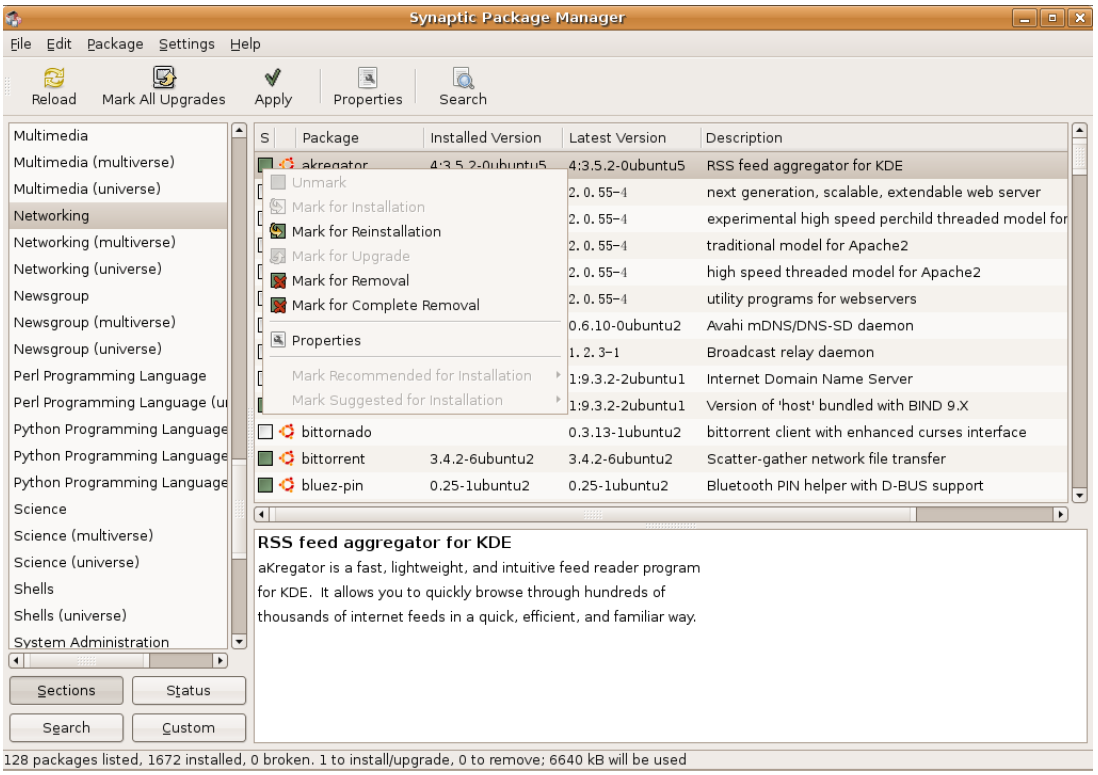

Figura 5: Synaptic, um Front-End para Gerenciamento de Pacotes .deb [5]

3.3.3 YUM

O YUM é o acrônimo para (*Yellow Dog Updater Modified*). É uma ferramenta avançada para o gerenciamento de pacotes presente por padrão no *Fedora Core*, porém é possível de ser utilizado nas demais distribuições *Red Hat*, uma vez que se busque os pacotes de instalação do mesmo [33].

Este gerenciador assemelha-se bastante ao apt-get do Debian que é capaz de resolver dependências referentes a pacotes DEB. O yum realiza esta função sobre pacotes RPM através de um arquivo de configuração para buscar por URLs que apontem para o software desejado em seus reposiórios.

Desta forma, assim como o apt-get busca programas a partir de um arquivo que contém os *links* para os repositórios, o yum usa o arquivo localizado em /etc/yum.conf. Semelhantemente ao sources.list do Debian, o yum.conf, também se encontra em diretórios de acesso restrito, sendo portanto obrigatório que um usuário possua privilégios *root* para poder alterá-los. A seguir na Figura 6, apresenta-se o *PackageKit*, uma interface gráfica para o sistema yum.

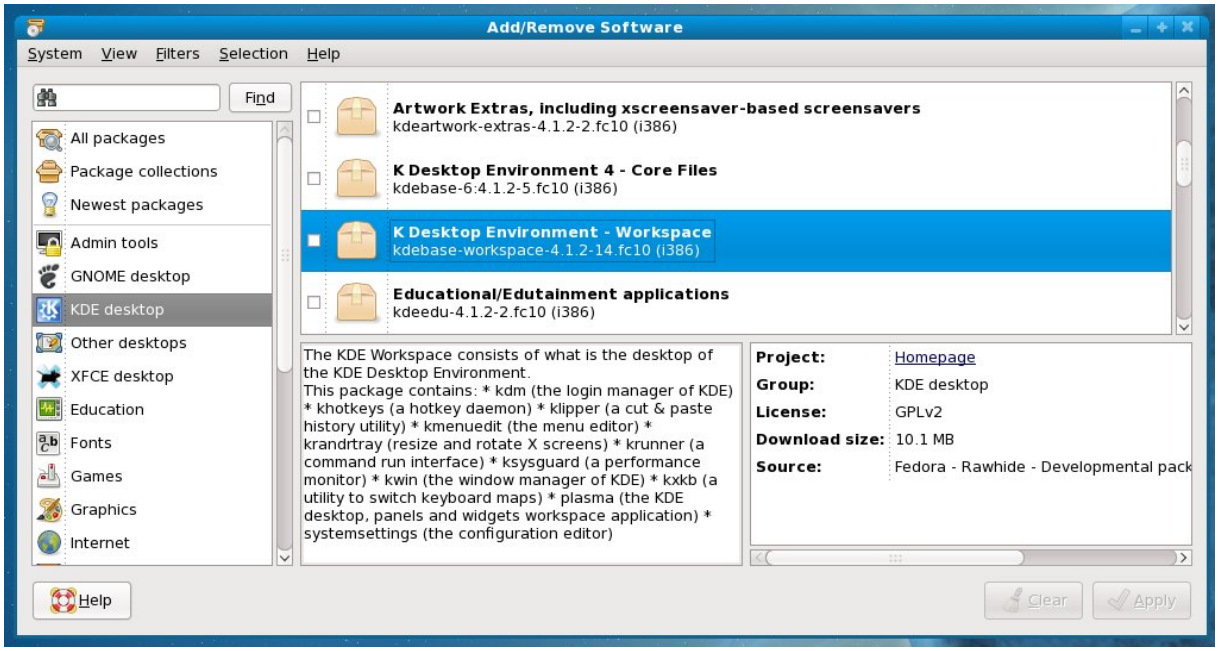

Figura 6: PackageKit, um Front-End Intimamente Integrado ao Fedora para o Gerenciador de pacotes YUM [6]

3.3.4 PKGTOOL

O pkgtool, faz parte da distribuição GNU/Linux Slackware, criada e mantida por Patrick J. Volkerding, esta distribuição usa em seu sistema de pacotes arquivos comuns, criados com o uso do gzip. O pkgtool tem compatibilidade com pacotes do tipo tar.gz e .tgz, sendo responsável por gerenciá-los em modo texto através de comandos e parâmetros que realizam a instalação, atualização ou remoção de softwares dentro de distribuições [7].

Como pode ser visto na Figura 7, o pkgtool possui uma interface gráfica amigável. Além disso, este sistema é capaz de montar um novo pacote compatível por meio da funcionalidade makepkg, além de converter um pacote RPM em um pacote compatível com o Slackware, por meio do comando *rpm2tgz*. Os seus demais comandos podem ser encontrados ao acessar-se a página oficial do Slackware sobre o pkgtool, que encontra-se disponível no domínio *slackware.com*.

O maior repositório para binários do Slackware se encontra em linuxpackages.net. No Slackware a lista de pacotes instalados pode ser vista em /var/log/packages. O conteúdo de um pacote Slackware é modelado da seguinte forma:

```
./
usr/
usr/bin/
usr/bin/makehejaz
usr/doc/
usr/doc/makehejaz-1.0/
usr/doc/makehejaz-1.0/COPYING
```

usr/doc/makehejaz-1.0/README
usr/man/
usr/man/man1
usr/man/man1/makehejaz.1.gz
install/
install/doinst.sh

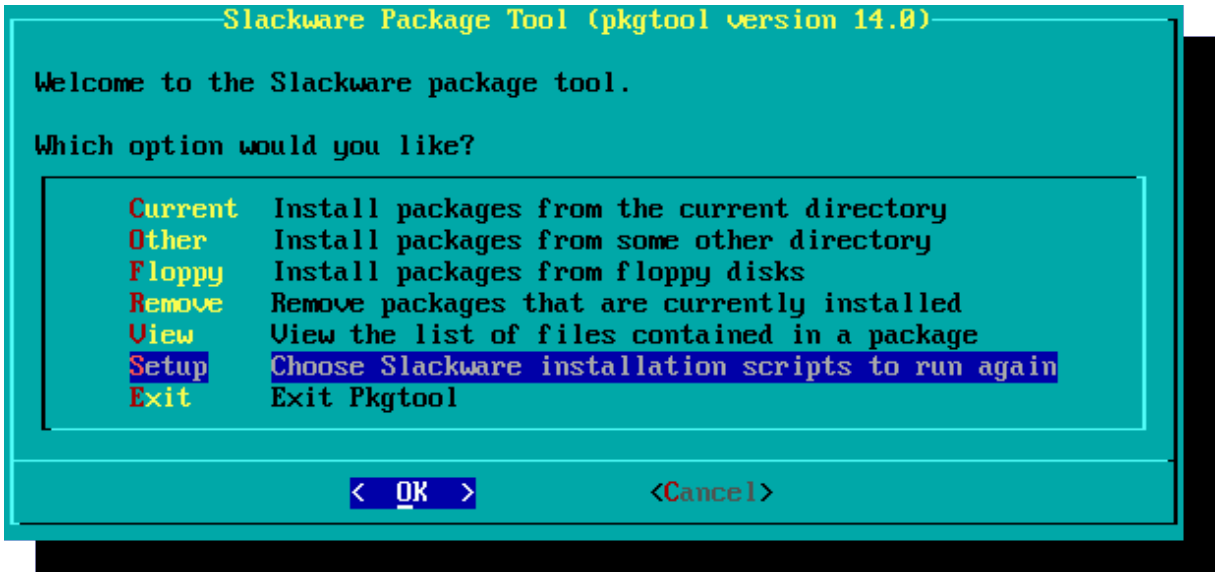

Figura 7: Sistema Pkgtools - o Gerenciador de Pacotes do Slackware [7]

3.3.5 YaST Software Management

O Libzypp é a solução do GNU/Linux OpenSUSE, fornecendo um sistema de gerenciamento de pacotes, sua interface gráfica é o software de gerenciamento YaST, o qual pode ser visualizado na Figura 8, por sua vez, o Zypper atua como sua interface de linha de comando. Com estes utilitários o OpenSUSE realiza a instalação, atualização ou remoção de pacotes instalados nesta distribuição [8].

O gerenciamento de pacotes neste caso é feito por meio do recurso de metadados que servem como informações adicionais compactadas junto de qualquer aplicação do OpenSUSE. Como nos demais sistemas, o OpenSUSE também mantém repositórios online, onde são hospedados os seus pacotes. Estes pacotes podem ser dos tipos:

- **tgz**: sendo basicamente arquivos de código fonte que utilizam esta compactação, não havendo padronização em seu conteúdo, desta forma, o mantenedor do pacote tem a liberdade de inserir qualquer coisa que lhe pareça útil. Estes tipos de pacotes tgz devem ser compilados para a execução do software.
- **rpm**: estes são pacotes criados pela Red Hat Linux e fazem parte da LSB, sendo utilizado atualmente por diversas distribuições Linux como sistema de embalagem.

- **deb**: os pacotes DEB por sua vez, são arquivos pré-compilados e também são compatíveis.

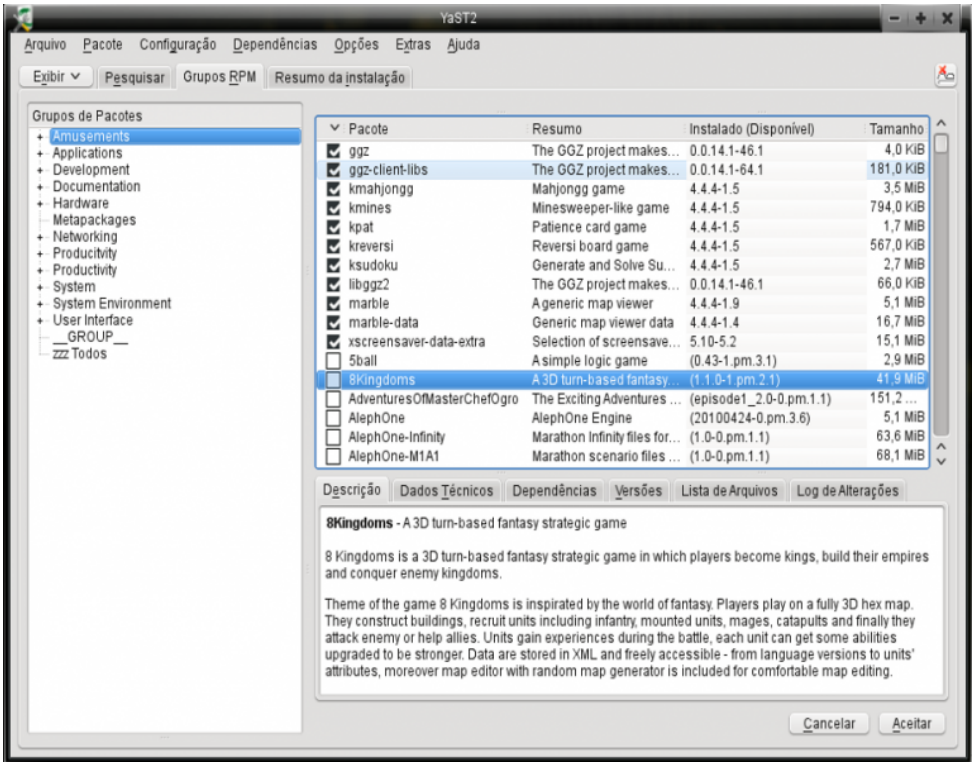

Figura 8: Fromt-End, Yast - o Gerenciador de pacotes do openSUSE [8]

3.4 Conclusões do capítulo

O presente Capítulo mostrou os Trabalhos Relacionados que serviram de inspiração para a criação de um modelo de gerenciamento para aplicações aplicado ao contexto de SDN.

CAPÍTULO 4

Software Package Manager for SDN Applications - SPM

4.1 Visão geral

Como visto no Capítulo 3, o uso de ferramentas que facilitem a manipulação de pacotes de aplicações é uma necessidade recorrente nos mais variados sistemas operacionais.

A manutenção das aplicações em cada controlador SDN, dá-se de modo independente, uma vez que controladores SDN não são acompanhados de ferramentas que gerenciem os módulos de aplicações instalados, ou seja, o tratamento destas aplicações ocorre de modo individual, dependendo da linguagem em que o controlador foi implementado.

No cenário atual, caso o administrador da rede não disponha de uma ferramenta gerenciadora de aplicações, caberá a ele identificar todos os controladores em uso na rede, as aplicações que rodam em cada um destes controladores e assim manipular, desta forma convencional e pouco escalável, todos estes dados.

Assim, após investigar e constatar a presença atual desta lacuna, verificou-se a necessidade pelo uso de uma ferramenta específica que torne estes métodos transparentes ao administrador da SDN, de modo a facilitar a aquisição de novas aplicações de redes para um ou mais controladores.

Uma vez que em uma SDN podem conter diferentes tipos de controladores, a adoção de uma ferramenta com esta finaildade, está relacionada com a otimização de tempo, sendo sua principal funcionalidade ocultar os métodos para que um controlador consiga uma nova aplicação, pois trata-se uma tarefa complexa ter que gerenciar vários

controladores de uma SDN e manipular as diversas aplicações instaladas em cada um deles.

Neste contexto, com inspiração nos já consagrados gerenciadores de pacotes do GNU/Linux e dada a necessidade pelo aperfeiçoamento destes sistemas para o ambiente dos controladores SDN, propõe-se a ferramenta Software Package Manager for SDN Applications - SPM, a qual oferece, ao gerenciador da SDN, um método auxiliar para o gerenciamento de aplicações nos controladores.

Com esta definição os elementos do plano de controle SDN se tornam gerenciáveis, pois a ferramenta SPM automatiza a manipulação de aplicações e reúne de forma escalável e simplificada as informações a cerca de vários controladores e suas aplicações dentro de uma SDN.

A ferramenta SPM [9] propõe um método aberto que define a padronização no empacotamento de aplicações hospedadas no servidor, centralizando a sua distribuição a partir de um repositório Web, onde estes pacotes devem seguir a modelagem definida por esta proposta, como pode ser visto na Seção 4.4.

Os pacotes ZIP hospedados no repositório Web, tem o objetivo de prover a compatibilidade com a ferramenta SPM e de manter uma coleção de pacotes padronizados, desse modo, disponibilizando uma infraestrutura que impulsione o desenvolvimento de novas aplicações, além de permitir que a ferramenta SPM realize a configuração de aplicações em cada controlador, funcionando como interface entre a SDN e o servidor na Web.

Estas aplicações SDN podem implementar diversas funcionalidades, dentre elas destacam-se: aplicações de segurança (*Firewall*), algoritmos de roteamento (OSPF ou RIP), mecanismos para controle de congestionamento (*Fast Recovery e Fast Retransmit*) e tantas outras.

Um administrador de rede ao configurar a ferramenta SPM em sua SDN, poderá utilizar-se de suas funcionalidades com o uso de algum dos comandos disponíveis, os quais estão descritos na Seção 4.5, estes comandos visam retornar a resolução de tarefas comuns neste contexto repositório-aplicação-controlador, como por exemplo: buscar, adicionar, remover ou atualizar aplicações presentes no *controlpath* SDN. Os principais benefícios com o uso da ferramenta SPM são:

- Poupa tempo de treinamento e/ou leitura na documentação do controlador para aprendizado do seu funcionamento no tocante a instalação de uma nova aplicação;
- Otimiza a rede, uma vez que, com esta definição não há a necessidade de se deslocar a cada servidor, podendo acessá-los remotamente, e gerenciar as aplicações desde que a ferramenta SPM esteja pré-configurada;
- Torna transparente ao administrador da rede a gerencia das aplicações de rede bem como a sua: instalação, remoção ou atualização de uma ou mais aplicações permitindo realizá-las remotamente em cada controlador;

- Reduz o tráfego de dados na rede uma vez que armazena o histórico de downloads de aplicações em um repositório local da SDN, tornando desnecessário o uso da Internet para buscar outra vez uma aplicação que já tenha sido utilizada anteriormente, pois, uma vez que se baixe uma aplicação uma cópia desta será armazenada no repositório local da rede no endereço: /var/lib/appmanager/;
- Mantém o administrador da SDN atualizado, pois ao verificar o banco de dados da ferramenta SPM ele terá acesso às informações sobre quais aplicações cada controlador possui, isto é possível por meio de arquivos de configurações que reúnem metadados com tais informações.

4.2 Componentes do Sistema

Como se trata de uma ferramenta que atua no gerenciamento de elementos do plano de controle SDN, o tipo do controlador (NOX, POX, Beacon e outros), bem como a sua linguagem não influenciam no tema abordado. Por esse motivo foi escolhido o controlador POX para a realização dos testes com a ferramenta SPM. O POX é um controlador criado para rápida prototipação de software e por isso é bastante popular em pesquisas a cerca de SDN.

A ferramenta SPM pode ser analisada nesta proposta como um dos componentes de um sistema onde atua como elemento principal, a mesma funciona como um agente que se comunica com os demais elementos deste sistema, o qual pode ser divido da seguinte forma:

- SDN repository: repositório onde são hospedadas os pacotes sob compactação ZIP para diferentes aplicações SDN as quais seguem a padronização definida nesta proposta;
- Application developer: é o agente responsável por desenvolver as aplicações SDN e enviá-las ao servidor Web;
- SDN administrator: é o administrador da SDN, sendo este o elemento responsável por gerenciar aplicações no controlador com o uso da ferramenta SPM;
- SPM: ferramenta desenvolvida nesta proposta para gerenciar aplicações SDN e que faz interface entre o servidor Web e elementos do plano de controle SDN;
- Controlador POX: implementado em linguagem Python, trata-se de um controlador SDN para prototipação descomplicada de novas aplicações de redes, por essas e outras o POX é muito utilizado em pesquisas que promovam investigações para Internet do Futuro.

Para a elaboração desta proposta foi necessário identificar o modo como as aplicações são manipuladas no POX. Com isto observou-se que o mesmo não possui uma

solução que trate isso de forma automatizada e transparente para o administrador da rede. Então a adoção da ferramenta SPM junto ao ambiente do controlador se dá de forma a expandir funcionalidades que não estão presentes em seu projeto inicial, com o objetivo de satisfazer a gestão simplificada de aplicações SDN e dos controladores, independentemente de sua arquitetura ou implementação.

A seguir na Figura 9 pode-se ver o fluxo de eventos da utilização da ferramenta SPM, compreendendo desde o desenvolvedor que envia a aplicação ao repoitório Web para aplicações SDN e passando pelo administrador da SDN que irá executar a ferramenta via CLI a fim de disponibilizar ou remover aplicações de um de seus controladores.

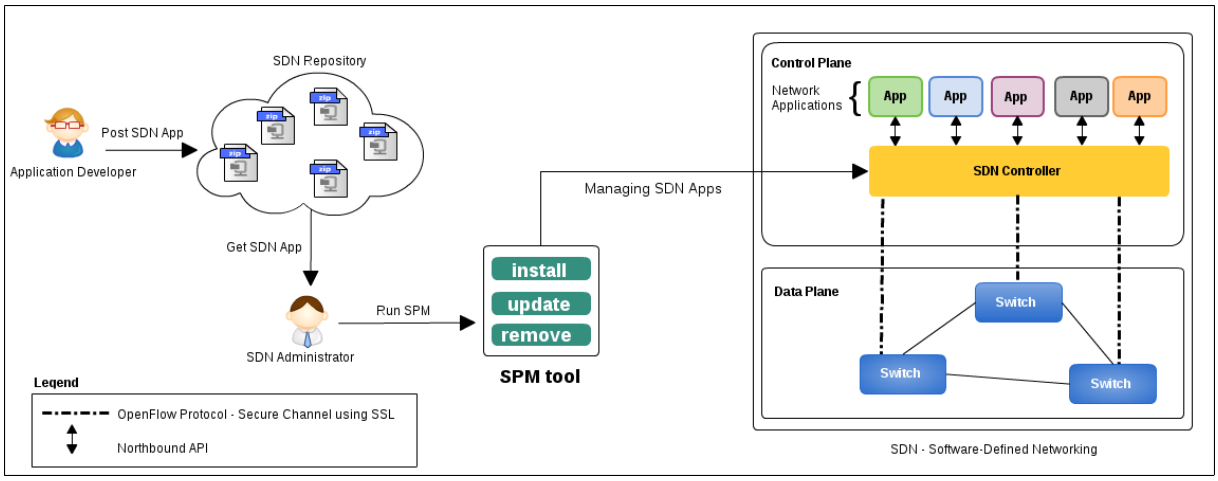

Figura 9: Fluxo de Trabalho SPM [9]

4.3 Arquitetura do SPM

Nesta Seção será detalhada a arquitetura da ferramenta SPM, onde apresenta-se os diagramas UML (*Unified Modeling Language*) para que com o uso dessa abordagem, permita-se destarcar o: caso de uso e o diagrama de sequências da ferramenta SPM. O uso dessa abordagem tem como objetivo auxiliar na visualização dos eventos e na relação de cada ator com o sistema, demonstrando assim, suas respectivas funções.

4.3.1 Diagrama de Caso de Uso

A Figura 10 apresenta o diagrama de caso de uso da ferramenta Software Package Manager for SDN Applications - SPM, onde é permitido ao administrador SDN selecionar dentre as opções de comandos disponíveis para interagir com o servidor Web de aplicações e o controlador SDN.

A ferramenta SPM acessa ao repositório Web em busca da lista de aplicações que lá são hospedadas, a partir disso a ferramenta poderá buscar ou não uma aplicação na Web, isto porque se a aplicação desejada já tiver sido utilizada anteriormente a mesma será encontrada no repositório local da SDN.

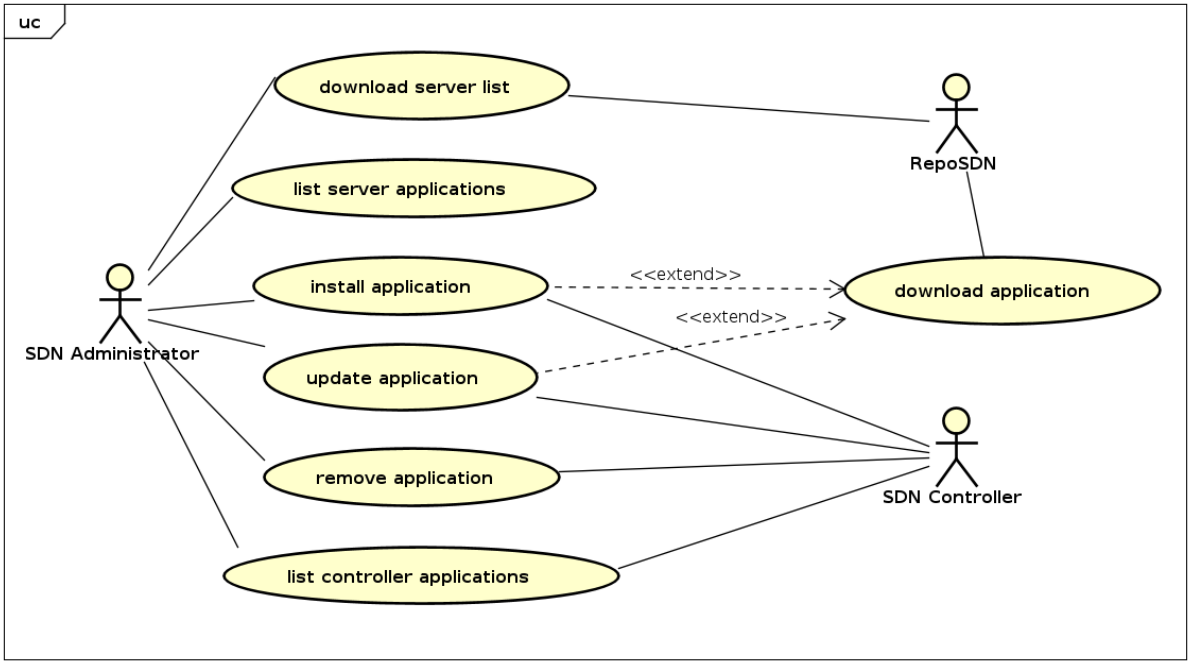

Figura 10: Caso de Uso da ferramenta Software Package Manager for SDN Applications - SPM [9]

4.3.2 Diagrama de Sequência

Com o uso da ferramenta SPM, um administrador SDN acessa ao repositório Web utilizando o comando *spm download*, com o qual receberá uma lista contendo os nomes de aplicações hospedadas neste repositório, feito isto a ferramenta SPM disponibiliza o comando *spm repolist*, para listar os pacotes hospedados no servidor Web, esta lista é descrita na Sub-seção 4.4.3.5. A partir dessa lista o administrador SDN irá utilizar o comando *spm install* para selecionar e baixar uma aplicação para o *appmanager/*, este por sua vez é um diretório que hospeda localmente as aplicações da SDN e será descrito na Sub-seção 4.4.3.4.

Após baixar um pacote de aplicação, a ferramenta SPM poderá ser utilizada pelo administrador SDN para acessar o *appmanager/* a fim de escolher uma aplicação que será instalada em um controlador de sua SDN. A partir do *appmanager/* a ferramenta SPM também pode ser utilizada para atualizar as aplicações de uma SDN por meio do comando *spm update*. Por fim, poderá remover uma aplicação de seu controlador através do comando *spm remove* e então listar as aplicações de um dos seus controladores ao utilizar o comando *spm list*.

O fluxo de execução destes comandos pode ser visto a seguir na Figura 11, por sua vez, os comandos da ferramenta SPM estão descritos mais adiante, na Seção 4.5.

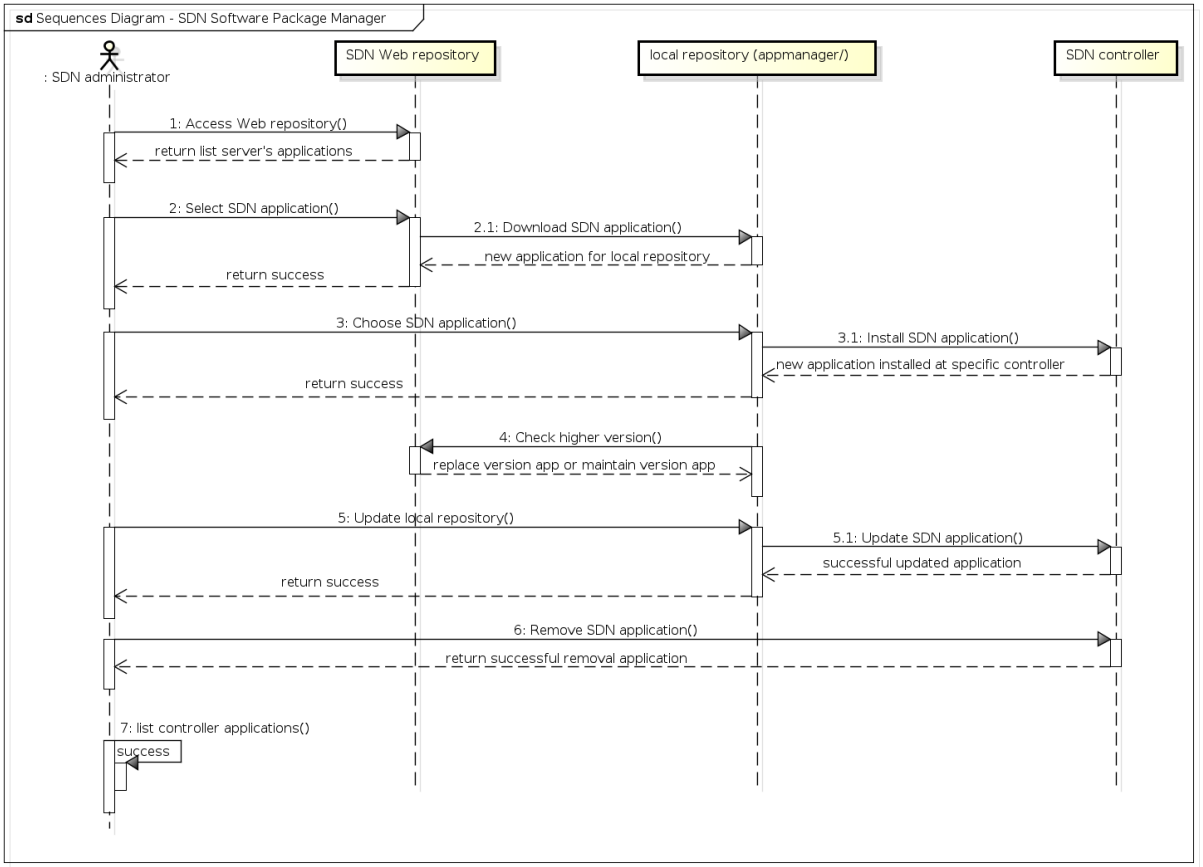

Figura 11: Diagrama de Sequência da ferramenta Software Package Manager for SDN Applications - SPM [9]

4.4 Detalhamento da Ferramenta SPM

4.4.1 Definição dos Nomes dos Pacotes

O nome de uma aplicação deverá ser exclusivo e para se tornar compatível com a ferramenta SPM este nome terá que obedecer a seguinte ordem de nomenclatura: o nome da aplicação, o controlador a que se destina e a versão do pacote de aplicação, devendo estas informações básicas estarem separadas por um traço (-), seguidos da extensão que representa a compactação (.zip), tal como o exemplo seguinte:

EX: App-Pox-1.0.zip

Todo pacote hospedado no servidor Web deve ter a nomenclatura citada anteriormente, caso contrário, não terá compatibilidade com a ferramenta SPM. Cada uma das partes que compõem o nome do pacote está descrito a seguir:

- **App** - leia-se o nome da aplicação.
- **Pox** - leia-se o tipo de controlador a qual foi desenvolvida.
- **1.0** - leia-se a versão do pacote da referida aplicação.

- **Zip** - a extensão que representa a compressão definida como padrão nesta proposta.

4.4.2 Padronização de Pacotes de Aplicações SDN

A definição do conteúdo de um pacote para aplicações SDN é fundamental para que a distribuição e o seu gerenciamento possam ser automatizados, o que torna escalável e simplificada a gestão de aplicações nos controladores da SDN. Com a padronização, a ferramenta SPM passa a gerenciar as aplicações da SDN a partir da administração do plano de controle, isto ocorre devido a compatibilidade entre o pacote e a ferramenta SPM.

A ferramenta SPM sempre irá buscar pelos seguintes arquivos presentes no pacote padronizado, que são: package.info, app, README. Cada um destes elementos serão descritos mais adiante nesta Seção. A seguir na Figura 12 é demonstrado um modelo de um pacote definido por esta proposta.

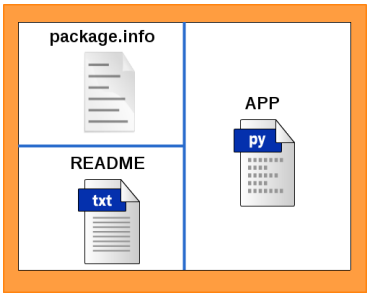

Figura 12: Modelo de Pacote SPM

4.4.2.1 package.info

O arquivo package.info reúne os metadados de um pacote de aplicação, estes metadados são informações passadas pelo desenvolvedor e que posteriormente serão extraídos para compor o arquivo metasdn.list. Neste arquivo estão disponibilizados 11 campos que correspondem à informações adicionais sobre o pacote as quais são definidas para auxiliar a ferramenta SPM no gerenciamento das aplicações.

Todos os campos que compõem o arquivo package.info são fundamentais nesta proposta, de modo que tanto a não observância de sua ordem, como também a falta de qualquer um dos mesmos, implicará no mal funcionamento da ferramenta SPM. Os metadados de um pacote estão descritos a seguir:

- A primeira linha deve conter o nome da aplicação.
- A segunda linha iniciada por Version-app deve conter a versão da aplicação.
- A terceira linha iniciada por Maintainer deve conter o endereço eletrônico do desenvolvedor.

- A quarta linha iniciada por Packet-size deve conter o tamanho do pacote contendo a aplicação.
- A quinta linha iniciada por Dependeces-app deve descrever as dependências da aplicação.
- A sexta linha iniciada por Name-controller deve conter o tipo de controlador (Floodligth, Nox, Pox).
- A sétima linha iniciada por Version-controller deve conter as versões do tipo do controlador compatível com a aplicação.
- A oitava linha iniciada por Architecture deve conter a arquitetura do sistema operacional compatível com a aplicação (32 ou 64 bits).
- A nona linha iniciada por Type deve conter o tipo de aplicação (topologia, roteamento, segurança, etc).
- A décima linha iniciada por Copyright deve conter a licença sob a qual é distribuída a aplicação.
- A décima primeira linha iniciada por Description deve ser precedida por uma descrição sobre o funcionamento da aplicação.

A seguir na Figura 13 será exibido o conteúdo presente no arquivo package.info, o qual foi extraído de um pacote da aplicação hub hospedada no repositório Web e que segue as definições contidas nesta proposta:

```
{
    "hub": {
        "Version-app": "1.0",
        "Maintainer": "http://www.noxrepo.org/pox/about-pox/",
        "Packet-size": "1.8 KB",
        "Dependeces-app": "no",
        "Name-controller": "pox",
        "Version-controller": "0.2.0",
        "Architecture": "all",
        "Type": "forward",
        "Copyright": "2012,2013 James McCauley",
        "Description": "It's a node with a huge number of links"
    }
}
```

Figura 13: Arquivo package.info da Aplicação hub

4.4.2.2 Módulo de instalação (APP)

Um módulo de instalação refere-se ao código fonte - no caso de aplicações desenvolvidas em linguagens interpretadas - ou a um binário - no caso de linguagens compiladas

- sendo que o app deve implementar os módulos que se comunicam com o controlador, ele representa a aplicação de rede, este arquivo pode ter extensões de linguagens como Python, Java, C/C++, dentre outras. Portanto a extensão desse arquivo depende do tipo de controlador para o qual a aplicação tenha sido desenvolvida.

4.4.2.3 Arquivo README

O arquivo README é um arquivo de texto que pode ser lido pelo usuário, este arquivo está relacionado a documentação do pacote, com ele o ADM SDN pode obter informações sobre a configuração, instalação, manual sobre o uso da aplicação, detalhes sobre a licença de distribuição, contato do distribuidor ou desenvolvedor, além de dependências da aplicação e informações reportadas pelo desenvolvedor a cerca de *bugs* conhecidos sobre a aplicação, assim como instruções sobre como proceder para reportá-los aos desenvolvedores.

4.4.3 Configurações SPM

Aqui estão descritos os componentes desenvolvidos com esta proposta, estes arquivos são fundamentais para o funcionamento da ferramenta SPM, pois com eles é realizado o gerenciamento sobre as aplicações no controlador da SDN a partir do repositório central.

4.4.3.1 spm.py

Este é o código fonte desenvolvido em linguagem Python o qual implementa a ferramenta SPM, ele deve estar localizado no caminho /usr/bin onde se encontram os demais programas do usuário.

4.4.3.2 installspm.sh

O SPM faz uso também de alguns diretórios no sistema, isto será descrito na Figura 14. Assim sendo, foi desenvolvido um instalador o qual fica responsável por configurar o SPM para que a ferramenta esteja disponível no Linux do usuário e o mesmo seja executado como qualquer outro programa.

O instalador do SPM é chamado de *installspm.sh*, que trata-se de um arquivo Shell Script desenvolvido para ser o responsável por preparar o sistema de modo que o SPM encontre os arquivos e diretórios descritos nesta proposta, pois com eles a ferramenta SPM pode gerenciar as aplicações no controlador.

Para que o SPM seja instalado o *installspm.sh* deve ser executado por um usuário com privilégios de *root* pois o mesmo acessa o /usr/bin/ que trata-se de um diretório de acesso restrito no Linux, com isto, ele implementa uma função para validar as permissões

do usuário corrente, assim, se este não tiver permissões será retornado uma mensagem de erro que alerta a impossibilidade de que a instalação continue.

Além disso é necessário que o instalador e o arquivo SPM se encontrem ambos no mesmo diretório para que seja realizada a configuração do SPM. No instalador também encontram-se as regras para que seja criado o repositório local denominado appmanager/, mais a diante na Figura 14, serão demonstrados os arquivos e diretórios manipulados pelo SPM.

Na Figura 14 fica apresentado os componentes da ferramenta SPM e que serão descritos após a Figura. As funções do instalador são.

- Verificar se o usuário possui privilégios de *root*;
- Criar o repositório local denominado appmanager/;
- Procurar o executável do SPM;
- Dar permissões chmod ao SPM;
- Instalar o SPM no diretório de programas do usuário.

4.4.3.3 spm.conf

O componente *spm.conf* é o responsável por armazenar atributos contendo informações necessárias ao funcionamento da ferramenta SPM, nele o administrador da SDN deverá registrar os seguintes atributos:

- **repository:** Faz referência ao endereço Web para o servidor;
- **cache:** Fornece o caminho para o repositório local chamando *appmanager/*;
- **controller:** Neste campo é possível registrar um nome específico cujo finalidade é identificar um determinado controlador, para isso é necessário especificar seu *path* que registra o caminho em que está instalado e o campo *type* que desigina o tipo do controlador (POX, Floodlight, Beacon, etc).

4.4.3.4 appmanager/

Este diretório chamado *appmanager/* é criado pelo instalador do SPM é nele que são mantidas localmente na SDN as aplicações uma vez baixadas do repositório Web, dessa forma, quando um administrador SDN necessitar de uma aplicação já utilizada anteriormente o SPM recorrerá primeiramente ao appmanager/, isto torna desnecessário que se acesse a Internet novamente;

4.4.3.5 reposdn.list

Este arquivo foi definido nesta proposta a fim de listar os nomes de todas as aplicações hospedadas no repositório Web, o usuário neste caso é o administrador SDN, o mesmo pode gerenciar uma aplicação por meio do nome da mesma através de argumentos passados juntos a um dos comandos SPM, sendo assim, o mesmo não precisará ler todos os outros nomes os quais não lhe interessem. Este arquivo terá o nome de *reposdn.list* sendo que seu conteúdo faz referência aos metadados contidos em cada arquivo package.info de uma aplicação hospedada no servidor.

4.4.3.6 metasdn.list

O repositório local também chamado de *appmanager/*, deverá conter um arquivo de configuração definido nesta proposta, este arquivo é denominado *metasdn.list*, sendo que em seu conteúdo há a lista dos nomes de aplicações contidas neste no repositório local da SDN.

Este arquivo é útil para o SPM pois nele constam informações acerca de todas as aplicações presentes no appmanager/, por sua vez, estas informações são extraídas do arquivo denominado package.info que está descrito na Subseção 4.4.2.1. A principal utilidade na definição do arquivo metasdn.list é dentre outras coisas, tornar ágil a identificação de versões das aplicações e caso necessário, atualizá-las para a versão mais atual.

4.5 Funcionamento da Ferramenta SPM

Nesta Seção serão descritos os *Command Line Interfaces* (CLI) que a ferramenta SPM disponibiliza para que o administrador da SDN possa gerenciar os pacotes de aplicações usados na rede. Ao abrir um terminal no GNU/Linux, o administrador da SDN poderá utilizar as soluções e com elas acessar aos arquivos de configuração descritos nas Sub-Seções 4.4.2.1, 4.4.3.6 e 4.4.3.5, os quais serão utilizados para tratar uma aplicação via CLI.

4.5.1 spm install

Com este comando o administrador da SDN conseguirá instalar uma aplicação no controlador POX, ou seja, com ele a tarefa se torna transparente e direcionada para um controlador especificado. Após o comando *spm install*, o administrador da SDN deve passar dois argumentos: o nome dado à um controlador onde se deseja instalar a aplicação, este nome deve estar de acordo com o que constar no arquivo *spm.conf*, o outro argumento é o nome da aplicação. A seguir temos um exemplo de utilização:

```
$ spm install ctr2 hub
```

4.5.2 spm update

O comando *spm update* visa retornar a atualização de uma aplicação previamente instalada em um dado controlador POX que também esteja registrado no arquivo *spm.conf*, isto significa que a ferramenta SPM verificará as versões das aplicações contidas no controlador e irá comparar com as versões contidas nos repositórios, assim, caso a versão da aplicação estiver desatualizada no controlador o utilitário entenderá que a aplicação mais atual deverá ser instalada. A seguir um exemplo de utilização:

```
$ spm update ctr2 hub
```

4.5.3 spm remove

Com este comando o administrador da SDN irá desinstalar uma aplicação existente em um controlador especificado, para isto, ele deverá passar junto a este comando o nome do controlador e a aplicação que ele deseja retirar do respectivo *control path*, como mostrado no exemplo a seguir:

```
$ spm remove ctr2 hub
```

4.5.4 spm download

Este comando irá baixar baixar uma lista contendo dados sobre os pacotes hospedados no servidor Web de aplicações SDN. A seguir temos um exemplo de utilização:

```
$ spm download
```

4.5.5 spm repolist

Com este comando é possível obter a listagem dos dados daqueles pacotes de aplicações hospedados no repositório Web e tomar uma decisão, de acordo com a necessidade que tiver, decidindo entre as opções spm install ou *spm update*. A seguir um exemplo de utilização:

```
$ spm repolist
```

4.5.6 spm list

Este comando deve estar acompanhado do nome de um controlador previamente registrado no arquivo *spm.conf* e tem por finalidade retornar ao usuário a lista de aplicações

disponíveis em um controlador especificado na busca. A seguir um exemplo de sintax para o referido comando:

```
$ spm list ctr2
```

4.5.7 spm help

Este comando retorna ao usuário breves informações a cerca dos comandos que dão solução à esta proposta, bem como suas funcionalidades e sintax, assim, o administrador da SDN poderá escolher o melhor comando dependendo do que a situação exigir. A seguir um exemplo:

```
$ spm --help
```

4.6 Desafios de Implementação

Para o desenvolvimento do SPM foi necessário conhecer o funcionamento dos elementos que compõem um *controlpath* SDN, neste contexto foi possível identificar a necessidade por uma ferramenta que otimize o plano de controle por intermédio da gestão dos controladores e seus módulos de aplicações.

Com esta análise houve a necessidade de especificar a modelagem dos pacotes de aplicações uma vez que sem esta padronização seria inviável torná-los compatíveis com uma ferramenta gerenciadora. Feito isto houve o estabelecimento de um servidor Web para estes pacotes, com a missão de impulsionar o desenvolvimento de aplicações e centralizar a sua distribuição.

Uma vez justificada a criação da ferramenta SPM foi definida a sua arquitetura bem como os componentes que compõem esta solução.

4.7 Conclusões do capítulo

A ferramenta SPM disponibiliza funções que satisfazem a idealização desta proposta, de modo que suas APIs auxiliam ao administrador SDN no gerenciamento de seus controladores e com isto escondem a complexidade na manutenção do plano de controle SDN, automatizando de forma simplificada a obtenção de novos pacotes de aplicações para seus respectivos controladores. Em resumo, as principais funcionalidades são:

- Define um modelo de empacotamento para aplicações SDN;
- Define arquivos de configuração para que a ferramenta possa gerenciar aplicações;

- Define um servidor Web para hospedar e distribuir tais pacotes;
- Realiza busca por uma aplicação no servidor;
- Estabelece um repositório local, o mantendo atualizado;
- Instala aplicação no controlador;
- Atualiza aplicação no controlador;
- Remove aplicação do controlador;
- Exibe aplicações disponíveis.

CAPÍTULO 5

Aplicabilidade da Ferramenta SPM

5.1 Introdução

Os principais objetivos traçados para o desenvolvimento desta proposta foram atingidos com pequenos ajustes em relação ao que foi inicialmente proposto, isto porque o gerenciamento dos pacotes de aplicações SDN exigiu novos elementos indispensáveis para a solução.

Neste Capítulo será apresentado a contribuição desta proposta para o paradigma SDN. Em seguida apresenta-se o ambiente dos componentes da ferramenta SPM. Posteriormente será descrito o cenário e metodologia de desenvolvimento utilizada, relacionando-os aos resultados alcançados após a implementação da proposta e por fim será feita uma análise do que fora apresentado.

5.2 Cenário de Atuação

Após definida a modelagem para os pacotes de aplicações foi construído o repositório de referência para que a ferramenta SPM possa acessá-lo e assim selecionar dentre os pacotes ali hospedados, resultando no gerenciamento automatizado das aplicações e dos seus controladores.

O componente responsável por instalar a ferramenta SPM é fundamental para explicar os resultados alcançados após a sua implementação. O instalador está descrito em 4.4.3.2, sendo este o responsável por preparar o Linux para que a ferramenta SPM possa gerenciar as aplicações SDN. O instalador faz com que o principal componente, onde estão implementadas as funcionalidades da ferramenta SPM, seja instalado no diretório de programas do usuário no Linux.

Após a conclusão do projeto, foi elaborado um manual de uso da SPM onde estão descritos ao usuário os requisitos necessários, bem como todos os passos a serem vencidos para que a ferramenta possa ser executada. Além disso, a ferramenta SPM[1] foi disponibilizada em um repositório, onde encontramos o seu código fonte escrito em linguagem Python, juntamente com os demais arquivos e o seu manual de uso em formato PDF.

Para melhor visualização dos resultados implementados neste trabalho, é possível visualizar na Figura 14 uma árvore de diretórios que ilustra a disposição de cada um dos componentes da solução proposta pela ferramenta SPM, que justificam-se pela obervância das definições encontradas em demais sistemas compatíveis com o GNU/Linux.

O uso do caminho /usr/bin/spm.py, deu-se pois este é utilizado por padrão para guardar programas do usuário do sistema. Por sua vez, o caminho /var/lib/ é normalmente usado para guardar coleções de dados, tal como faz o repositório local desta proposta que guarda os pacotes de aplicações disponíveis na SDN e encontra-se no referido caminho por meio do diretório denominado *appmanager*. Já o caminho /etc foi escolhido pois nele se encontram as configurações utilizadas pelos utilitários instalados por um usuário Linux, isto determinou a localização do arquivo spm.conf e reposdn.list, os quais guardam, respectivamente, as principais configurações da SDN e do repositório Web.

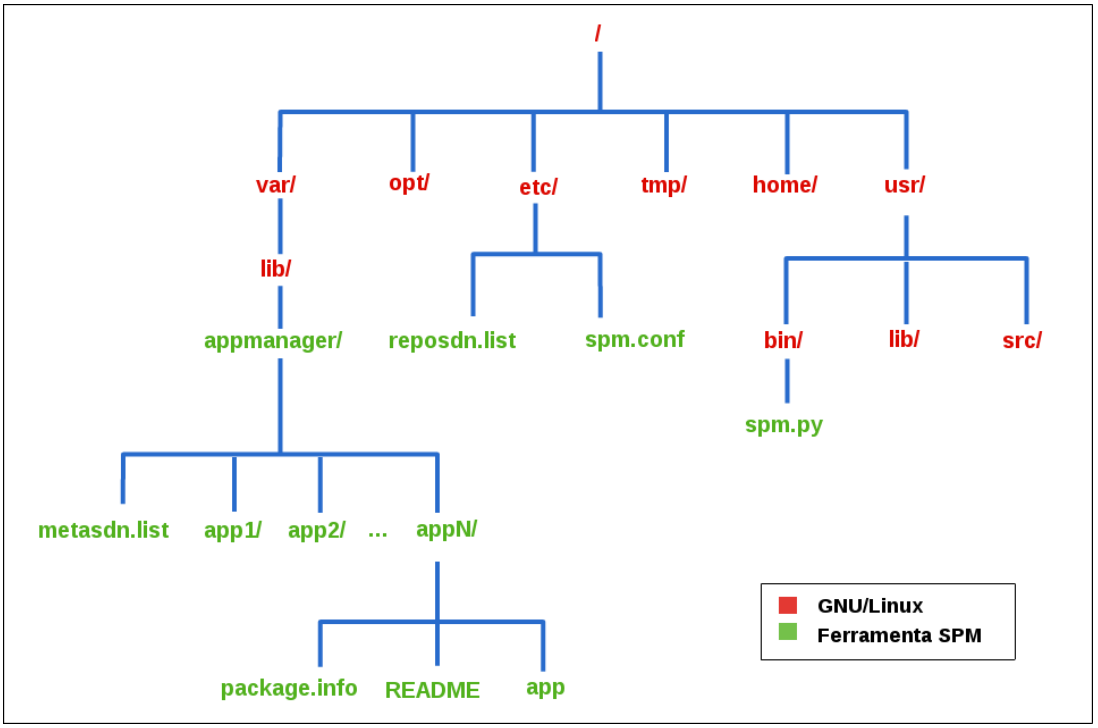

Figura 14: Componentes da Solução [9]

[1] www.github.com/gercom/spm

5.3 Utilização da Ferramenta

Nesta Seção e em suas demais Subseções serão demonstrados os resultados alcançados, onde apresentamos a instalação, a configuração e o gerenciamento de aplicações utilizando a metodologia proposta para a ferramenta SPM.

5.3.1 Instalação da Ferramenta SPM

Para que a Ferramenta SPM seja instalada são necessários alguns arquivos, que são: installspm.sh, metasdn.list, reposdn.list, spm.conf e spm, os arquivos necessários estão disponíveis em: github.com/gercom/SPM.

Todos estes arquivos devem ser baixados para o mesmo diretório a fim de que o instalador da Ferramenta possa encontrá-los durante a instalação. Para instalar a Ferramenta SPM, deve-se ir até o diretório onde se encontram os arquivos necessários e invocar via linha de comando o instalador, certficando-se de utilizar dos privilégios de super usuário, da seguinte forma:

```
# bash installspm.sh
```

Após a execução do instalador descrito em 4.4.3.2, observa-se a mensagem de que foi criado o repositório local da SDN no caminho: /var/lib/appmanager.

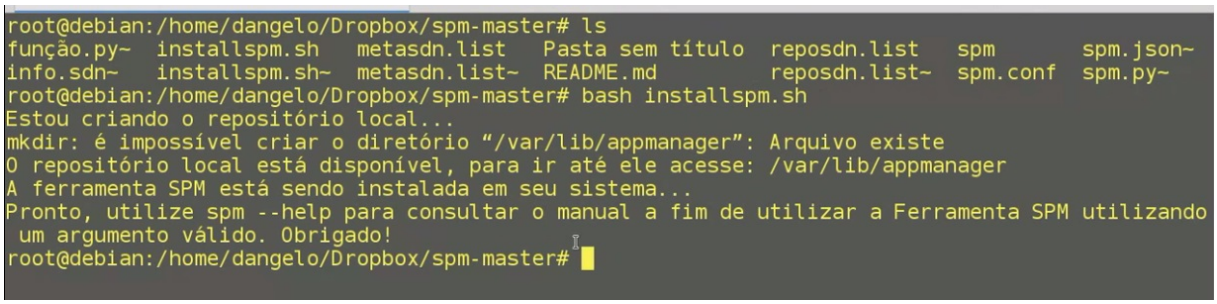

Figura 15: Instalação da Ferramenta SPM

5.3.2 Executando o Primeiro Comando

Inicialmente utilizamos o comando spm –help, ele retorna os demais comandos disponíveis para que a SPM possa gerenciar as aplicações na SDN. Ao lado de cada comando vê-se um breve comentário sobre seu funcionamento.

```
root@debian:/home/dangelo/Dropbox/spm-master# spm --help
Usage: spm [OPTIONS] COMMAND [ARGS]...

Options:
  --help  Show this message and exit.

Commands:
  download  Download the packages list which hosted on the Web repository
  install   Use to install an application on a specific controller
  list      Use to list applications on a specified controller
  remove    This command removes applications from a specified controller
  repolist  This command lists to you all the applications available on the
            web repository
  update    Use this command to update an installed application at specific
            controller
root@debian:/home/dangelo/Dropbox/spm-master#
```

Figura 16: Comando help

5.3.3 Abastecendo o Repositório Web

O repositório de aplicações está hospedado no domínio:

http://reposdn.inovarci.com.br/

O desenvolvedor tem a sua disposição uma interface *back-end*, onde é possível acessá-la para o envio de novos pacotes de aplicações ao servidor Web, devendo apenas observar a padronização definida em 4.4.1, adicionando o nome do pacote a ser submetido e escolhendo a sua categoria de acordo com o controlador.

5.3.3.1 Menu para Envio de Aplicações

Esta tela mostra que após efetuar seu login, o desenvolvedor terá que preencher algumas informações como o nome do pacote e a versão da aplicação, depois acessar outra tela e selecionar uma categoria para o pacote, feito isto, poderá salvar um novo pacote de aplicação a ser disponibilizado no servidor.

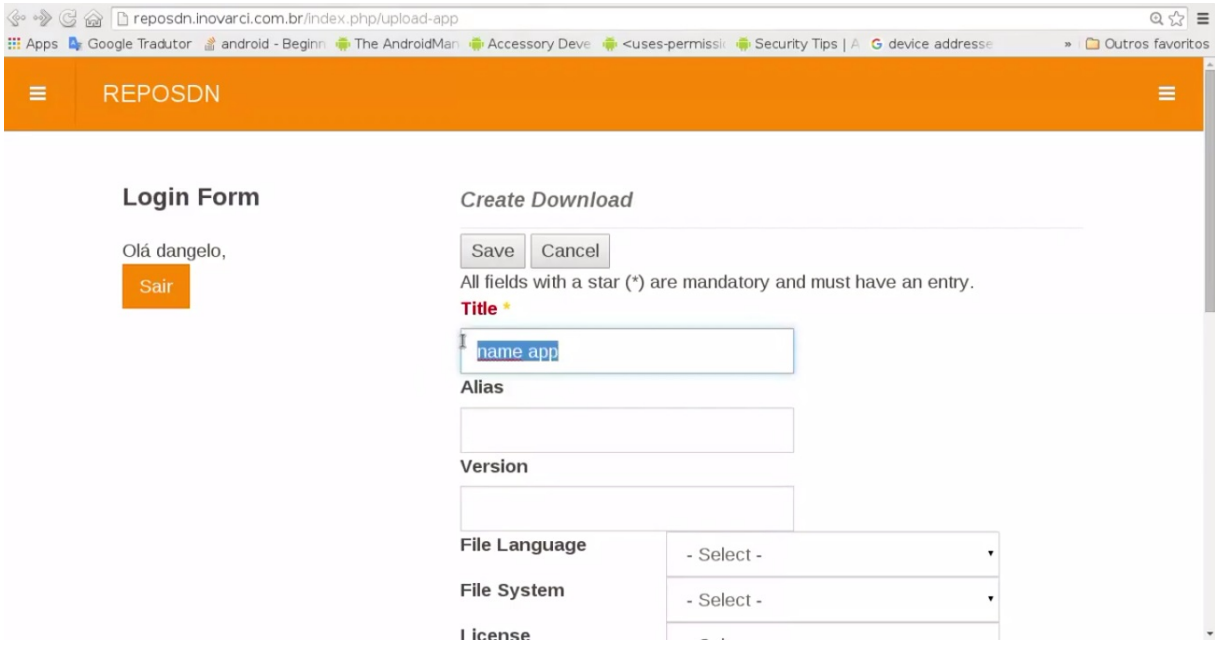

Figura 17: Back-End para Envio de Aplicações ao Servidor Web

5.3.3.2 Selecionando a Categoria da Aplicação

O servidor Web tem disponível três categorias em que se enquadram os pacotes ZIP contendo aplicações, estas representam três diferentes tipos de controladores que são: Nox, Floodlight e POX. Assim, uma vez que se tenha selecionado o arquivo zipado e definido um nome e sua versão, o desenvolvedor terá que selecionar a categoria do pacote, como mostrado a seguir:

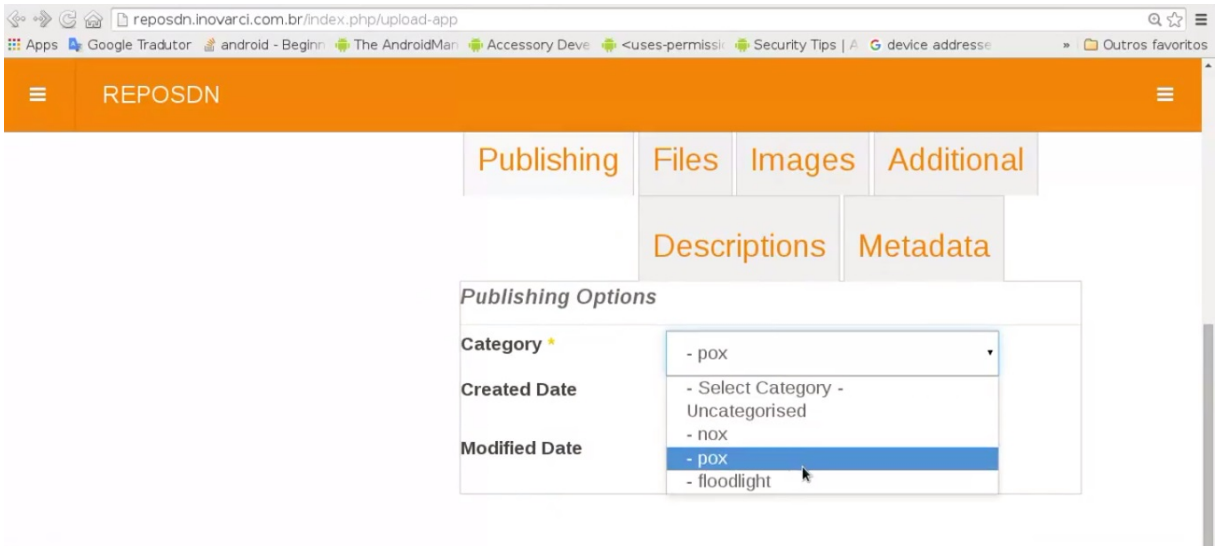

Figura 18: Menu para Seleção de Categoria das Aplicações Enviadas ao Servidor Web

5.3.3.3 Acessando o Servidor na Web

O repositório Web pode hospedar pacotes de aplicações para os controladores mencionados anteriormente. Na Figura a seguir, temos um servidor Apache acessando o repositório do controlador POX, onde vê-se alguns pacotes ZIP os quais seguem a nomenclatura definida nesta proposta, contendo o nome da aplicação, a versão e o controlador a que se destina a aplicação, seguido da extensão ZIP.

Figura 19: Servidor de Aplicações

5.3.3.4 Conteúdo Hospedado

Foi feito o donwload de um pacote ZIP a partir do servidor Web. O pacote baixado segue a definição encontrada na Secão 4.4.2, onde em seu conteúdo encontram-se três arquivos que são: o arquivo README, a aplicação e o arquivo de metadados do pacote, como pode ser visto na Figura a seguir.

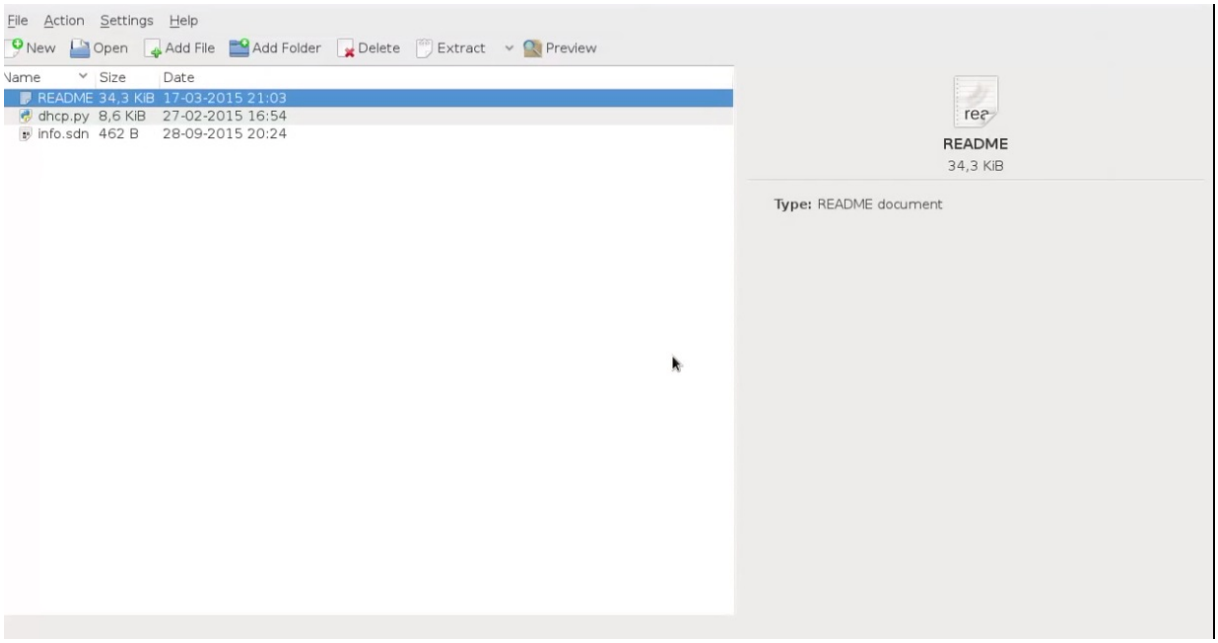

Figura 20: Conteúdo dos Pacotes Hospedados no Servidor Web

5.3.4 Baixando a Lista de Aplicações Disponíveis na Web

Com a linha de comando aberta utilizamos o comando donwload, ele irá acessar o repositório Web e baixar a lista chamada de reposdn.list, esta lista estará sempre atualizada com as novas aplicações disponíveis no repositório.

Para visualizar quais aplicações estão disponíveis, utilizamos o comando spm repolist, ele retorna as aplicações do repositório Web as apresentando por tipo de controlador, como pode ser visto na Figura 21.

```
root@debian:/home/dangelo/Dropbox/spm-master# spm download
downloading Web repository list...
root@debian:/home/dangelo/Dropbox/spm-master# spm repolist
Controller: pox
Applications: ['dhcp-1.0', 'hub-1.0', 'hub-2.0', 'l2_learning-1.0', 'skeleton-1.0', 'skeleton-2.
0']
Controller: floodlight
Applications: ['firewall-1.0', 'forwarding-1.0', 'hub-1.0']
Controller: nox
Applications: []
root@debian:/home/dangelo/Dropbox/spm-master#
```

Figura 21: Comandos download e repolist

5.3.5 Configurando a SDN

Na Figura 14 pode-se ver detalhamente onde são encontrados os componentes da Ferramenta SPM. No diretório /etc/ encontra-se o arquivo spm.conf que é estruturado de modo a instanciar os campos:

- **repository:** onde recebe o endereço para o repositório Web, local onde a farramenta

SPM irá buscar por aplicações;

- **cache:** neste campo será indicado o caminho para o repositório local da SDN, onde deverão ser baixadas as aplicações;
- **controller:** onde deverá ser editado e nele adicionado um nome aleatório dado a cada controlador disponível na SDN;

No campo controller há outros três campos, onde devem existir as seguites informações:

- **path:** informar o caminho até ao controlador;
- **type:** o tipo de cada controlador SDN: NOX, Floodlight ou POX; e
- **apps:** este campo é mantido atualizado pela própria ferramenta SPM, uma vez que a partir deste campo serão gerenciadas as aplicações de cada controlador.

A seguir vemos que antes do inicio da realização dos testes, estavam a disposição os controladores: ctr1, ctr2, ctr3 e ctrN. Em cada um dos controladores vemos seus respectivos caminhos e tipos e nenhuma aplicação instalada.

```
{
    "repository": "http://reposdn.inovarci.com.br/jdownloads",
    "cache": "/var/lib/appmanager/",
    "controller": {
        "ctr1": {
            "path": "/home/dangelo/floodlight-0.91/",
            "type": "floodlight",
            "apps": null
        },
        "ctr2": {
            "path": "/home/dangelo/pox/",
            "type": "pox",
            "apps": null
        },
        "ctr3": {
            "path": "/home/dangelo/pox2/",
            "type": "pox",
            "apps": null
        },
        "ctrN": {
            "path": "/home/dangelo/ryu/",
            "type": "ryu",
            "apps": null
        }
    }
}
```

Figura 22: Arquivo spm.conf

5.3.6 Instalando Aplicações com a Ferramenta SPM

Após consultar as aplicações disponíveis no servidor Web por meio do comando spm repolist, os testes transcorreram com a instalação de duas novas aplicações no controlador ctr2, o qual foi apresentado na Figura 22, para isso utilizou-se o comando spm install, como pode ser visto na Figura a seguir.

Inicialmente instalou-se a versão 1.0 da aplicação hub no controlador ctr2, onde ao final da execução observou-se uma mensagem de "Instalação Completa", em seguida foi instalada, também em ctr2, a versão 1.0 da aplicação skeleton, que igualmente retornou a mensagem de "Instalação Completa".

Adicionalmente é possível ver a sequência de execuções do comando spm install, onde este inicialmente realiza o download do pacote de aplicação, tendo como destino o repositório local que foi definido nesta proposta em /var/lib/appmanager, como também pode ser visto na Figura 14.

```
root@debian:/home/dangelo/Dropbox/spm-master# spm install ctr2 hub
http://reposdn.inovarci.com.br/jdownloads/pox/hub-1.0-pox.zip /var/lib/managerapp/hub-1.0-pox.zi
p
hub-1.0-pox
/var/lib/managerapp/hub-1.0-pox.zip
Archive:  /var/lib/managerapp/hub-1.0-pox.zip
  inflating: /tmp/hub-1.0-pox/README
  inflating: /tmp/hub-1.0-pox/hub.py
  inflating: /tmp/hub-1.0-pox/info.sdn
Instalação Completa!!!
root@debian:/home/dangelo/Dropbox/spm-master# spm install ctr2 skeleton
http://reposdn.inovarci.com.br/jdownloads/pox/skeleton-1.0-pox.zip /var/lib/managerapp/skeleton-
1.0-pox.zip
skeleton-1.0-pox
/var/lib/managerapp/skeleton-1.0-pox.zip
Archive:  /var/lib/managerapp/skeleton-1.0-pox.zip
  inflating: /tmp/skeleton-1.0-pox/info.sdn
  inflating: /tmp/skeleton-1.0-pox/README
  inflating: /tmp/skeleton-1.0-pox/skeleton.py
Instalação Completa!!!
root@debian:/home/dangelo/Dropbox/spm-master#
```

Figura 23: Comandos spm install

5.3.7 Destino das Aplicações no POX

Os testes utilizaram um dos controladores Pox disponíveis ao qual foi dado o nome de ctr2. O caminho até ctr2 pode ser visto na Figura 22. Utilizando a linha de comando, acessamos o caminho e entramos em pox/ext/, sendo este o local onde o POX armazena por *default* as suas aplicações. O diretório ext/ não está presente no arquivo spm.conf, pois foi definido que incumbe a ferramenta SPM fazer o tratamento do mesmo.

Neste momento do primeiro acesso ao caminho onde está instalado o controlador ctr2, podemos ver que inicialmente o mesmo não dispunha de aplicações em seu *control path*, notando-se apenas a presença de um arquivo README. Após a realização de duas instalações que podem ser vistas na Subseção 5.3.6, realizamos novamente o acesso ao caminho de ctr2 e constatamos que as aplicações haviam sido corretamente instaladas com o auxilio da ferrameta SPM.

```
root@debian:/home/dangelo/pox/ext# pwd
/home/dangelo/pox/ext
root@debian:/home/dangelo/pox/ext# ls
README
root@debian:/home/dangelo/pox/ext# ls
hub.py  README  skeleton.py
root@debian:/home/dangelo/pox/ext#
```

Figura 24: Diretório para Instalação de Aplicações no POX

5.3.8 Removendo Aplicações com a Ferramenta SPM

Para testar o comando spm remove utilizamos os mesmos argumentos de spm install, ou seja, nesta Subseção demonstramos a remoção das aplicações anteriormente instaladas na Subseção 5.3.6, onde foi utilizado o controlador POX chamado de ctr2.

Primeiramente foi removida a aplicação hub, depois a aplicação skeleton, o que, em ambos os casos, retornou a mensagem "Dropped databse", isto significa que além de retirar as aplicações solicitadas no referido controlador, também atualizou-se o arquivo spm.conf, de onde foram sacadas as aplicações que lhe constavam.

```
root@debian:/home/dangelo/Dropbox/spm-master# spm remove ctr2 hub
Dropped the database
root@debian:/home/dangelo/Dropbox/spm-master# spm remove ctr2 skeleton
Dropped the database
root@debian:/home/dangelo/Dropbox/spm-master#
```

Figura 25: Comandos spm remove

5.3.9 Atualizando Aplicações com a Ferramenta SPM

Para demonstração do comando spm update, realizamos novamente as instalações vistas na Subseção 5.3.6, retornando ao controlador ctr2 com as aplicações hub e skeleton em suas versões 1.0.

Ao usar-se novamente o comando spm repolist, identificou-se que há, no servidor Web, aplicações com versões superiores se comparadas àquelas aplicações previamente instaladas em ctr2. Então para atualizá-las, utiliza-se o comando spm update, o qual recebe os mesmos argumentos dos comandos spm install e spm remove.

Assim, a ferramenta SPM fará uma comparação entre as versões das aplicações instaladas na SDN e as aplicações correspondentes disponíveis no repositório Web. Uma vez que sejam identificadas versões superiores, então a ferramenta SPM irá, respectivamente, remover a versão defasada e instalar as versões 2.0 da aplicação hub e skeleton.

```
root@debian:/home/dangelo/Dropbox/spm-master# spm update ctr2 hub
There is a new version available at SDN Repository...
Dropping old version...
Installing new version...
http://reposdn.inovarci.com.br/jdownloads/pox/hub-2.0-pox.zip /var/lib/managerapp/hub-2.0.zip
hub-2.0-pox
/var/lib/managerapp/hub-2.0-pox.zip
Archive:  /var/lib/managerapp/hub-2.0-pox.zip
  inflating: /tmp/hub-2.0-pox/hub.py
  inflating: /tmp/hub-2.0-pox/info.sdn
  inflating: /tmp/hub-2.0-pox/README
root@debian:/home/dangelo/Dropbox/spm-master# spm update ctr2 skeleton
There is a new version available at SDN Repository...
Dropping old version...
Installing new version...
http://reposdn.inovarci.com.br/jdownloads/pox/skeleton-2.0-pox.zip /var/lib/managerapp/skeleton-
2.0.zip
skeleton-2.0-pox
/var/lib/managerapp/skeleton-2.0-pox.zip
Archive:  /var/lib/managerapp/skeleton-2.0-pox.zip
  inflating: /tmp/skeleton-2.0-pox/info.sdn
  inflating: /tmp/skeleton-2.0-pox/README
  inflating: /tmp/skeleton-2.0-pox/skeleton.py
root@debian:/home/dangelo/Dropbox/spm-master#
```

Figura 26: Comandos spm update

5.3.10 Ferramenta SPM Listando Aplicações de um Controlador

Para listar aplicações de um determinado controlador é possível utilizar o comandos spm list que aceita como argumento um dos controladores da SDN, neste caso, o controlador ctr2. Este comando retorna as aplicações instaladas em ctr2 e suas respectivas versões, como pode ser visto a seguir.

```
root@debian:/home/dangelo/Dropbox/spm-master# spm list ctr2
{'hub': '2.0', 'skeleton': '2.0'}
root@debian:/home/dangelo/Dropbox/spm-master#
```

Figura 27: Comandos spm list

5.3.11 Configurações Atualizadas Após o Uso da Ferramenta SPM

Todos os componentes da ferramenta SPM são atualizados de modo a garantir o gerenciamento automatizado das aplicações de uma SDN, como apresentado nas Subseções a seguir.

5.3.11.1 Acessando o Arquivo spm.conf Atualizado

A seguir, na Figura 28, pode-se visualizar que o arquivo spm.conf foi atualizado com as novas aplicações instaladas nos testes de validação, o acesso ao mesmo deu-se após os testes descritos, onde buscou-se pelo caminho /etc/spm.conf, a fim de exibir-se o seu conteúdo.

Figura 28: Arquivo spm.conf Após a Execução da Ferramenta SPM

5.3.11.2 Acessando o Arquivo metasdn.list Atualizado

O arquivo metasdn.list é atualizado sempre que a ferramenta SPM gerencia uma nova aplicação de um controlador da SDN. A seguir na Figura 29, observa-se os metadados das aplicações instaladas e que posteriormente foram atualizadas para versões superiores, de acordo com os testes de validação descritos anteriormente. Estes metadados são extraídos do arquivo package.info, que pode ser encontrado em qualquer pacote que siga a definição apresentada nesta proposta.

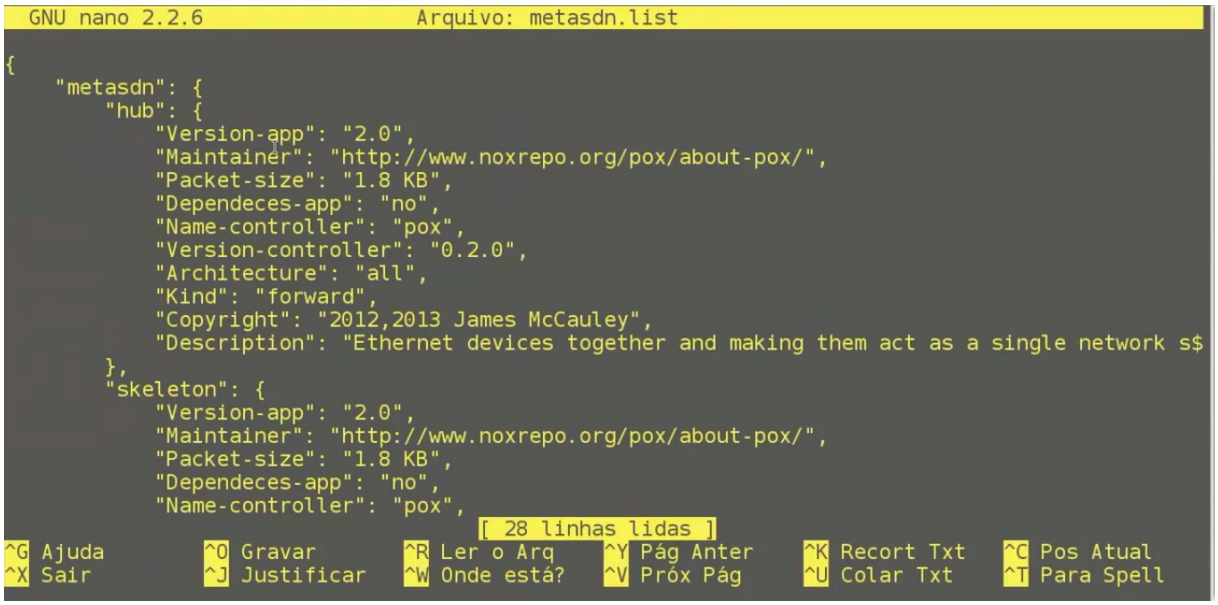

Figura 29: Arquivo metasdn.list Após a Execução da Ferramenta SPM

5.4 Contribuição Científica

Com a proposta apresentada neste trabalho de conclusão de curso, obteve-se o aceite do trabalho intitulado "Software Package Manager for SDN Applications - SPM" [9], após submissão de um artigo para um evento internacional com Qualis B4, onde foi publicado um *Application Paper* no *VIII Latin American Network Operations and Management Symposium (LANOMS 2015)*[2], este evento ocorreu sob supervisão da UFPB nos dias 01-03 de outubro de 2015, em João Pessoa - PB.

5.5 Conclusões do Capítulo

Com os resultados descritos neste Capítulo, demonstrou-se a implementação de uma ferramenta que propõe a solução para os problemas propostos inicialmente, além disso, ao criar-se um gerenciamento automatizado de aplicações e controladores SDN, permitiu-se uma publicação em um evento internacional que assegura a contribuição científica a cerca do avanço para o paradigma SDN.

[2]http://www.lanoms.org/2015/

CAPÍTULO 6

Conclusões e Trabalhos Futuros

6.1 Considerações Finais

O objetivo deste trabalho foi propor uma padronização aos pacotes de software contendo aplicações SDN, para assim gerenciá-las a partir do plano de controle de redes programáveis, com o uso da ferramenta SPM.

Com o estudo a cerca do controlador POX, permitiu-se o entendimento necessário para definição do conteúdo de um pacote que, uma vez padronizado, tornou possível aplicar-lhes uma metodologia de distribuição centralizada na Web com a adoção de práticas de gerenciamento de pacotes semelhante a de ferramentas já consagradas no GNU/Linux.

A compatibilidade entre a ferramenta SPM e o repositório Web, permite agora ao administrador da SDN selecionar, dentre os comandos SPM, aquele melhor adaptado a sua necessidade, a fim de obter, via CLI, o gerenciamento de aplicações em seus respectivos controladores de forma simplificada e transparente, concluindo satisfatoriamente o que fora proposto neste trabalho de conclusão de curso.

6.2 Trabalhos Futuros

Como trabalhos futuros pretende-se desenvolver uma interface gráfica, criando um Web Service, com o qual seja possível integrar as funcionalidades encontradas na solução proposta pela ferramenta SPM, assim, com o uso de um navegador Web, teremos uma visão ampla da rede.

O gerente da SDN poderá melhor organizar sua rede com o uso do sistema gráfico

da ferramenta SPM, que permitirá, dentre outras coisas: apresentar tabelas, estatísticas e imprimir relatórios a cerca do uso de aplicações em todos os controladores.

A partir do mapeamento desses dados permite-se a tomada de decisões e o direcionamento de metas onde se possa aumentar o desempenho da rede, excluindo componentes ociosos na SDN ou aumentando o número de controladores e aplicações de acordo com a identificação de uma demanda.

Referências

[1] H. Networking, "OpenFlow," http://pro-networking-h17007.external.hp.com/br/pt/solutions/technology/openflow/index.aspx, 2012, [Online; acessado em 11 de outubro de 2014].

[2] cpqd, "OpenFlow e redes definidas por software: Um novo paradigma de controle e inovação em redes de pacotes," http://www.cpqd.com.br/cadernosdetecnologia/Vol7_N1_jul2010_jun2011/pdf/artigo6.pdf, 2011, [Online; acessado em 9 de janeiro de 2015].

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

[4] R. Pinheiro, B. Pinheiro, R. Esteves, and A. Abelem, "Reposdn: An repository organization and coordination method of software defined networks applications," in *Network Operations and Management Symposium (NOMS), 2014 IEEE*, May 2014, pp. 1–4.

[5] LinuxFocus, "Package management made easy," http://new.linuxfocus.org/cms/?q=node/7, [Online; acessado em 29 de outubro de 2015].

[6] L. Format, "The lfx test," http://www.linuxformat.com/content/lxf-test-hands-fedora-10?op=modload, [Online; acessado em 13 de outubro de 2015].

[7] S. Linux, "slackware docs," http://docs.slackware.com/slackware:faq, [Online; acessado em 9 de outubro de 2014].

[8] OpenSuse, "Gerenciamento de Software do YaST," https://pt.opensuse.org/Gerenciamento_de_Software_do_YaST, [Online; acessado em 9 de outubro de 2014].

[9] D. Mendes, B. Pinheiro, E. Cerqueira, and A. Abelém, "Software package manager for sdn applications - spm," in *LANOMS 2015 - Application Session ()*, oct 2015, pp. 145–155.

[10] O. N. Lab, "POX wiki," https://openflow.stanford.edu/display/ONL/POX+Wiki, [Online; acessado em 19 de novembro de 2014].

[11] N. Chowdhury and R. Boutaba, "Network virtualization: state of the art and research challenges," *Communications Magazine, IEEE*, vol. 47, no. 7, pp. 20–26, July 2009.

[12] M. Jarschel, S. Oechsner, D. Schlosser, R. Pries, S. Goll, and P. Tran-Gia, "Modeling and performance evaluation of an openflow architecture," in *Teletraffic Congress (ITC), 2011 23rd International*, Sept 2011, pp. 1–7.

[13] D. Sampaio, "A Internet mudou a nossa vida," http://www.publico.pt/opiniao/noticia/, 2014, [Online; acessado em 15 de setembro de 2015].

[14] Computerworld, "Esgota o estoque de IPv4 na América Latina," http://computerworld.com.br/seguranca/2014/06/10/esgota-o-estoque-de-ipv4-na-america-latina, 2014, [Online; acessado em 23 de fevereiro de 2015].

[15] C. Huang, J. Zhu, M. Luo, and W. Chou, "A new mechanism for sdn network virtualization service," in *Smart Communications in Network Technologies (SaCoNeT), 2014 International Conference on*, June 2014, pp. 1–6.

[16] Princeton, "Future Internet Architecture: Clean-Slate Versus Evolutionary Research," https://www.cs.princeton.edu/~jrex/papers/cacm10.pdf, 2014, [Online; acessado em 27 de janeiro de 2015].

[17] ONF, "Software-Defined Networking: The New Norm for Networks," https://www.opennetworking.org/images/stories/downloads/sdn-resources/white-papers/wp-sdn-newnorm.pdf, 2012, [Online; acessado em 8 de junho de 2014].

[18] SDN, "Baixar Citations," http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=arnumber=5984813isnumber=5984804, [Online; acessado em 12 de junho de 2014].

[19] H. Hata, "A study of requirements for sdn switch platform," in *Intelligent Signal Processing and Communications Systems (ISPACS), 2013 International Symposium on*, Nov 2013, pp. 79–84.

[20] ONF, "OpenFlow: Enabling Innovation in Campus Networks," http://archive.openflow.org/documents/openflow-wp-latest.pdf, 2008, [Online; acessado em 9 de janeiro de 2015].

[21] ONF.org, "OpenFlow Switch Errata," https://www.opennetworking.org/images/stories/downloads/sdn-resources/onf-specifications/openflow/openflow-spec-v1.0.2.pdf, 2013, [Online; acessado em 20 de janeiro de 2015].

[22] Cisco, "Software-Defined Networks and OpenFlow," http://www.cisco.com/web/about/ac123/ac147/archived_issues/ipj_16-1/161_sdn.html, 2013, [Online; acessado em 16 de janeiro de 2015].

[23] ONF, "OpenFlow Switch Specification," https://www.opennetworking.org/images/stories/downloads/sdn-resources/onf-specifications/openflow/openflow-spec-v1.4.0.pdf, 2013, [Online; acessado em 16 de janeiro de 2015].

[24] Nox, "About POX," http://www.noxrepo.org/pox/about-pox/, [Online; acessado em 24 de julho de 2014].

[25] Nicyra, "POX," www.noxrepo.org/pox/about-pox/, 2013, [Online; acessado em 29 de outubro de 2015].

[26] GitHub, "The POX Controller," https://github.com/noxrepo/pox, 2013, [Online; acessado em 24 de julho de 2014].

[27] R. Pinheiro, B. Pinheiro, and A. Gomes Abelem, "Model of organization and distribution of applications for software defined networks: Sdnrepo," in *Communications and Electronics (ICCE), 2014 IEEE Fifth International Conference on*, July 2014, pp. 188–193.

[28] Distrowatch.com, "Ranking de Visitas," http://distrowatch.com/, [Online; acessado em 20 de agosto de 2014].

[29] die.net, "rpm(8) - Linux man page," http://linux.die.net/man/8/rpm, [Online; acessado em 23 de setembro de 2014].

[30] Debian, "Debian Policy Manual - Chapter 3 - Binary Packages," https://www.debian.org/doc/debian-policy/ch-binary.html, [Online; acessado em 15 de julho de 2014].

[31] debian.org, "Como usar o APT," https://www.debian.org/doc/manuals/apt-howto/index.pt-br.html, [Online; acessado em 15 de julho de 2014].

[32] Debian.org, "Manual do Administrador Debian," http://debian-handbook.info/browse/pt-BR/stable/sect.manipulating-packages-with-dpkg.html, [Online; acessado em 15 de julho de 2014].

[33] die.net, "yum(8)- Linux man page," http://linux.die.net/man/8/yum, [Online; acessado em 9 de outubro de 2014].

APÊNDICE A – Código do Intalador da Ferramenta SPM

Aqui nesta Seção, apresenta-se o código, em linguagem shell script, que executa o instalador da ferramenta SPM. Os demais códigos encontram-se disponibilizados no Github, no link descrito na Seção 5.3.1

```
#!/bin/bash

function chk_root()
{
  # Para prosseguir é feita antes essa verificação de usuário root
  if [ $(whoami) != "root" ]; then
    echo "Preciso de privilégios root para continuar!"
    exit 2
  fi
}

# chama a função anteior para verificação de super usuário
chk_root

echo "Estou criando o repositório local..."
# cria o repositório local
mkdir /var/lib/appmanager
echo "O repositório local está disponível, para ir até ele acesse:
 /var/lib/appmanager "

# verificação do arquivo python que implementa a ferramenta SPM
if [ ! -f spm ]; then
  echo "Não pude encontrar o arquivo SPM, coloque o arquivo na pasta
  corrente onde se encontra este instalador!"
exit 1
fi

# preparação do python que implementa a ferramenta SPM, dando
```

```
permissão ao mesmo
if [ ! -x spm ]; then
  chmod a+x spm
fi

# mensagem de instalação da ferramenta SPM
echo "A ferramenta SPM está sendo instalada em seu sistema..."

cp metasdn.list /var/lib/appmanager/
cp spm.conf /etc/
cp spm /usr/bin
echo "Pronto, utilize spm --help para consultar o manual a fim
de utilizar a Ferramenta SPM utilizando um argumento válido.
Obrigado!"
```

APÊNDICE B – Lista de Aplicações do Repositório Web

A baixo segue o código do arquivo reposdn.list utilizado durante os testes. Este arquivo é atualizado automaticamente sempre que o comando *spm download* for utilizado após uma nova aplicação ter sido disponibilizada no servidor Web.

```
{
    "nox": [],
    "floodlight": [
        "firewall-1.0",
        "forwarding-1.0",
        "hub-1.0"
    ],
    "pox": [
        "hub-1.0",
        "hub-2.0",
        "l2_learning-1.0",
        "skeleton-1.0",
        "skeleton-2.0"
    ]
}
```