LUCIANO GOYA BILLOTTA

ARTEFATO DE SOFTWARE PARA ALOCAÇÃO DINÂMICA DE RECURSOS

Dissertação apresentada ao Programa de Pós-Graduação em Engenharia Elétrica do Centro Tecnológico da Universidade Federal do Espírito Santo, como requisito parcial para obtenção do Grau de Mestre em Engenharia Elétrica.
Orientador: Prof. Dr. Anilton Salles Garcia.

VITÓRIA
2011

Dados Internacionais de Catalogação-na-publicação (CIP)
(Biblioteca Central da Universidade Federal do Espírito Santo, ES, Brasil)

Billotta, Luciano G. 1982-
B599a Artefato de software para alocação dinâmica de recursos / Luciano Goya Billotta. - 2011.
122 f.: il.

Orientador: Anilton Salles Garcia.

Dissertação (Mestrado em Engenharia Elétrica) - Universidade Federal do Espírito Santo, Centro Tecnológico.

1. Telecomunicações - Sistemas de comutação. 2. Rede de Computador - Protocolos. 3. Redes de computadores. I. Garcia, Anilton Salles. II. Universidade Federal do Espírito Santo. Centro Tecnológico. III. Título.

CDU: 621.3

LUCIANO GOYA BILLOTTA

ARTEFATO DE SOFTWARE PARA ALOCAÇÃO DINÂMICA DE RECURSOS

Dissertação submetida ao programa de Pós-Graduação em Engenharia Elétrica do Centro Tecnológico da Universidade Federal do Espírito Santo, como requisito parcial para a obtenção do Grau de Mestre em Engenharia Elétrica.

Aprovada em 11 de novembro de 2011.

COMISSÃO EXAMINADORA

Prof. Dr. Anilton Salles Garcia
Universidade Federal do Espírito Santo
Orientador

Prof. Dra. Rosane Bodart Soares
Universidade Federal do Espírito Santo

Prof. Dr. Carlos Alberto Malcher Bastos
Universidade Federal Fluminense

Dedico esse trabalho aos meus pais e irmã...

Agradecimentos

Agradeço primeiramente a Deus por ter me dado inspiração e força nos momentos em que mais precisei, aos meus pais por sempre me apoiarem nas decisões que tomo em minha vida, ao meu orientador por me proporcionar um aprendizado em que foi possível colocar a teoria em prática e a todos os envolvidos do Projeto Futura RNP que estiveram à disposição nos momentos de dúvidas e que muito me ensinaram.

Resumo

Este trabalho propõe um artefato de software para alocação dinâmica de recursos, baseado nas pesquisas realizadas no Projeto TIAMHAT, desenvolvido em cooperação com a Rede Nacional de Ensino e Pesquisa.

Esse tipo de software é de alta importância para as redes de pesquisa ao redor do mundo dado que a comunidade científica necessita de comunicação através de redes com capacidade para suportar alto tráfego de dados também conhecidas como *e-science*.

O artefato proposto é baseado em duas soluções estudadas no Projeto e, por isso, este trabalho também se propõe a descrever essas tecnologias, assim como as arquiteturas e protocolos envolvidos na alocação com o intuito de se chegar a um maior entendimento de como a nova proposta foi implementada.

Espera-se que o artefato proposto seja aceito pela comunidade científica e, por se tratar de uma solução aberta, que seja aperfeiçoada ao longo do tempo pelos próprios usuários e incorporada a softwares que já possuem mais funcionalidades para a alocação dinâmica de recursos.

Palavras-chave: Telecomunicações - Sistemas de comutação, Rede de Computador - Protocolos, Redes de computadores.

Abstract

This work proposes an artifact of a software able to dynamically allocate resources, based in researches in TIAMHAT Project.

This kind of software is very important for research networks around the world because of the high demand by scientific community that needs communication through high traffic networks also known as *e-science*.

This new artifact is based in two solutions studied in the Project, that is why this work also describes those technologies with their architectures and protocols involved in allocation with the goal to make it easy to understand how the new proposal was implemented.

It's expected this proposed artifact to be accepted by the scientific community and be improved by users because of the open solution aspect and maybe incorporated in softwares with more functionalities in dynamic allocation resources.

Keywords: Telecommunications - Switching Systems, Computer Network - Protocols, Computers Network.

Lista de Figuras

3.1 Shim Header 7
3.2 Arquitetura Centralizada e Descentralizada 8
3.3 Diagrama de uma rede DRAGON 11
3.4 Diagrama Módulos AutoBAHN 16

4.1 Objeto RSVP-TE 22
4.2 Diagrama MIB 26
4.3 Diagrama de troca de Mensagens GMPLS 29

5.1 Diagrama do Artefato de Sistema 32
5.2 Inserção da adaptação para Cisco 33
5.3 Código específico para Cisco 34
5.4 Imagem da Tela de Login do Artefato 35
5.5 Imagem da Tela Principal do Artefato 36
5.6 Imagem do Formulário de Inserção de Nós 36
5.7 Formulário para criação do LSP 37
5.8 Código do Framework PySNMP 41
5.9 Diagrama dos Testes 43
5.10 Tela Principal do Sistema com LSP Agendado 45
5.11 Tela com Formulário de Criação do Nó 46
5.12 Tela com Informações do Nó Criado 47

5.13 Tela com Formulário de Criação do LSP 48
5.14 Tela com Informações do LSP Ativo 49
5.15 Tela com Informações do LSP Desativado 50
5.16 Tela com Configuração do Switch 51
5.17 Tela com Configuração do Switch 52

6.1 Diagrama com novos módulos 55

A.1 Diagrama de Caso de Uso 61
A.2 Fluxograma de criação de Nós 68
A.3 Fluxograma de criação de Porta Trunk 68
A.4 Fluxograma de criação e agendamento de LSPs 69
A.5 Fluxograma de remoção de Portas Trunk 69
A.6 Fluxograma de Desativação de LSPs 70
A.7 Fluxograma de Desativação de Agendamento 70
A.8 Fluxograma de Remoção de Nó 71
A.9 Fluxograma de remoção de LSPs 71
A.10 Diagrama UML do Artefato 72

B.1 Inserção da adaptação para Cisco 76
B.2 Inserção de nova opção de Fabricante 77

C.1 Imagem da Tela de Login do Artefato 78
C.2 Imagem da Tela Principal do Artefato 79
C.3 Imagem do Formulário de Inserção de Nós 80
C.4 Tela com informações sobre o Nó 80
C.5 Formulário para criação do LSP 81
C.6 Tela do Artefato LSP Agendado 82

Lista de Tabelas

5.1 Tabela comparativa entre as soluções 38

Nomenclatura

AAA Authentication, Authorization and Accounting

AAAS Authentication, Authorization and Auditing Subsystem

API Application Programming Interface

ASTB Application Specific Topology Builder

ATM Asynchronous Transfer Mode

AutoBAHN Automated Bandwidth Allocation across Heterogeneous Networks

BSS Bandwidth Scheduler Subsystem

CLI Command Line Interface

CSA Client System Agent

DCN Dynamic Circuit Networking

DM Domain Manager

DRAGON Dynamic Resource Allocation via GMPLS Optical Networks

ESnet Energy Sciences Network

FEDERICA Federated E-Infrastructure Dedicated to European Researchers Innovating in Computing Network Architectures

GMPLS Generalized Multiprotocol Label Switching

GpENI Great Plains Environment for Network Innovation

HTTP HyperText Transfer Protocol

IDM Inter Domain Manager

IEEE Institute of Electrical and Electronic Engineers

IETF Internet Engineering Task Force

LFIB Label Forwarding Information Base

LMP Link Management Protocol

LSA Link State Advertisement

LSP Label Switched Path

LSR Label Switching Router

MIB Management Information Base

MPLS-TE Multiprotocol Label Switching - Traffic Engineering

MVC Model-View-Controller

NARB Network Aware Resource Broker

NMS Network Management System

NSF National Science Foundation

OID Object Identifier

OPWA One Pass With Advertising

OSCARS On-Demand Secure Circuits and Advance Reservation System

OSPF-TE Open Shortest Path - Traffic Engineering

PDU Protocol Data Unit

PSS Path Setup Subsystem

QoS Quality of Service

RFC Request for Comments

RM Reservation Manager

RNP Rede Nacional de Ensino e Pesquisa

RSVP-TE Resource Reservation Protocol - Traffic Engineering

SNMP Simple Network Protocol

SOAP Simple Object Access Protocol

SQL Structured Query Language

TCP Transmission Control Protocol

TDM Time Division Multiplexing

TED Traffic Engineering Database

TIAMHAT Tecnologias de Aprovisionamento Dinâmico de Conexões para Redes Híbridas

TLV Type-Lenght-Variable

UDP User Datagram Protocol

UML Unified Modeling Language

USM USer Security Model

USN UltraScience Network

VACM View-Based Access Control Model

VCI Virtual Channel Identifier

VLAN Virtual Local Area Network

VLSR Virtual Label Switch Router

VPI Virtual Path Identifier

VPN Virtual Private Network

WBUI Web-Based User Interface

WDM WaveLenght Division Multiplexing

WSDL Web Service Description Language

Sumário

1 Introdução **1**

1.1 Motivação . . . 3

1.2 Justificativa . . . 3

1.3 Objetivos Gerais . . . 4

1.4 Objetivos Específicos . . . 4

1.5 Metodologia . . . 4

1.6 Principais Resultados . . . 5

1.7 Organização do Texto . . . 5

2 Trabalhos Relacionados **1**

2.1 Introdução . . . 1

2.2 Experimentos de Alocação Dinâmica de Recursos . . . 1

2.3 Conclusão . . . 3

3 Arquiteturas de Alocação Dinâmica de Circuitos **5**

3.1 Introdução . . . 5

3.1.1 Projeto TIAMHAT . . . 5

3.2 GMPLS . . . 6

3.2.1 *Shim Header* . . . 6

3.2.2 LSP Tunnel . . . 8

3.2.3 Arquitetura 8
3.3 DRAGON e OSCARS 10
3.3.1 DRAGON 10
3.3.2 OSCARS 12
3.4 DRAGON/OSCARS 13
3.5 AutoBAHN 14
3.5.1 Objetivo 14
3.5.2 Arquitetura 15
3.6 Vantagens e Desvantagens 17

4 Os protocolos **19**
4.1 Introdução 19
4.2 RSVP-TE 19
4.2.1 Mensagens RSVP-TE 20
4.2.2 Objetos RSVP-TE 22
4.3 OSPF-TE 23
4.4 SNMP 24
4.4.1 MIB 25
4.4.2 Segurança 27
4.5 Estabelecimento de LSPs 28
4.5.1 LSP no GMPLS 28
4.5.2 LSP no DRAGON/OSCARS 28
4.5.3 LSP no AutoBAHN 29

5 Artefato de Software **31**
5.1 Introdução 31
5.2 Visão Geral do artefato 31
5.3 Adaptabilidade 33

5.4 Funcionamento do Artefato . . . 34
5.5 Comparativo entre as Soluções . . . 38
5.6 Frameworks . . . 40
5.6.1 Python . . . 40
5.6.2 PySNMP . . . 41
5.6.3 Django . . . 42
5.7 Testes . . . 43
5.7.1 Primeiro Teste . . . 43
5.7.2 Segundo Teste . . . 45
5.7.3 Terceiro Teste . . . 48
5.8 Conclusão . . . 51

6 Conclusões e Sugestões de Trabalhos Futuros 53
6.1 Conclusão . . . 53
6.2 Trabalhos Futuros . . . 54

Referências Bibliográficas 57

A Desenvolvimento do Artefato de Software 60
A.0.1 Requisitos do Software . . . 60
A.0.2 Diagrama de Caso de Uso . . . 61
A.0.3 Casos de Uso . . . 61
A.1 Eventos . . . 67
A.2 Diagrama UML . . . 72

B Manual de Instalação 73
B.1 Pacotes . . . 73
B.2 Configuração do Artefato . . . 74
B.2.1 Apache . . . 74

B.2.2 Cron 75
B.3 Adaptação para novos equipamentos 75

C Manual de Uso **78**
C.1 Criação de Nós 79
C.1.1 Inserção de Portas Trunks 79
C.2 Criação de LSPs 81
C.2.1 Agendamento 81

D Código do artefato **83**
D.1 Organização dos arquivos 83
D.2 Configuração do Framework 84
D.3 Configuração do arquivo core 89
D.4 Configuração da Aplicação 103
D.5 Configuração de Estilo 120

Capítulo 1

Introdução

Atualmente observa-se um grande crescimento na demanda por aplicações científicas de alto desempenho, como física de alta energia, astronomia, bioinformática, tele medicina, visualização remota, entre outras, denominadas *e-science*. Tais aplicações refletem uma nova realidade de pesquisa onde existe colaboração entre diversos grupos situados em localidades distintas, inclusive em outros continentes.

Algumas características dessas aplicações são o grande volume de informações manipuladas, que podem chegar à ordem de terabytes e petabytes de dados, e a sensibilidade a atrasos excessivos. Por essa razão, as redes que dão suporte a esse tipo de aplicação devem ser adaptadas para atender aos requisitos impostos pelas mesmas. Nesse contexto, as redes ópticas dinâmicas, baseadas na comutação puramente óptica, surgem como uma alternativa promissora para suportar essas aplicações (1).

Um desafio nessa área diz respeito ao gerenciamento e controle dessas redes de forma automática e dinâmica, o que é importante quando se faz necessário o estabelecimento de conexões com diferentes granularidades, que podem atravessar vários domínios administrativos e que possuem requisitos estritos de desempenho (2). Por essa razão se faz necessária a definição de soluções para compor os planos de controle e gerenciamento da rede.

Alternativas nesse sentido vêm sendo propostas nos últimos anos para constituir o plano de controle das redes ópticas dinâmicas, destacando-se a arquitetura GMPLS (*Generalized Multiprotocol Label Switching*).

A utilização de um plano de controle distribuído baseado em GMPLS (3) em redes ópticas dinâmicas é adequada para situações onde se deseja um estabelecimento rápido de conexões e facilidades de engenharia de tráfego. O GMPLS oferece essas funcionalidades por meio dos protocolos: OSPF-TE (*Open Shortest Path - Traffic Engineering*), para roteamento, RSVP-TE (*Resource Reservation Protocol - Traffic Engineering*), para sinalização e LMP (*Link Management Protocol*), para autodescoberta de recursos, além de ser facilmente integrado às redes ópticas de nova geração, onde os rótulos (*labels*) são associados aos comprimentos de onda da fibra. Em suma, o plano de controle GMPLS oferece inteligência ao núcleo da rede, o que é de fundamental importância para aplicações avançadas de rede, onde se deseja um controle melhorado dos recursos.

Cabe ressaltar, no entanto, que os equipamentos utilizados atualmente nos núcleos de redes de pesquisa não possuem as extensões necessárias para que os protocolos propostos pelo GMPLS possam interagir adequadamente. Além disso, os novos equipamentos que possuem tal capacidade ainda apresentam custos elevados e acabam inviabilizando a mudança para as novas funcionalidades do plano de controle.

Assim, diversas iniciativas no mundo estão sendo implementadas em software para dar capacidade ao plano de controle para realizar alocações dinâmicas sem que os equipamentos de rede utilizados até então tenham necessidade de serem substituídos.

Duas soluções de alocação dinâmica de recursos executadas em software foram extensivamente estudadas dentro de um projeto de cunho nacional com o intuito de trazer estas capacidades para a rede que a instituição gerencia. Ao longo dos estudos, foram enfrentados diversos obstáculos para o correto funcionamento das soluções. Os empecilhos dizem respeito a dificuldades na instalação, manutenção e utilização dos softwares.

Assim, foi proposto o desafio de se desenvolver um novo software que fosse capaz

da alocação dinâmica de recursos, ao mesmo tempo que diminuisse os empecilhos enfrentados pelas duas soluções estudadas.

Esta dissertação portanto, explica os estudos feitos das soluções, os protocolos envolvidos para alocação dinâmica de recursos, o desenvolvimento realizado para se chegar ao artefato de software, assim como um comparativo em relação às soluções estudadas.

1.1 Motivação

Este trabalho surgiu da oportunidade do estudo dentro do Projeto TIAMHAT (4) de duas iniciativas, ambas buscando o objetivo de se criar um plano de controle em que fosse possível dar às redes de pesquisa a capacidade de alocação dinâmica de recursos.

Através das pesquisas percebeu-se que essas iniciativas se mostraram maduras por terem projetos pilotos em fase de testes. Uma delas é a iniciativa DRAGON/OSCARS (*Dynamic Resource Allocation via GMPLS Optical Networks/On-Demand Secure Circuits and Advance Reservation System*) (5) que é proveniente do Projeto Internet2 dos EUA e a outra é o AutoBAHN (*Automated Bandwidth Allocation across Heterogeneous Networks*) (6) proveniente do Projeto GÈANT da Europa.

Através do estudo dessas soluções, foi possível enriquecer o conhecimento e adquirir maior entendimento de como um plano de controle capaz da alocação dinâmica de recursos é implementado. E assim, trazer mais confiança para o desenvolvimento de um novo artefato que pudesse executar a alocação dinâmica.

1.2 Justificativa

A partir dos estudos e implementações produzidas no Projeto TIAMHAT, descobriram-se vantagens e desvantagens em cada uma das soluções citadas.

Ao longo dos estudos foi possível perceber diversos empecilhos de manutenção, instalação e utilização das soluções DRAGON/OSCARS e AutoBAHN. Após várias

pesquisas e um maior entendimento das tecnologias utilizadas na implementação dos softwares, foi possível projetar um artefato de software com as mesmas características e funcionalidades das iniciativas estudadas, com o adicional de ser mais simples na manutenção, instalação e utilização.

1.3 Objetivos Gerais

É desenvolvido um artefato de software com capacidade para alocação dinâmica de recursos em redes de pesquisa com o objetivo de se ter uma solução mais simples em sua instalação, manutenção e utilização em relação às soluções estudadas.

1.4 Objetivos Específicos

Propor uma implementação do software para alocação dinâmica de recursos que atinja os seguintes objetivos:

- Simplicidade na instalação, manutenção e utilização;
- Flexibilidade na adaptação para novos switches;
- Simplicidade na adição de módulos para novas funcionalidades;
- Facilidade na utilização com interface simples e intuitiva.

1.5 Metodologia

Esta dissertação foi desenvolvida através de:

- Pesquisa em livros e artigos científicos;
- Pesquisa na internet;
- Leitura de RFCs (*Request for Comments*);

- Leituras de documentações dos softwares estudados;
- Testes dos softwares em redes experimentais do laboratório (Labtel2) e RNP (Rede Nacional de Ensino e Pesquisa);
- Testes do artefato de software proposto em rede experimental do laboratório (Labtel2).

1.6 Principais Resultados

Implementou-se um artefato de software funcional, que atinge aos objetivos citados, trazendo simplicidade na sua instalação, utilização, manutenção e inserção de novas funcionalidades. Tornou possível um maior entendimento na criação de planos de controle e um facilitador para que usuários da comunidade científica adotem esse artefato de software e o aperfeiçoem.

1.7 Organização do Texto

O trabalho está organizado da seguinte maneira: o capítulo 2 faz um breve levantamento de trabalhos relacionados com alocação dinâmica de recursos e experiências desenvolvidas por vários grupos de pesquisadores, no capítulo 3 é descrito o protocolo GMPLS e também são explicadas as tecnologias DRAGON/OSCARS e AutoBAHN, como foram pensadas e implementadas suas arquiteturas; o capítulo 4 é destinado à explicação dos protocolos que possibilitam a alocação dinâmica de recursos e também como os protocolos atuam para que a alocação seja feita; o capítulo 5 é destinado a explicar como o artefato proposto foi desenvolvido, além de uma comparação entre as soluções e os testes realizados para validação do mesmo; e finalmente o capítulo 6 explana sobre as conclusões do trabalho e trabalhos futuros que poderão ser feitos com base neste novo artefato.

Capítulo 2

Trabalhos Relacionados

2.1 Introdução

Neste capítulo é apresentado um breve levantamento de trabalhos relacionados com alocação dinâmica de recursos e experiências desenvolvidas por vários grupos de pesquisadores espalhados pelo mundo.

2.2 Experimentos de Alocação Dinâmica de Recursos

Segundo (7) a internet do Futuro deve ser baseada em configuração de circuitos e a Dynamic Circuit Networking (DCN) é a chave para a próxima geração da infraestrutura da Internet. Porém a DCN está nos seus estágios de desenvolvimento iniciais e possui dois problemas a serem enfrentados: o primeiro, um algoritmo ideal para alocar de maneira eficiente o recurso com a banda requerida dentro de uma rede dinâmica de recursos; o segundo, mecanismos dos protocolos para disseminar e coletar informações de roteamento dentro do domínio DCN.

Os autores sugerem uma arquitetura de rede e extensões com base no MPLS com capacidade para a alocação dinâmica de recursos. A arquitetura sugerida é baseada em

quatro camadas: a rede física, camada de agendamento, camada de roteamento e ponta (interface de requisição dos usuários).

Como complemento ao argumento de que a Alocação Dinâmica de Recursos ainda está em fase inicial encontra-se o artigo (8) que argumenta que o assunto ainda é recente e encontra-se em uma fase ainda de intensa discussão e desenvolvimento. Deixa-se claro no mesmo artigo que somente redes dessa natureza são capazes de suportar a demanda das redes de *e-science*.

Os autores de (8) ainda sugerem um plano de controle que deve possuir a capacidade de aprovisionamento de circuitos em uma variedade de redes distintas e diversas camadas de tecnologia. Para isso, chega-se à conclusão de que a infraestrutura que surgir das diversas pesquisas sobre o assunto, virá dos melhores esforços das soluções de redes atuais e que será extremamente heterogênea tanto no plano de controle quanto no plano de dados.

Segundo o mesmo artigo (8), uma possível solução de plano de controle capaz de operacionalizar todas essas características chaves de: roteamento, computação de caminhos e sinalização, será através de uma solução através do protocolo GMPLS. A solução experimentada e implementada deu-se nas redes ESnet (*Energy Sciences Network*), USN (*UltraScience Network*) e Internet2. O trabalho prático é fruto de uma adaptação do software DRAGON/OSCARS (abordado na seção 3.3) para incluir os novos serviços de multidomínio através de webservices.

Por outro lado, o artigo (9) fala sobre o isolamento entre as redes de ensino e pesquisa e as redes atuais de produção. Os autores propõem que o isolamento pode ser contornado através de infraestruturas virtuais, assim, o FEDERICA (*Federated E-Infrastructure Dedicated to European Researchers Innovating in Computing Network Architectures*) se propõe a ser um ambiente de virtualização (tanto de computação como de recursos de rede) que proverá o ambiente necessário para as pesquisas da Internet do Futuro. O FEDERICA trabalhou em conjunto com experimentos da rede GpENI

(*Great Plains Environment for Network Innovation*), uma rede de pesquisa nos EUA para Alocação Dinâmica de Circuitos.

No artigo (10) encontra-se uma explanação sobre a experiência de se implementar circuitos dinâmicos dentro da rede GpENI. A rede é criada através do backbone da Internet2 e a solução experimentada no artigo se utiliza do software OSCARS para a alocação dinâmica dos recursos.

Por último, o artigo (11) apresenta como foi desenhada e produzida uma implementação de uma rede orientada ao usuário de forma a garantir a banda requisitada. A solução proposta foi implementada dentro da rede ESnet e se utiliza dos protocolos RSVP-TE, OSPF-TE e MPLS-TE (*Multiprotocol Label Switching - Traffic Engineering*). É utilizado a solução OSCARS como um controlador de domínio dos recursos da rede dentro da ESnet.

2.3 Conclusão

Conclui-se que o assunto abordado nesta dissertação encontra-se ainda em uma fase de experimentação e desenvolvimento de conceitos, como é possível notar pela pequena quantidade de artigos abordando o assunto. Porém é possível identificar certas tendências, através das pesquisas feitas, de que estão sendo buscadas soluções dentro dos protocolos que compõe o MPLS e em um futuro próximo o GMPLS. Percebe-se também que a maioria das experimentações testadas são baseadas em software usando a combinação DRAGON/OSCARS e adaptações dos mesmos, deixando claro que uma melhora em suas funcionalidades é bem-vinda.

Capítulo 3

Arquiteturas de Alocação Dinâmica de Circuitos

3.1 Introdução

Neste capítulo é apresentada uma descrição sucinta do GMPLS e das tecnologias estudadas dentro do Projeto TIAMHAT, o DRAGON/OSCARS e o AutoBAHN. São descritas as arquiteturas dos softwares, o processo em cada solução no momento da alocação de recursos e também as vantagens e desvantagens de cada uma.

3.1.1 Projeto TIAMHAT

O Projeto TIAMHAT (Tecnologias de Aprovisionamento Dinâmico de Conexões para Redes Híbridas) é um projeto integrante da primeira fase de um projeto maior chamado de Futura RNP. O TIAMHAT tem como objetivo estudar as tecnologias existentes de alocação dinâmica de recursos. Fazem parte dessa força tarefa pesquisadores de Universidades do Espírito Santo, Rio de Janeiro, Ceará e Pará.

O Projeto Futura RNP é um projeto da Rede Nacional de Ensino e Pesquisa (RNP) com o objetivo de implementar uma rede de nova geração dentro de seu *backbone*.

Assim, diversas entidades acadêmicas do Brasil foram convocadas para que fossem estudadas e testadas as soluções existentes no mundo. Dentro do projeto, chegou-se à conclusão de que as soluções mais plausíveis a serem estudadas, para posteriormente serem implementadas, seriam a solução DRAGON/OSCARS e a solução AutoBAHN.

O projeto encontra-se (no momento da escrita desta dissertação) em sua fase de testes piloto com uma adaptação da solução DRAGON/OSCARS implementada.

Essas duas soluções tem suas interações baseadas no comportamento do GMPLS (*Generalized Multiprotocol Label Switching*) e por isso, convém explicá-lo primeiramente.

3.2 GMPLS

O GMPLS (*Generalized Multiprotocol Label Switching*) define protocolos de roteamento e de sinalização para a criação de *Label Switched Paths* (LSPs) em várias tecnologias de transporte (3) e (12).

O GMPLS é uma tecnologia derivada do MPLS (*Multiprotocol Label Switching*), que executa o transporte de pacotes, desenvolvida pela *Internet Engineering Task Force* (IETF). Os protocolos são baseados na identificação de cada pacote com um identificador (*label*), através do qual cada roteador pode determinar para onde o pacote deve ser encaminhado.

Em uma rede MPLS, os pacotes são rotulados pela inserção de um pedaço adicional de informação chamado de *Shim Header*.

3.2.1 *Shim Header*

O *Shim Header* do MPLS possui um *label* de 20 bits que é usado para determinar o caminho em que o pacote deve seguir (Figura 3.1). Cada nó na rede, chamado de LSR (*Label Switching Router*), que na verdade é um roteador capaz de comutar *labels*,

mantém uma tabela de busca LFIB (*Label Forwarding Information Base*) que determina o próximo nó do pacote.

O LFIB contém um mapeamento de interface de entrada, *label* de entrada para interface de saída, *label* de saída. Assim, quando um pacote é recebido, o LSR, ao saber por qual interface entrou, consegue determinar o *label* do *shim header* do pacote. São buscados os valores no LFIB para descobrir para qual interface de saída deve ser encaminhado o pacote e que novo *label* deve ser colocado.

O caminho que o pacote MPLS segue pela rede é chamado de *Label Switched Path* (LSP). Uma vez que o pacote foi "rotulado" no começo do LSP, seu caminho até o ponto final já é sabido por causa do mapeamento no LFIB que contém a informação a respeito de cada LSR por onde ele deve passar.

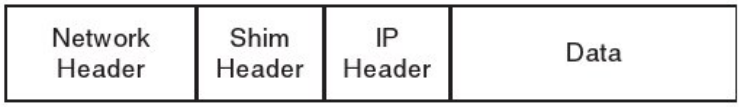

Figura 3.1: Shim Header inserido entre o Protocolo de Rede e o Cabeçalho IP (3).

A montagem das tabelas LFIBs se faz através das ferramentas de rede. Com um Sistema de Gerenciamento de Rede (*Network Management System - NMS*) que envia mensagens de gerenciamento para cada LSR visando estabelecer os mapeamentos de labels necessários. Isso é perfeitamente aceitável em uma rede pequena, porém torna-se inviável à medida que essa rede cresce, por ser necessária uma grande troca de mensagens entre um ponto central e os LSRs pertencentes à rede.

O GMPLS utiliza o conceito de label para estender as funcionalidades do MPLS para outras tecnologias de transporte. Em redes de transporte, os recursos físicos são exatamente as quantidades a serem comutadas. Por exemplo, em uma rede WDM (*WaveLength Division Multiplexing*) os lambdas são comutados, em uma rede TDM (*Time Division Multiplexing*) os *timeslots* são comutados.

Assim, um label que identifica o caminho de dados também identifica precisamente o recurso físico. Portanto, em uma rede de comutação de lambdas, um label identifica um comprimento de onda específico, em uma rede TDM o label identifica um timeslot específico e em uma rede de comutação de fibras o label identifica uma porta específica ou uma fibra.

3.2.2 LSP Tunnel

Algumas vezes um ou mais LSPs podem ser agrupados dentro de um único LSP, o que é chamado de tunelamento de LSPs ou *LSP Tunnel*. Esse procedimento traz grandes benefícios para o núcleo da rede, pois minimiza o custo de gerenciamento de vários LFIB ao mesmo tempo. Esse tipo de agrupamento leva a cenários variados de aplicações como a construção de VPNs (*Virtual Private Network*), e o gerenciamento de um grande número de LSPs de maneira menos custosa ao núcleo da rede.

3.2.3 Arquitetura

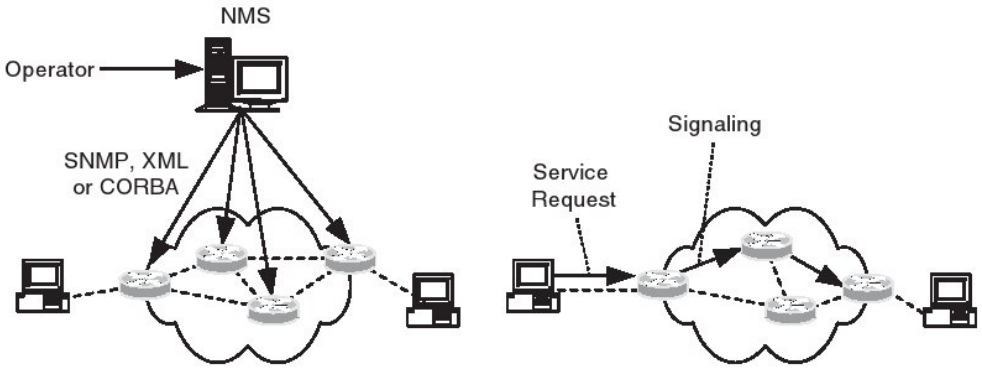

Figura 3.2: Arquitetura Centralizada e Descentralizada (3).

Como mostrado na figura 3.2, é possível ver dois tipos de diagramas de arquitetura: Centralizada à esquerda e Descentralizada à direita. A Arquitetura Centralizada ocorre quando o controle e configurações dos nós para estabelecimento de LSPs ficam centrais

a um software ou máquina (*Network Management System*), enquanto que a Arquitetura Descentralizada ocorre quando o controle e configurações dos nós para estabelecimento de LSPs são realizadas nos próprios nós (3).

O GMPLS tende a seguir uma arquitetura descentralizada, isso só se faz possível através dos protocolos de sinalização (Capítulo 4). Cada LSR mantém sua tabela LFIB atualizada de acordo com as mensagens trocadas entre seus nós adjacentes. Cada par de interfaces troca as mensagens necessárias para saber os recursos disponíveis naquele enlace e assegura um label para isso.

Assim (parte direita da figura 3.2) é possível ter uma solução mais flexível e escalável com grandes números de nós em um mesmo domínio. O serviço de requisição é enviado para a rede, e a rede é que fica responsável pelo roteamento do LSP através dos LSRs.

Com isso são adicionadas as funcionalidades de comutação de diversas tecnologias de transporte como:

- Pacotes (comutação baseada no MPLS *shim header*)
- Camada 2 (*Layer* 2) (comutação baseada no cabeçalho de camada 2 como ATM VPI/VCI (*Asynchronous Transfer Mode Virtual Path Identifier/Virtual Channel Identifier*), Vlans)
- Timeslot (TDM) (comutação baseada na divisão de *slots* (unidades) de tempo)
- Lambda (comutação baseada na divisão de comprimentos de ondas)
- Waveband (comutação baseada na divisão de grupamentos de lambdas)
- Fibra (comutação baseada na divisão por fibras ou portas)

3.3 DRAGON e OSCARS

Nesta seção os softwares DRAGON e OSCARS são explicados em separado por serem softwares distintos. Eles foram integrados a partir de um momento pois percebeu-se que suas funções eram complementares. O esclarecimento da integração dos dois softwares é melhor explicado nas seção DRAGON/OSCARS 3.4.

3.3.1 DRAGON

O software DRAGON é resultado do projeto *Dynamic Resource Allocation via GMPLS Optical Networks* patrocinado pelo *National Science Foundation* (NSF) (5). Atualmente o DRAGON é parte integrante do *Dynamic Circuit Network Software Suite* (DCN) do projeto Internet2, um consórcio de redes avançadas nos EUA com o intuito de promover e facilitar o intercâmbio de informações entre pesquisadores para desenvolver novas tecnologias para a Internet. Mais informações em (13).

Objetivo

O projeto DRAGON/OSCARS junto com o DCN tem por objetivo criar mecanismos que permitam aprovisionar recursos dinamicamente através de redes de múltiplos domínios administrativos. A idéia é que, com esses mecanismos, seja possível garantir a um usuário final a requisição de recursos (exemplo: banda, número de VLAN, etc) que sejam automaticamente aprovisionados por software e liberados quando não forem mais necessários (14).

Arquitetura

O projeto DRAGON desenvolveu um plano de controle baseado em GMPLS que é composto dos seguintes componentes principais:

- NARB (*Network Aware Resource Broker*): Componente responsável por gerenciar um domínio e gerenciar requisições inter-domínio. Possui as funções de: roteamento inter-domínio, abstração de topologia, gerenciamento de LSPs e aplicação de políticas.

- VLSR (*Virtual Label Switch Router*): Oferece uma solução para o plano de controle intra-domínio, permitindo que os dispositivos não-GMPLS possam fazer parte do estabelecimento de caminhos fim a fim.

- CSA (*Client System Agent*): Software que participa do protocolo GMPLS, que fica nos limites do plano de dados e que permite a alocação fim-a-fim do sistema cliente.

- ASTB (*Application Specific Topology Builder*): Permite que uma aplicação faça requisições de LSPs específicos.

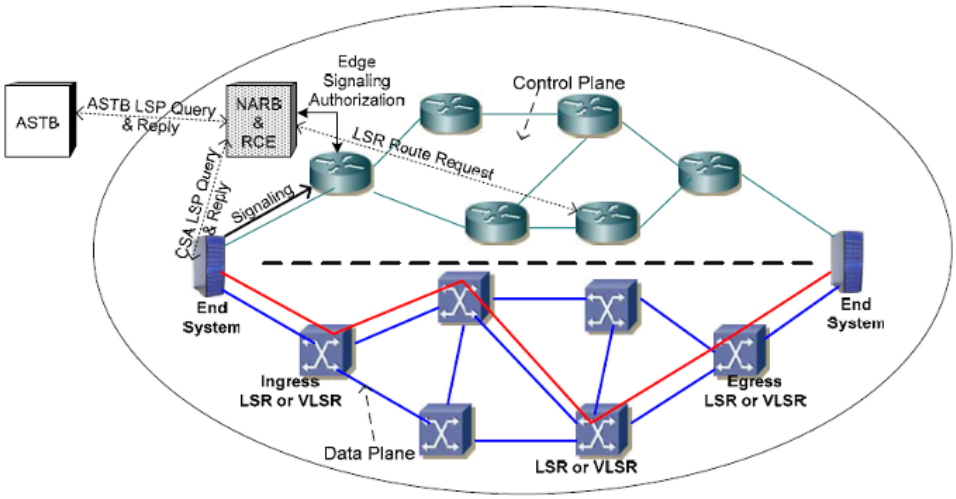

Figura 3.3: Diagrama de uma rede com o software DRAGON (15).

Como mostrado na figura 3.3, o software DRAGON é instalado em cada máquina (podendo ser virtual) que deve controlar um switch através do módulo VLSR, outra

máquina (podendo ser também virtual) deve executar o DRAGON com o módulo NARB (15).

Percebe-se que o software DRAGON tenta emular a arquitetura descentralizada derivada do GMPLS, pois seu componente VLSR é instalado no que é chamado de nó. Cada nó pertencente à rede administrada é formado por uma máquina (computador) conectado a um *switch*. Essa máquina, junto ao componente DRAGON instalado, é quem recebe e requisita configurações no *switch* para a alocação dinâmica de recursos.

Esses componentes instalados nas máquinas também são responsáveis por enviar informações dos switches a que estão conectados, para o componente NARB. O componente NARB deve ser instalado em uma máquina dentro do domínio administrativo. É ele o responsável pelos cálculos de caminhos e troca de mensagens de requisições no próprio domínio administrativo, além de lidar com envio e recebimento de informações de domínios adjacentes.

Conclui-se que o componente VLSR faz o papel de um switch que possui implementação dos protocolos próprios para GMPLS e o NARB faz o papel de um NMS (*Network Management System*), gerenciando e fazendo requisições de LSPs à rede.

3.3.2 OSCARS

Objetivo

O OSCARS (*On-Demand Secure Circuits and Advance Reservation System*) teve como objetivo inicial prover um serviço de protótipo que permite o aprovisionamento dinâmico de recursos com garantia de banda da rede ESnet (*Energy Sciences Network*), atualmente seu objetivo é criar um serviço de alocação dinâmico de caminhos (LSPs) que seja simples para o usuário. A ESnet é uma rede de alta velocidade dos Estados Unidos que interliga centenas de pesquisadores de Departamentos de Energia em mais de 40 instituições e se conecta em 100 outras redes, mais informações em (16) e (17).

Arquitetura

O OSCARS é formado por quatro componentes:

- *Reservation Manager* (RM) - Responsável por trocar mensagens de reserva de recursos através do padrão de linguagem *Web Service Description Language* (WSDL) usando protocolo padrão *Simple Object Access Protocol* (SOAP), o que permite interoperabilidade com aplicações diferentes. Ele é responsável também pela comunicação com domínios OSCARS adjacentes com o intuito de prover circuitos inter-domínio.
- *Authentication, Authorization and Auditing Subsystem* (AAAS) - Responsável por autenticar e autorizar os usuários que desejam utilizar o sistema para requisitar LSPs.
- *Bandwidth Scheduler Subsystem* (BSS) - É o responsável por monitorar as reservas feitas nos links dentro da infraestrutura ESnet.
- *Path Setup Subsystem* (PSS) - É o responsável por estabelecer e destruir os LSPs requisitados pelo sistema.

O software também oferece uma interface baseada em Web, dentro do padrão *Web-Based User Interface* (WBUI), o que dá maior usabilidade e capacidade de gerência da rede para os usuários no momento da requisição de LSPs, facilitando o seu uso no modo gráfico ao invés de linhas de comandos em um prompt, como é feito com o DRAGON. Por essa facilidade em sua interface gráfica, ele foi eleito para ser utilizado junto com o DRAGON na composição do pacote DCN Suite da Internet2.

3.4 DRAGON/OSCARS

O pacote com os softwares DRAGON e OSCARS juntos, formam o que comumente se chama DRAGON/OSCARS dentro do pacote DCN Suite da Internet2 (13). Eles

são integrados através de uma API (*Application Programming Interface*) fornecida pelo OSCARS dentros dos padrões de linguagens WSDL e o padrão de protocolo SOAP, possibilitando a comunicação entre os dois softwares.

O OSCARS tem como responsabilidade, dentro do pacote da Internet2, executar as funções de autenticação dos usuários, agendamento dos LSPs e monitoramento dos LSPs que já estão em uso, enquanto que o DRAGON é utilizado para a configuração dos switches dentro de um domínio administrativo. Em outras palavras, o OSCARS executa os componentes RM, AAAS e BSS, e o DRAGON executa a função do PSS.

O software pesquisado no Projeto TIAMHAT foi a solução do pacote DCN Suite da Internet2 com os dois softwares DRAGON e OSCARS já integrados.

3.5 AutoBAHN

O sistema *Automated Bandwidth Allocation across Heterogeneous Networks* (AutoBAHN) é um software piloto dentro do projeto GN2 que é patrocinado pela Comissão Européia. As atividades desenvolvidas para a alocação dinâmica de circuitos são focadas com o intuito de desenvolver a nova geração da rede GÉANT2, usando tecnologias de transporte para adicionar novos serviços baseados na rede IP.

A rede GÉANT2 é uma rede acadêmica de banda larga que serve a comunidade de ensino e pesquisa na Europa. Ela conecta mais de 30 milhões de pesquisadores através de múltiplos domínios e topologias, sendo distribuída por cerca de 30 países. Mais informações em (18).

3.5.1 Objetivo

O sistema AutoBAHN foi pensado como um software capaz de complementar o plano de controle existente na rede em que atua, além de adicionar um plano de controle quando este não existe. Ele provê uma camada de negócios para a coordenação de aprovisio-

namento entre domínios através de funções de autenticação e autorização, roteamento inter-domínio e monitoramento dos roteamentos.

O software foi desenvolvido pensando-se que ele deveria ser o intermediário entre os usuários (ou aplicações) e a rede, ou seja, ele interpreta as intenções de uso da rede e as traduz para a rede. A própria rede deve interagir com o software dando informações sobre serviços disponíveis e seus parâmetros de qualidade.

Como o software foi desenvolvido para atuar na rede européia, foi pensado que o sistema deveria interligar vários tipos de tecnologias em redes diferentes. Assim o software foi desenvolvido com o intuito de suportar: circuitos de camada 2 (*Ethernet Private Line* e *Ethernet Virtual Private Line* - VLANs) e cicuitos de camada 1 (STM-1 to optical wavelengths, 1 GE, 10 GE).

3.5.2 Arquitetura

O software do AutoBAHN é baseado em módulos. O software como um todo fica instalado somente em uma máquina dentro do domínio (equipamentos cadastrados e gerenciados pelo mesmo administrador de rede).

Assim, pode-se notar que a arquitetura do AutoBAHN é claramente centralizada, que, através de seus módulos, recebe as informações dos equipamentos cadastrados e a partir das informações da rede junto às informações que o usuário (ou aplicação) deseja fazer, executa as ações necessárias para tal.

Os módulos são divididos em *Inter Domain Manager* (IDM) e *Domain Manager* (DM) com seu módulo interno *Technology Proxy* (TP).

O *Inter Domain Manager* (IDM) é o módulo responsável pelas operações de inter-domínio e de reservas de circuitos no domínio. Isso inclui todas as atividades necessárias para o monitoramento do domínio, como comunicação inter-domínio, negociação de recursos com domínios adjacentes, processamento de requisições e divulgação da topologia.

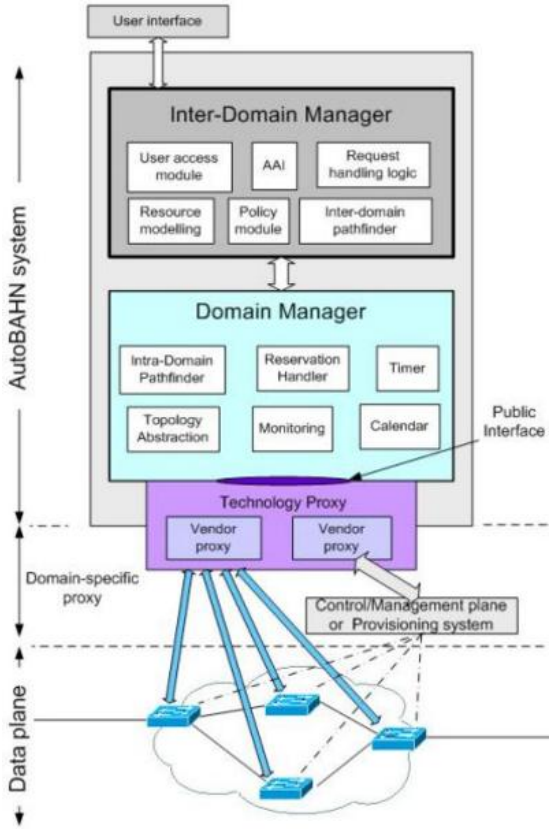

Figura 3.4: Diagrama dos módulos AutoBAHN (6).

O *Domain Manager* (DM) é o módulo que controla o plano de dados do domínio e possui funcionalidades como interfaceamento com o sistema de gerenciamento da rede e sinalização de protocolos.

O *Domain Manager* também participa na administração de recursos inter-domínio. O módulo consegue trabalhar, através de uma interface com o IDM, em funções como: abstração da topologia, agendamento, reserva de recursos e monitoramento.

O *Technology Proxy* é o módulo que faz a interface com os equipamentos dentro da rede, e assim encaminha as informações necessárias para o Domain Manager. Ele possui a característica de ser adaptável com os equipamentos com quem ele se comunica. E pode ser extensível através da inserção de novos módulos para diferentes equipamentos.

Como mostrado na figura 3.4, na parte inferior ficam os nós da rede que o soft-

ware controla através de comandos específicos do módulo *Technology Proxy* que troca informações com os módulos *Domain Manager* e *Inter Domain Manager* (6).

3.6 Vantagens e Desvantagens

Através do Projeto TIAMHAT foi possível obter experiências práticas sobre a instalação, manutenção, usabilidade e escalabilidade desses softwares e percebeu-se que ainda não são robustos suficientes para entrarem em uma rede de produção.

Os softwares cumprem o objetivo citado no que se refere à alocação dinâmica de recursos, porém a um custo muito alto de falhas, excesso de instalações de componentes com funcionalidades ainda não compreendidas, documentações escassas, interface gráfica pouco intuitiva e, o principal, a falta de escalabilidade e possibilidade de expansão do software em um tempo razoável.

O principal agravante do sistema AutoBAHN é o fato de não possuir seu código aberto para modificações e, apesar de haver a possibilidade de alterações na idealização do software, não foi possível, mesmo através do Projeto Futura RNP, receber qualquer autorização para o acesso ao código.

Por outro lado, o sistema DRAGON/OSCARS possui seu código aberto e seu agravante se encontra na não modularização do código, uma vez que ao se fazer modificações para a compatibilidade com os equipamentos disponíveis dentro do Projeto TIAMHAT, um grande esforço foi despendido para reconstruir quase todo o software apenas para a inclusão das modificações necessárias. Ou seja, o software ainda é pouco escalável, pois para cada novo equipamento, praticamente um novo software deve ser feito com suas adaptações.

Um quadro comparativo é apresentado na seção 5.5 com as principais diferenças entre as soluções estudadas e o artefato proposto.

Capítulo 4

Os protocolos

4.1 Introdução

Este capítulo apresenta uma descrição mais detalhada sobre os protocolos necessários para que a alocação dinâmica de recursos ocorra e como eles interagem entre si para que o LSP seja estabelecido nas soluções estudadas. Alguns desses protocolos não são utilizados por completo nos sistemas estudados, porém convém entendê-los, pois as arquiteturas e funcionamento foram baseados em suas funções.

4.2 RSVP-TE

A sinalização do GMPLS é construída a partir das mensagens do RSVP-TE *Resource Reservation Protocol - Traffic Engineering*, que é derivado das especificações do RSVP. O GMPLS teve sua sinalização desenvolvida como mensagens abstratas que mapeiam as mensagens RSVP-TE, permitindo que a arquitetura do GMPLS fique aberta para a implementação de outros protocolos de sinalização, porém o utilizado hoje é o RSVP-TE.

O protocolo RSVP é usado por um *host* para requisitar reservas de um fluxo de

dados específico de uma aplicação particular dentro da rede. O protocolo trabalha em IPv4 ou IPv6 e foi desenvolvido para trabalhar em conjunto com um protocolo de roteamento, pois o RSVP tem a capacidade apenas de requisitar reservas para um serviço específico que a rede pode oferecer, enquanto o protocolo de roteamento é o responsável por garantir a Qualidade de Serviço (QoS) (19).

O RSVP se utiliza de mecanismos de sinalização para executar tarefas como classificação de pacotes, controle de admissão e agendamento das reservas. No momento de requisição de uma reserva, o RSVP verifica se o enlace possui os recursos e se o usuário tem permissão para executar tal tarefa, caso contrário o RSVP retorna uma mensagem de erro.

O RSVP-TE foi desenvolvido para qualquer tipo de interface: pacote, layer-2, *time-division multiplexed*, lambda, ou comutação de fibra com o objetivo de ter um nível o mínimo possível de intervenção manual no processo de engenharia de tráfego (20).

4.2.1 Mensagens RSVP-TE

Existem dois tipos fundamentais de mensagens no RSVP: *Path* e *Resv*.

Cada host envia uma mensagem de requisição de reserva RSVP (*Path*) para o próximo nó. Essas mensagens devem seguir exatamente o caminho reverso que as mensagens Resv irão utilizar. Eles criam e mantém um estado de reserva em cada nó ao longo do caminho. Cada host transmite uma mensagem de Path para os nós anteriores no caminho selecionado. Essas mensagens Path gravam o "estado path" em cada nó do caminho. Esse estado path inclui pelo menos o endereço IP do dispositivo anterior que é usado para encaminhar as mensagens Resv na direção reversa.

A mensagem Path contém as seguintes informações em relação ao próximo dispositivo:

Sender Template - Descreve o formato dos pacotes de dados que o enviador irá originar. Os sender templates contêm exatamente o mesmo format e filter specs que

aparecem nas mensagens Resv. Portanto, o Sender Template pode também especificar apenas o endereço IP do enviador e opcionalmente a porta UDP/TCP e assume o protocol ID especificado para a sessão.

Sender Tspec - Define as características do tráfego de dados que o disposivito anterior irá gerar. Esse Tspec é usado para prevenir mais de uma reserva do mesmo recurso.

Adspec - Informação que é recebida pelo nó e encaminhada para o controlador de tráfego local (utilizando a técnica do *OPWA - One Pass With Advertising*) a fim de atualizar a informação, assim que é atualizada, essa informação é gravada no nó e encaminhada na mensagem para os nós anteriores.

As mensagens Path são enviadas com os mesmos endereços de origem e destino dos dados, assim eles são encaminhados corretamente através de nós não-RSVP. Por outro lado, mensagens Resv são enviadas salto-a-salto.

O OPWA é utilizado para obter informações que podem prever um caminho fim-a-fim de QoS (*Quality of Service*). O resultado das divulgações que são enviadas pelo RSVP pode ser usado pelo dispositivo inicial para construir ou dinamicamente ajustar uma requisição apropriada de reserva.

Os outros tipos de mensagens que o RSVP possui são: PathErr e ResvErr (Mensagens de erro), PathTear e ResvTear (Mensagens de liberação do LSP) e ResvConf (mensagem de confirmação do LSP).

As mensagens de sinalização do GMPLS RSVP-TE são carregadas por datagramas IPs enviados entre controladores de sinalização. Isso é uma função herdada da especificação RSVP original, com funções modificadas para que as mensagens do plano de controle sigam um caminho bem definido entre os controladores de sinalização e possam estabelecer um caminho mais estável no plano de dados. A sinalização GMPLS é encapsulada dentro de datagramas IP que são endereçados para o próximo controlador de sinalização.

Cada mensagem GMPLS RSVP-TE é construída de acordo com um molde (*template*) comum. A mensagem dentro do datagrama IP começa com um cabeçalho da mensagem (*message header*). Esse cabeçalho é comum a todas as mensagens de sinalização e identifica o tipo e o tamanho da mensagem. O cabeçalho também inclui um *checksum* para detectar qualquer corrupção da mensagem, uma vez que nenhum protocolo de transporte é usado para prover essa funcionalidade.

4.2.2 Objetos RSVP-TE

Dentro das mensagens de sinalização RSVP-TE logo após o cabeçalho comum a todas as mensagens, seguem os objetos. Cada objeto é usado para codificar um conjunto de informações, por exemplo, o parâmetro que descreve o fluxo de dados e a banda para o LSP.

Os objetos são codificados de maneira padrão através de *type-length-variable* (TLV). Cada objeto pode ser construído por uma série de sub-objetos codificados também como TLV com suas especificações obrigatórias para que seja válida. Há também a possibilidade de adição de objetos opcionais para inclusão de funções.

Os objetos mais utilizados dentro das mensagens RSVP-TE são: LABEL - Através de uma mensagem Path é inserido um objeto de LABEL-REQUEST que requisita, dentro do caminho calculado, o label a ser definido para o LSP, que, em caso de sucesso, é retornado através de um objeto adicionado no RSVP-TE, LABEL, que é propagado para a rede através das mensagens Resv.

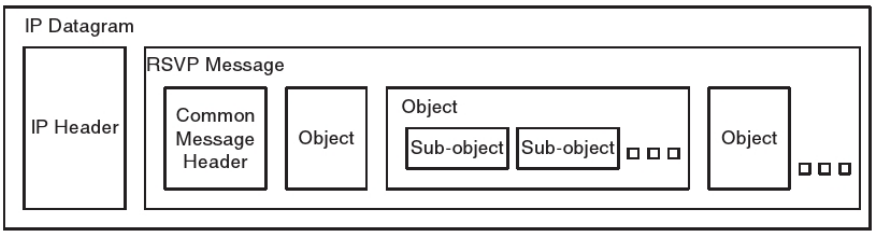

Figura 4.1: Objeto RSVP-TE (3).

EXPLICIT-ROUTE - Objeto que encapsula uma concatenação de vários saltos (hops) que constituem o caminho ao qual o LSP deve seguir. Através desse objeto os caminhos do LSP podem ser pré-determinados, independentes do roteamento convencional IP.

A figura 4.1 apresenta um diagrama de um objeto RSVP-TE sendo encapsulado por um datagrama IP formado por um cabeçalho comum e uma série de objetos (3).

4.3 OSPF-TE

OSPF *Open Shortest Path First* é um protocolo de roteamento de estado de enlace da arquitetura TCP/IP. Ou seja, ele é responsável por distribuir informações aos roteadores adjacentes sobre atividades dos enlaces e os custos de encaminhar dados através das interfaces. Cada roteador OSPF mantém uma base de dados idêntica descrevendo a topologia do *Autonomous System's*. Dessa base de dados, uma tabela de rotas é calculada construindo-se uma árvore de menor caminho (*Shortest-path tree*) (21).

Todos os pacotes OSPF trocados são autenticados. Isso significa que somente roteadores confiáveis podem participar no roteamento do *Autonomous System's*.

São cinco os tipos de pacotes que o OSPF troca: *Hello* (descoberta e manutenção de vizinhos), *Database Description* (Resumo do conteúdo da base de dados), *Link State Request* (Download da base de dados), *Link State Update* (Atualização da base de dados), *Link State Ack* (Divulgação por flooding).

A base de dados é formada através de vários LSAs *Link State Advertisement* gerados pelo roteador. É muito importante que todas as bases de dados dos roteadores fiquem sincronizadas. O OSPF simplifica isso fazendo com que somente os roteadores adjacentes permaneçam sincronizados. O processo de sincronização começa a partir do momento em que um roteador tenta estabelecer adjacência com outro.

Cada roteador descreve sua base de dados enviando uma sequência de pacotes de

descrição da base de dados *Database Description* para os vizinhos. Cada pacote de descrição da base de dados contém um conjunto de LSA pertencente ao roteador. Quando um vizinho percebe que há um LSA mais atual do que na sua própria cópia da base de dados, ele cria a notificação de que esse novo LSA deve ser requisitado.

Percebeu-se que o uso do OSPF faz os roteadores encaminharem os pacotes pelos mesmos caminhos de menor custo, fazendo com que ocorram congestionamentos eventuais nas redes. Para evitar isso, foi desenvolvido o OSPF-TE (*Open Shortest Path - Traffic Engineering*) para que os caminhos de dados sejam diversos, tendo o objetivo de se garantir determinadas quantidades de uso de recursos e qualidade de serviço.

Assim, foi desenvolvida uma base de dados, chamada de TED (*Traffic Engineering Database*) de informações complementares à base de dados do OSPF, com informações dos *TE Links* sobre os recursos disponíveis, como, por exemplo, a banda disponível, o máximo e mínimo de banda a ser reservada, etc.

É através dessas informações, distribuídas entre os LSRs dentro do domínio, que são calculados os caminhos para se formar um LSP com características específicas que determinada aplicação necessita.

O OSPF-TE possui a característica de ser opaco, ou seja, os roteadores que não possuem a capacidade de processar as informações do TED, apenas as recebem e reencaminham para outros roteadores, tornando a informação disponível para todos os roteadores do domínio e sendo informação útil apenas para aqueles que possam processá-lo. O OSPF-TE se utiliza do *Opaque Link State Advertisement*, uma função herdada do OSPF para realizar exatamente essa tarefa.

4.4 SNMP

O *Simple Network Management Protocol* SNMP é um padrão de protocolos utilizados para o gerenciamento de equipamentos de redes IP. Segundo (22) o SNMP em si é

apenas o protocolo necessário para a comunicação entre entidades de uma infraestrutura de gerência.

O seu modelo de gerenciamento inclui três entidades distintas - Agente, Gerente e o *Proxy*.

Agente é um software executado dentro de um equipamento que tem acesso à rede (23).

O Gerente executa ações como uma estação de gerenciamento de agentes SNMP que faz uma triagem através de agentes SNMP remotos em equipamentos da rede, em busca de informações sobre esses equipamentos. Agentes SNMP também podem sinalizar a estação quando alguma condição específica é atingida.

O Proxy é a entidade responsável pela comunicação do SNMP na rede.

Atualmente existem sete versões do protocolo (SNMPv1, SNMPsec, SNMPv2p, SNMPv2c, SNMPv2u, SNMPv2, e SNMPv3).

4.4.1 MIB

Variáveis em SNMP existem dentro de uma hierarquia chamada *Management Information Base* (MIB). A MIB, atualmente em sua versão MIB-II, define uma estrutura em árvore para todos os objetos que podem ser gerenciados através de SNMP.

Cada nível dentro da árvore representa instituições ou implementações feitas pelos fabricantes do produto gerenciado e, por padrão, cada nó na árvore é representada por um número. Assim, para cada objeto a ser gerenciado através do SNMP existe uma identificação dos números chamados de *Object Identifier* OID. Por exemplo o objeto SysDescr(1) pode ser identificado pelo OID 1.3.6.1.2.1.1.1, de acordo com o nível hierárquico para se chegar ao objeto, como mostra a figura 4.2, onde os retângulos em destaque apresentam o caminho que se deve seguir para alcançar o objeto (24).

Em qualquer implementação do SNMP, ao requisitar um OID é recebido um array (lista) de objetos. Para que o objeto em si seja manipulado e/ou lido é necessário

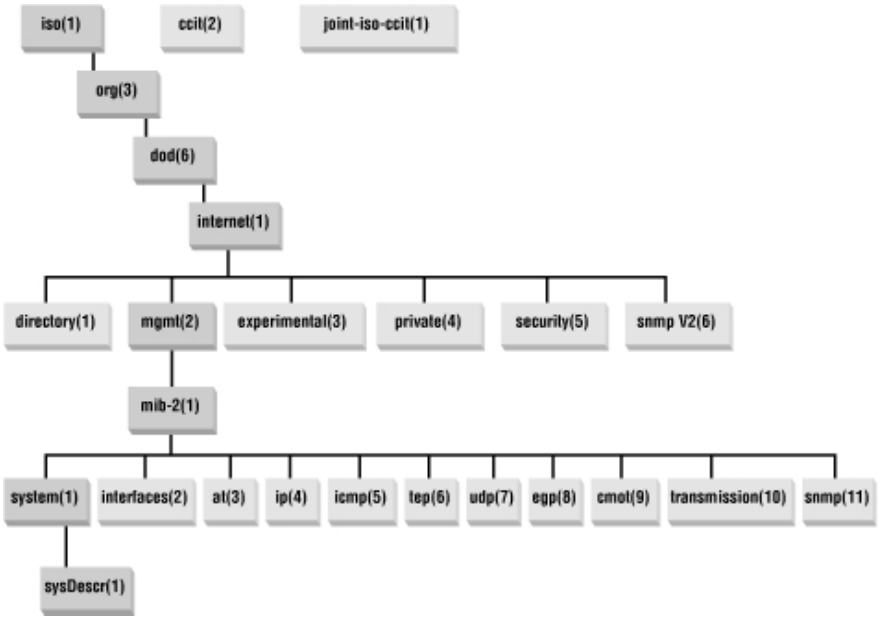

Figura 4.2: Diagrama da MIB com a estrutura em árvore (24).

adicionar o ".0" no OID para que seja recebido o primeiro objeto da lista requisitada.

Os comandos necessários ou PDU *Protocol Data Unit* existentes no SNMP são:

- get-request - Comando para requisitar alguma variável dentro do sistema
- get-next-request - Comando para requisitar uma lista (*array*) de variáveis seguinte ao item requisitado
- get-bulk-request - Comando para requisitar muitas variáveis ao mesmo tempo
- set-request - Comando para configurar alguma variável
- trap/snmpV2-trap - Sinaliza a partir de um evento ocorrido
- inform-request - Implementação no SNMPv2/3 igual ao trap porém com garantia de chegada do PDU através de uma confirmação
- response - PDU que carrega as respostas dos comandos dados ou também da sinalização enviada pelo trap

Objetos de gerência comumente utilizados nos equipamentos de rede foram reunidos e formaram o padrão do SNMP. Outros módulos mais específicos da MIB são normalmente criados pelos fornecedores de equipamentos de rede (23).

4.4.2 Segurança

O SNMP em seus primórdios possuía uma falha muito grande em sua segurança. Qualquer pessoa sabendo o endereço de um equipamento e de posse de algum objeto OID conseguia ter acesso e modificar suas variáveis. Nessa época foi até chamado de *Security is Not My Problem* (Segurança não é meu problema). Essa grande quantidade de versões do protocolo reflete a preocupação dos desenvolvedores em endereçar questões de segurança no protocolo.

Política de Segurança

Foi adicionado ao longo das versões do SNMP o que se chama de comunidade (*community*), assim é possível que um agente SNMP execute ações ou tenha acesso a informações de outros equipamentos apenas se ele pertencer à mesma comunidade.

Existem comunidades para apenas leitura ou para leitura e escrita de variáveis nos equipamentos remotos.

Na versão mais nova v3, através das RFCs 3414 e 3415 é proposta a definição de um *User Security Model* (USM) e um *View-Based Access Control Model* (VACM). Neles são utilizadas proteções criptografadas para a autenticação e para as mensagens, enquanto que o VACM utiliza um controle de acesso melhor implementado. Mais informações em (25).

4.5 Estabelecimento de LSPs

4.5.1 LSP no GMPLS

O estabelecimento de um LSP é iniciado pelo LSR de início do caminho enviando uma mensagem de LSP Setup para o próximo salto no caminho do LSP. Isso é determinado pela consulta na rota explícita do LSP ou pelo cálculo até o destino final. O LSP *Setup* carrega o identificador da sessão pai e o identificador do LSP, e também os parâmetros requisitados pelo LSP (Requisição de Label, Sender-TSpec, e Explicit Route objects). O LSP não existe até que seja aceito pelo último LSR, que envia uma mensagem de LSP *Accept* para recomendar o label que deve ser usado para identificar o tráfego e confirmar a reserva de recurso.

Uma possível troca de mensagem pode ocorrer enquanto ainda é enviado o pedido de LSP *Setup* para todos os LSR à frente e um LSR do caminho imediatamente aceitar o pedido. Esse procedimento pode induzir o LSR inicial a acreditar que o LSP já esteja estabelecido antes que a requisição tenha chegado ao LSR final e isso impede que os LSRs à sua frente falhem na configuração do LSP. O mecanismo usado, portanto, é como o mostrado na figura 4.3: O LSP *Setup* é encaminhado até o salto final. Em cada LSR os parâmetros do tráfego são checados para assegurar que o LSP pode ser suportado, e o próximo salto é determinado. Quando o LSP *Setup* atinge o LSR final, é criada a mensagem LSP *Accept* que é retornada salto por salto até atingir o LSR inicial.

A cada LSR, o label é divulgado da interface posterior para a anterior. Quando o LSP *Accept* é recebido pelo LSR inicial o caminho está pronto para transmitir os dados.

4.5.2 LSP no DRAGON/OSCARS

Como já explicado sobre sua arquitetura, o DRAGON/OSCARS tenta emular a descentralização que o GMPLS propõe e a sua maneira de realizar a alocação dinâmica de

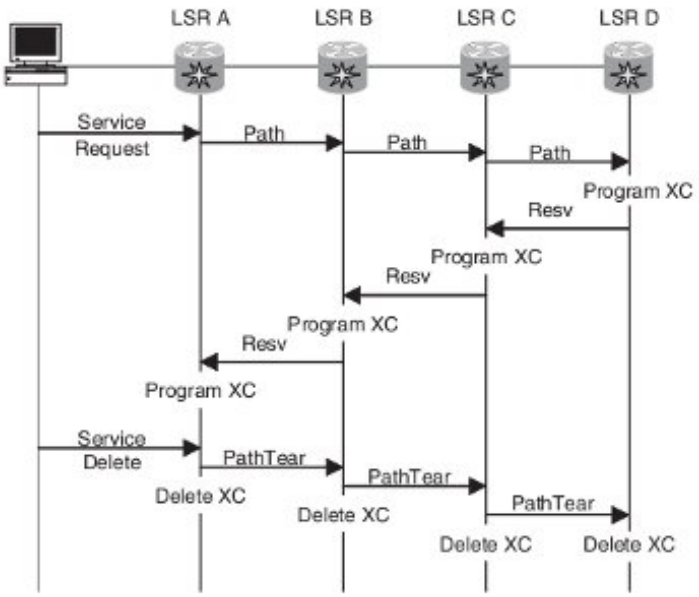

Figura 4.3: Diagrama de troca de mensagens GMPLS (3).

recursos.

Todos os cálculos de caminhos são executados pelo componente NARB e o LSP calculado é encaminhado para a rede através do RSVP-TE.

Tanto os protocolos RSVP-TE como OSPF-TE são presentes no software, e a interação entre os protocolos segue o mesmo procedimento. Porém, no momento em que são trocados as informações de reserva de recursos através do RSVP-TE, a alocação só ocorre graças a comandos CLIs (*Command Line Interface*) existentes nos nós para que a configuração dos LSPs sejam executados dentro dos switches.

4.5.3 LSP no AutoBAHN

Como a arquitetura do AutoBAHN segue a linha de um *Network Management System* NMS, centralizando todos os seus comandos em um software somente, para a troca de informações entre os switches de seu domínio ele se utiliza dos comandos CLIs e/ou do SNMP para receber de tempos em tempos as informações de links ativos, banda

disponível, etc.

Assim que surge a demanda de um novo LSP, o cálculo é feito pelo software com as informações obtidas da rede e todo o LSP é montado através do envio de mensagens SNMP e/ou comandos CLIs para as configurações dos switches participantes da rede.

Capítulo 5

Artefato de Software

5.1 Introdução

Neste capítulo é apresentado como o artefato foi idealizado e também as soluções aproveitadas das idéias dos softwares DRAGON/OSCARS e AutoBAHN, assim como um comparativo entre as soluções. Também é explicado como é o funcionamento interno do artefato desenvolvido, para esclarecer da melhor forma a solução proposta.

Além disso, são também descritas as tecnologias utilizadas para a implementação do protótipo, o Framework Django e PySNMP, que minimizaram o tempo de implementação do artefato. Neste capítulo também são apresentados os testes do protótipo implementado em uma rede experimental.

5.2 Visão Geral do artefato

O artefato foi construído para uma arquitetura centralizada, funcionando em uma máquina (Computador) e sendo capaz de administrar todos os switches em que é o controlador, assim como o AutoBAHN realiza as suas funções.

O artefato é constituído de três módulos: Módulo de Usuário, Módulo de Sistema,

Módulo de Rede.

Módulo de Usuário - módulo capaz de servir uma página web (através de qualquer programa *browser*) para que o usuário do sistema possa interagir com o sistema e fazer a demanda pelos serviços de rede. Troca mensagens com o Módulo de Sistema.

Módulo de Sistema - módulo que tem como função receber as demandas do usuário através do módulo do usuário e enviar as informações específicas para o Módulo da Rede. Interage com um sistema de banco de dados para persistência de dados.

Módulo de Rede - módulo que tem como função aplicar as funções específicas que o usuário demanda de acordo com as características dos equipamentos administrados. Interage com os equipamentos de rede através do protocolo SNMP.

A figura 5.1 apresenta os módulos citados com os frameworks de onde são derivados e também, em destaque, o módulo de adaptação para Switches Cisco.

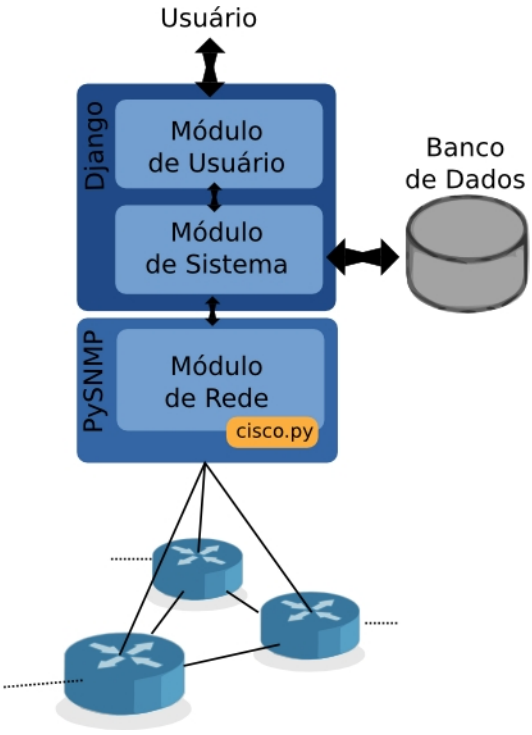

Figura 5.1: Diagrama do Artefato de Sistema com seus três módulos.

5.3 Adaptabilidade

Todo o artefato foi implementado através de dois Frameworks: Django e PySNMP. Ambos são construídos através da linguagem de programação Python. Os dois módulos (usuário e sistema) se utilizam do Django para funcionar, enquanto que o módulo de rede funciona através do PySNMP.

O módulo de sistema foi implementado para realizar funções básicas (criar vlan; estabelecer lsp, criar portas trunks, etc...) sempre avaliando as funções de acordo com o tipo de fabricante que o nó possui. O módulo de rede foi implementado para realizar as configurações de equipamentos de cada fabricante respeitando as suas especificidades de OIDs através do SNMP, lembrando a maneira que o DRAGON/OSCARS realiza suas configurações através de comandos CLIs.

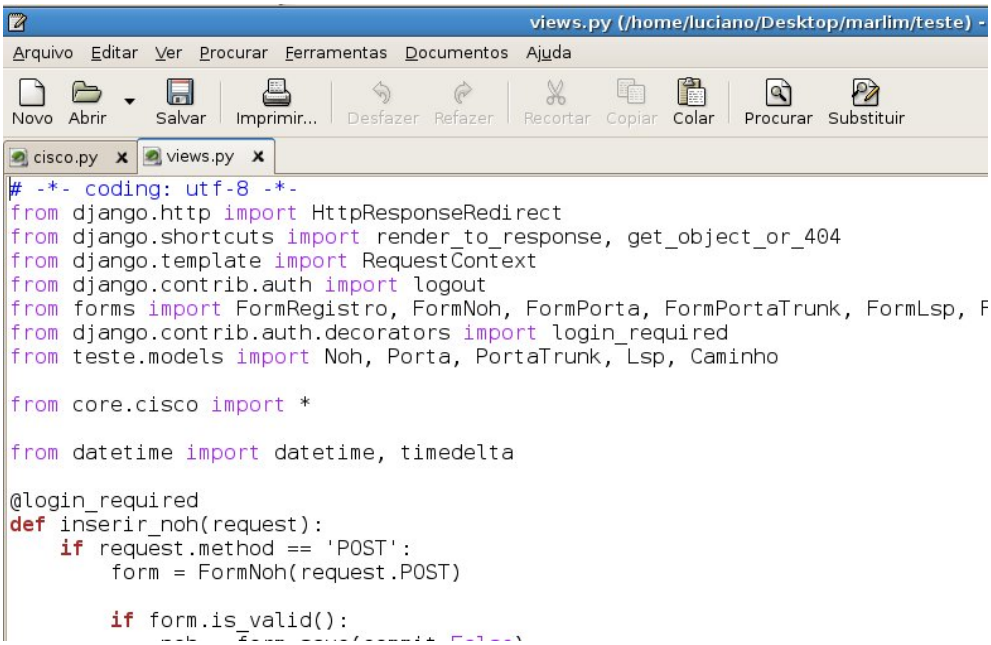

Figura 5.2: Inserção da adaptação para Cisco.

Dessa forma, obteve-se um artefato de software que possui uma boa adaptabilidade, pois assim que um novo arquivo é criado com os OIDs específicos para alguma marca

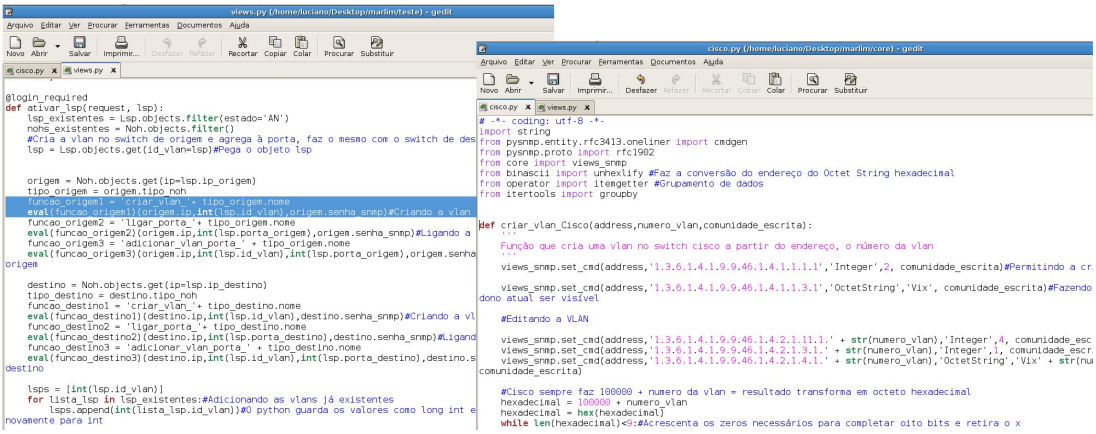

Figura 5.3: Código específico para Cisco.

de determinado fabricante é possível a sua inserção com alterações mínimas dentro do artefato. Enxerga-se essa característica na figura 5.2 em que apenas uma linha no código do protótipo *(from core.cisco import *)* deve ser adicionada para que o pacote com adaptações para o Switch Cisco seja carregado. É possível também enxergar isso na figura 5.3 onde é mostrado que a chamada de funções não se altera no código do protótipo, ou seja, o código específico com os OIDs para equipamentos Cisco fica em um arquivo a parte.

No caso da criação dos arquivos de configuração para os switches Cisco, os OIDs foram mais facilmente encontrados através da página SNMP Object Navigator, que a própria Cisco disponibiliza para agilizar o processo de entendimento e busca dos Object Identifiers. Mais informações em (26).

5.4 Funcionamento do Artefato

Para ilustrar o funcionamento do artefato e deixar mais claro as funcionalidades do mesmo, é apresentado a seguir um exemplo passo a passo das funções executadas pelo mesmo para se estabelecer um LSP.

1. O usuário, através de um browser de internet, coloca o endereço apropriado do

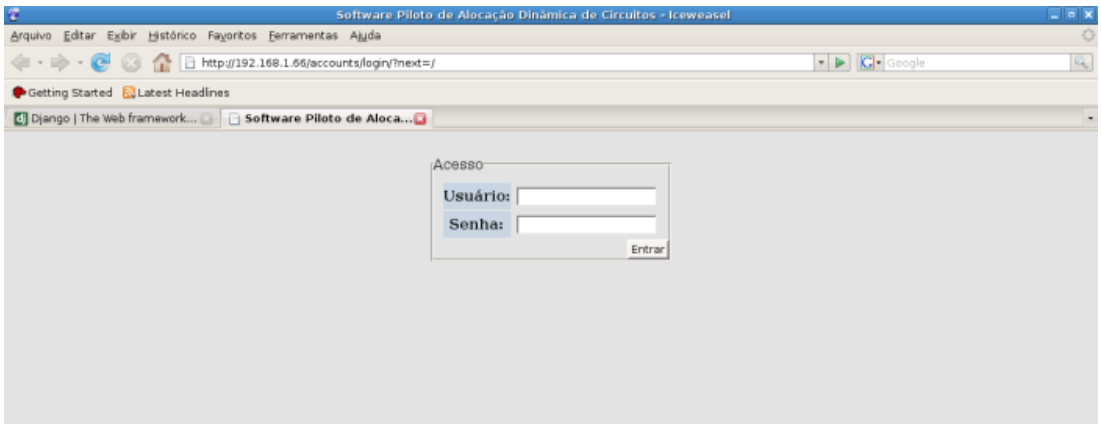

Figura 5.4: Imagem da Tela de Login do Artefato.

artefato. Nesse momento, o Módulo de Usuário é executado e, através da ajuda do Framework Django, são dadas respostas com pacotes HTTP para servirem a página de login do artefato. Fig 5.4.

2. O usuário, preenche o nome de usuário e senha. O Módulo de Usuário se utiliza de funções do Django para fazer a autenticação e autorização do perfil do usuário, e envia como resposta pacotes HTTP para servirem a página principal do artefato. Fig 5.5

3. O usuário clica no link "Inserir Nós". É ainda utilizado o Módulo de Usuário para servir as telas no browser. Fig 5.6

4. Assim que o usuário terminar de preencher os campos para inserção de nós e clicar no botão Inserir, é ativado o Módulo de Sistema que recebe, através do Módulo de Usuário, as informações dos campos preenchidos e, com a ajuda do Framework Django, interage com o Banco de Dados para a persistência dos dados. O Módulo de Usuário devolve como resposta pacotes HTTP para servirem novamente a página principal.

5. O Usuário decide então criar um LSP e clica no link "Criar LSPs". O Módulo de

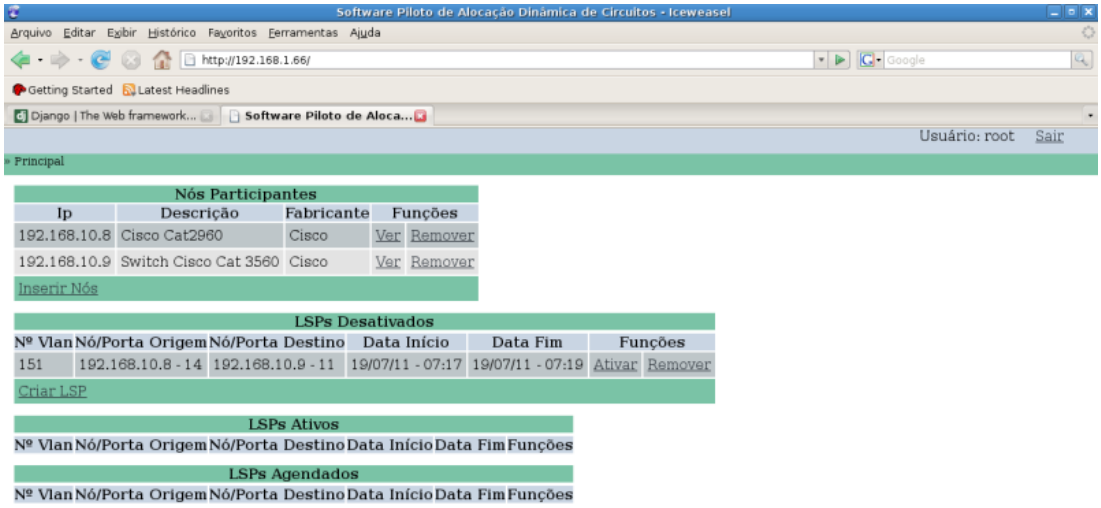

Figura 5.5: Imagem da Tela Principal do Artefato.

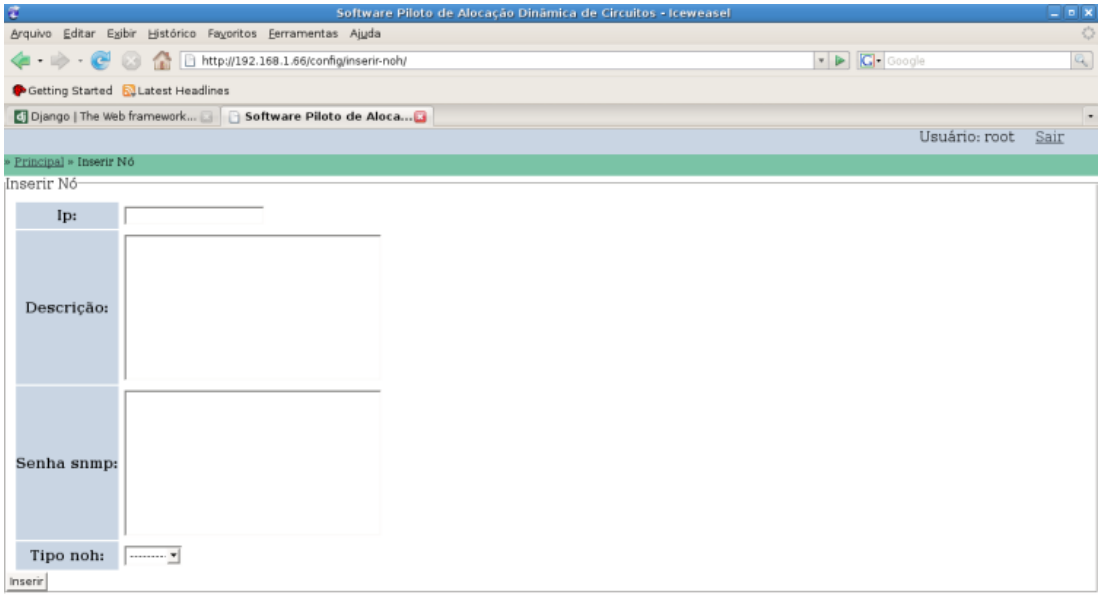

Figura 5.6: Imagem do Formulário de Inserção de Nós.

Usuário mais uma vez dá como resposta pacotes HTTP para servirem o formulário de preenchimento dos dados do LSP. Fig 5.7.

6. Assim que o usuário completa o preenchimento do formulário, o Módulo de Sistema é ativado para receber as informações, através do Módulo do Usuário, e interagir novamente com o Banco de Dados para persistir os dados. O Módulo de Usuário devolve como resposta pacotes HTTP para mais uma vez servir a página principal.

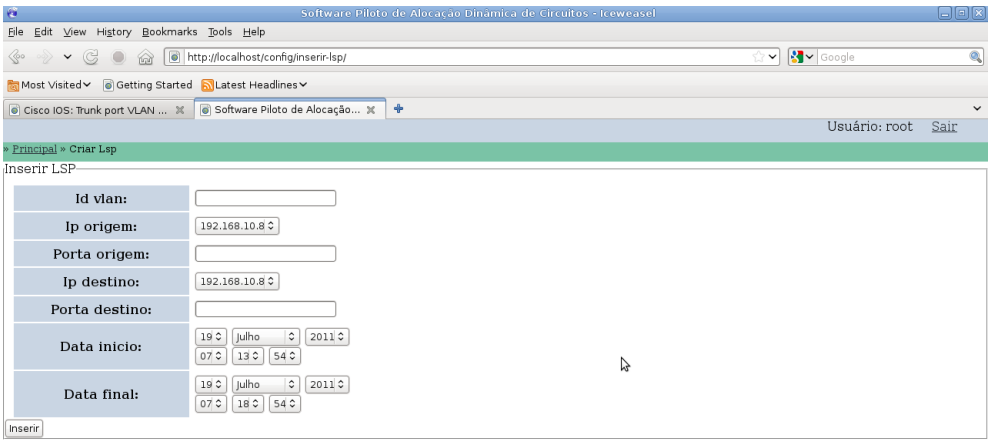

Figura 5.7: Formulário para criação do LSP.

7. O usuário clica no link “Ativar LSP”, assim o Módulo de Rede é ativado. O Módulo de Rede troca informações com o Módulo de Sistema para configurar os switches.

8. O Módulo de Rede verifica as informações inseridas no LSP e carrega o arquivo de configuração específico para o switch. Com a ajuda do PySNMP e os arquivos de OIDs específicos, o Módulo de Rede executa, nos dois switches informados, os seguintes passos: Cria a VLAN ID no 1º Switch, Insere a VLAN ID na Porta Trunk do 1º Switch, Insere a porta do 1º Switch na VLAN ID, Liga a porta no 1º Switch; Cria a VLAN ID no 2º Switch, Cria a VLAN ID no 2º Switch, Insere

a VLAN ID na Porta Trunk do 2º Switch, Insere a porta do 2º Switch na VLAN ID, Liga a porta no 2º Switch.

Ao final desses passos, é possível a comunicação através do LSP configurado nos switches da rede, pelo artefato de software.

5.5 Comparativo entre as Soluções

A seguir há uma comparação entre as características das soluções estudadas nesta dissertação e o artefato proposto.

Tabela 5.1: Tabela comparativa entre as soluções

Características	DRAGON/OSCARS	AutoBAHN	Artefato
Acesso	Browser	Browser	Browser
Código aberto	Sim	Não	Sim
Linguagem	C++/Java	Java	Python
Arquitetura	Descentralizada	Centralizada	Centralizada
Adaptável	Não	Não	Sim
Configuração	CLI	CLI	SNMP

A partir da comparação dos dados é possível perceber que algumas características do artefato proposto são derivados das soluções DRAGON/OSCARS e AutoBAHN. O principal diferencial do artefato, como pode-se ver através da tabela 5.1 é o fato dele ser adaptável (seção 5.3) e ter a capacidade de configurar os equipamentos de rede através do SNMP.

Outros pontos a serem comparados são:

Instalação

O DRAGON/OSCARS, necessita de 2 módulos separados para o seu correto funcionamento. Porém, é necessário a instalação de 3 a 4 pacotes adicionais em seu servidor Web para o funcionamento adequado.

O AutoBAHN necessita de apenas um módulo para seu funcionamento. Porém, para que funcione corretamente é necessário a instalação de um servidor Web com diversos pacotes para que a linguagem Java se adeque corretamente.

O Artefato de Software necessita de apenas um módulo. É necessário a instalação adicional de pelo menos 3 pacotes em seu servidor Web para o correto funcionamento do mesmo.

Manutenção

Foi demandado alguns meses de codificação e modificação, por parte dos pesquisadores dentro do Projeto TIAMHAT, do código praticamente inteiro do DRAGON/OSCARS, afim de adicionar uma funcionalidade que operasse um novo switch.

Não foi possível a modificação e manutenção do código do AutoBAHN pelo fato dele ser fechado.

Para adicionar um novo switch ao Artefato, é necessário a criação de um arquivo e a adição de apenas uma linha de código dentro do arquivo central de funcionamento do mesmo.

Utilização

O DRAGON/OSCARS, possui uma divisão de arquivos bastante confusa e a configuração da rede, no primeiro momento em que o domínio administrativo é configurado, é bastante trabalhosa. Seu banco de dados, durante a sua utilização, se corrompe facilmente.

O AutoBAHN possui problemas na persistência dos dados e corrompimento dos mesmos.

O Artefato possui uma utilização intuitiva e não apresenta problemas de corrompimento dos dados como os softwares citados acima, porém por se tratar de um protótipo, ainda possui algumas deficiências na validação de informações adicionadas em seu

banco.

5.6 Frameworks

A utilização dos Frameworks citados facilitou a criação do artefato e diminuiu o tempo de implementação do mesmo, assim como agregou um facilitador em sua instalação e manutenção.

5.6.1 Python

A utilização da linguagem Python para o desenvolvimento do artefato deve-se à vários fatores, entre eles: ser um software livre; possuir facilidade de busca por códigos complexos na Internet; facilidade de suporte em comunidades existentes; possuir códigos já integrados com os protocolos de rede; e ser uma linguagem utilizada pelo autor desta dissertação.

Por ser um linguagem interpretada, ela funciona, sem maiores problemas e adaptações, em qualquer arquitetura de computador e/ou sistema operacional, seja o Windows, Linux ou Mac. Além disso, para os iniciantes na linguagem, o Python possui uma curva de aprendizagem pequena e possui boa legibilidade, o que facilita o entendimento de códigos escritos por outros programadores (27).

A linguagem permite integração, sem complicações, com códigos escritos em outras linguagens tipo C/C++ e Java. Isso pode se tornar útil à medida que o artefato proposto tenha uma evolução e demande uma maior performance em determinados trechos de código. A substituição pode ser feita por linguagens de maior performance sem maiores alterações no artefato como um todo.

Pelo fato da linguagem Python ser interpretada, ela pode ser executada diretamente nos sistemas operacionais dos equipamentos de rede. Por consequência, o artefato possui a capacidade para evoluir para um código mais otimizado e adaptado e ser executado

inteiramente dentro de um switch.

5.6.2 PySNMP

O PySNMP é um framework, escrito em Python, que facilita a troca de informações através do protocolo SNMP com os equipamentos do domínio administrativo.

O PySNMP tem como função gerar os protocolos padrões do SNMP. São executadas automaticamente pelo Framework funções tais como: controle do limite do pacote de dados; autenticação e autorização; e abertura e fechamento das portas de comunicação entre as entidades SNMP. O programador somente se preocupa com os parâmetros de OID que ele deve passar dentro de uma função do próprio PySNMP (figura 5.8).

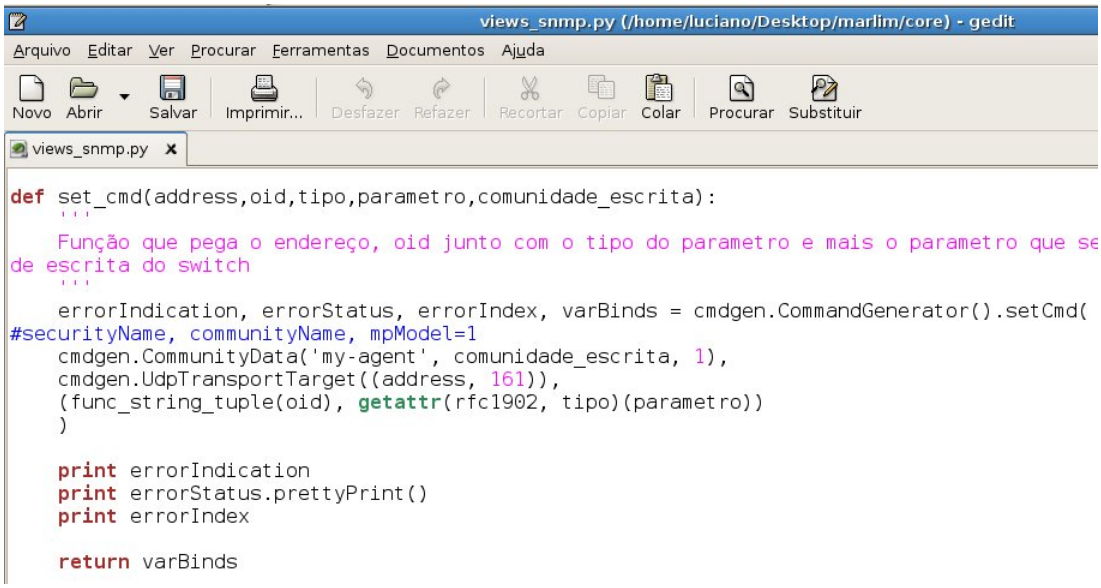

Figura 5.8: Código do Framework PySNMP.

Os comandos são feitos através de um objeto chamado *cmdgen*, é nele que são executados os comandos básicos dos protocolos seguidos dos parâmetros que o programador deseja trabalhar (figura 5.8).

O PySNMP foi desenvolvido na linguagem Python de acordo com a evolução do protocolo SNMP e por isso ele consegue trabalhar com todas as versões desenvolvidas

do protocolo. O artefato trabalha somente com o SNMPv2 por ser suficiente para executar as funções a que se destina.

Os comandos que fazem parte do artefato trabalham com a idéia do GET/SET do SNMP, o programa executa funções através do comando SET e configura os switches para que o LSP seja estabelecido.

No artefato, o PySNMP é utilizado dentro do Módulo de Rede, onde são executadas as chamadas via SNMP para a configuração dos switches.

5.6.3 Django

O Django é um framework, escrito em Python, com foco para desenvolvimento ágil de sistemas e aplicações web.

Ele possui bibliotecas e funções, escritas em Python, que são gerais para uma aplicação web, funções como: autenticação e autorização de logins; servidor de páginas; segurança, etc. O Django também é muito útil para a interação com um banco de dados. Com o uso do framework, o programador não necessita tocar nas tabelas SQL (*Structured Query Language*) para fazer a manipulação dos dados, o Django possui bibliotecas específicas para fazer esse tipo de serviço.

Assim, o programador apenas deve se preocupar com o desenvolvimento do software a que se propôs desenvolver, sem ter a preocupação de escrever funções que o próprio Django já traz embutidas. O que facilita no desenvolvimento e rapidez na construção de uma aplicação.

O Framework foi desenvolvido no padrão de arquitetura do *Model-View-Controller* (MVC) onde cada camada atua em papéis distintos sem interferir nas funções das outras. Existem pacotes de funções e arquivos específicos para se mexer em cada uma dessas camadas. A camada *Model* é responsável pelas funções relacionadas ao banco de dados, a camada *View* é responsável pelas funções que servem as páginas web e a camada *Controller* é responsável pelo controle e execução das funções criadas pelo

programador para um determinado software.

No artefato, o Django é responsável por servir as páginas web (Módulo de Usuário), fazer a interação com o banco de dados (Módulo de Sistema) e fazer as chamadas das funções que executam as funções específicas de configuração de um switch (Módulo de Sistema).

5.7 Testes

Foram utilizados, para fins de testes de funcionalidade do artefato desenvolvido, os equipamentos disponibilizados do projeto Futura RNP e TIAMHAT. Nos testes é utilizado um servidor Dell com duas máquinas virtuais, um notebook Dell onde está instalado o artefato, switch Cisco Catalyst 2900, switch Cisco Catalyst 2960 e switch Cisco Catalyst 3560E.

A instalação da rede de testes segue o diagrama da figura 5.9. O switch Catalyst 2900 é utilizado apenas para interconexão entre os equipamentos.

Os testes foram executados de acordo com os processos levantados, quando no planejamento do artefato, seguindo os procedimentos de Engenharia de Software necessários para se atingir um nível operacional satisfatório (Seção A).

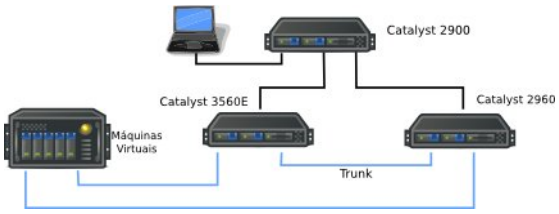

Figura 5.9: Diagrama com os equipamentos de rede.

5.7.1 Primeiro Teste

O objetivo do primeiro teste é o de verificação do processo de criação de nós e suas respectivas Portas Trunk.

Para isso são seguidos os passos:

- O Usuário se loga no sistema
- É apresentado uma tela com informações sobre Nós inseridos no sistema, LSPs Desativados, LSPs Ativos e LSPs Agendados - (Fig 5.10)
- Clica-se no link “Inserir Nós” da área de Nós
- É apresentado um formulário com campos sobre o endereço IP, Descrição e Senha SNMP do dispositivo - (Fig 5.11)
- É colocado as informações do Switch Cisco Catalyst 2960: endereço IP 192.168.10.8; descrição ’Cisco Catalyst 2960’; e senha SNMP ’private’.
- Clica-se no botão “Inserir”
- É apresentado a mesma tela de informações sobre Nós inseridos no sistema, LSPs em funcionamento, LSPs já utilizados e também LSPs agendados com o Novo Nó inserido no sistema
- Clica-se no link “Ver Nó” respectivamente ao Novo Nó inserido
- É apresentado uma tela com as informações do Nó: endereço IP 192.168.10.8; descrição ’Cisco Catalyst 2960’; e senha SNMP ’private’
- Clica-se no link “Inserir Porta Trunk”
- Insere-se o número da Porta Trunk 23
- É apresentado a mesma tela com as informações do Nó, agora com a informação adicional da Porta Trunk adicionada - (Fig 5.12)

Para verificação do Primeiro Teste é necessário se conectar via Telnet com o switch, através do mesmo endereço 192.168.10.8. Através do comando ’show running-configuration’ é possível constatar que a porta 23 possui configurações de Trunk Allowed - (Fig 5.12).

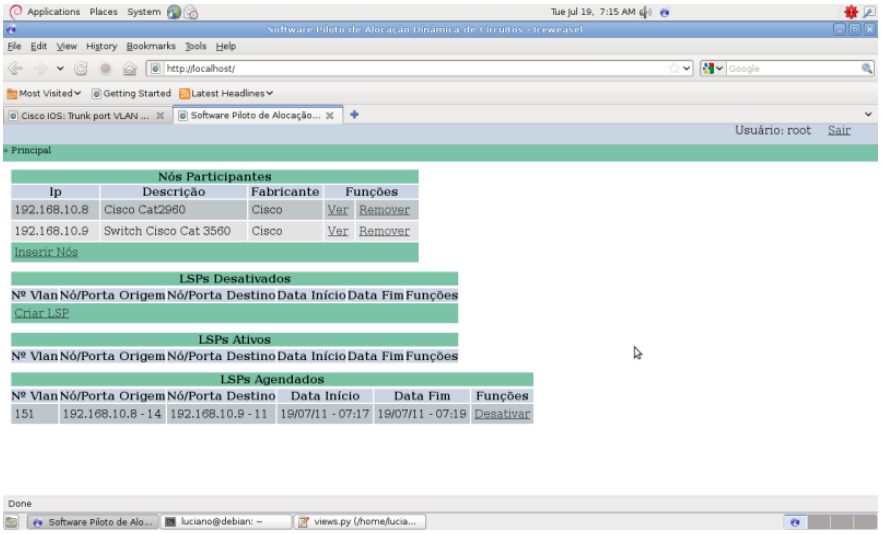

Figura 5.10: Tela Principal do Sistema e LSP Agendado.

Conclui-se que o software executa o processo de criação de nós satisfatoriamente.

5.7.2 Segundo Teste

O objetivo do segundo teste é o de verificação do processo de criação de LSPs.

Para isso são seguidos os passos:

- São inseridos no sistema os nós do: Switch Catalyst 2960 com endereço IP 192.168.10.8, Porta Trunk 23 e senha 'private'; e Switch Catalyst 3560E com endereço IP 192.168.10.9, Porta Trunk 19 e senha 'cipo'
- Os Switches são interconectados através de um cabo de rede respectivamente nas portas trunks citadas, 23 e 19
- São criados duas máquinas virtuais com endereços IP distintos dentro da mesma sub-rede: 192.168.0.20 e 192.168.0.19
- Cada máquina virtual tem sua porta conectada à um switch distinto: 192.168.0.20 é conectado ao Switch Catalyst 2960 na porta 11 e 192.168.0.19 é conectado ao Switch Catalyst 3560E na porta 14

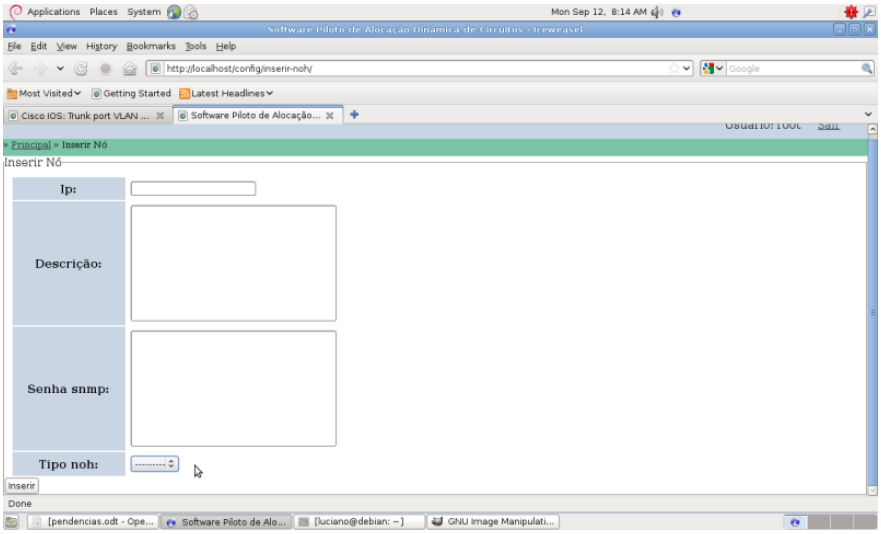

Figura 5.11: Tela com Formulário de Criação do Nó.

- Dentro de cada máquina virtual executa-se o comando Ping com o endereço da outra máquina virtual
- No sistema, clica-se no link “Criar LSP”
- É apresentado um formulário com campos sobre a ID da Vlan, Nó de Origem, Nó de Destino, Horário de Início e Fim de funcionamento do LSP - (Fig 5.13)
- É informado a ID da VLAN como 151, o nó de origem sendo o Switch Catalyst 2960 e Porta 11, o nó de destino sendo o Switch Catalyst 3560E e Porta 14, Horário de Início atual e Horário de Finalização para dois minutos à frente
- Clica-se no link “Salvar LSP”
- O Sistema apresenta a tela inicial com as informações do Novo LSP criado
- Clica-se no link “Ativar LSP”
- O Sistema apresenta a tela inicial com as informações do Novo LSP criado, acusando-o de estar ativo - (Fig 5.14)

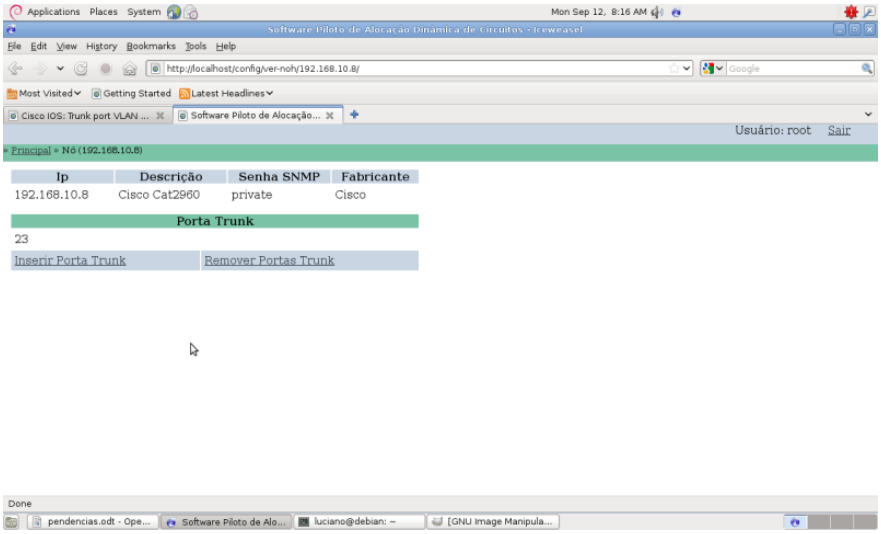

Figura 5.12: Tela com Informações do Nó Criado.

Para verificação do Segundo Teste é necessário verificar o resultado do comando Ping dentro das máquinas virtuais, que durante todo o processo de criação do LSP e ativação, é possível perceber que não há comunicação entre elas. Após certo período depois de clicado o link "Ativar LSP" nota-se que as máquinas virtuais começam a receber respostas através do comando Ping, indicando de que nesse momento elas possuem comunicação.

É possível também verificar o Segundo Teste através de conexão Telnet em um dos dois Switches, ao se conectar em um deles e executar o comando "show running-configuration" nota-se que existe uma Vlan com o mesmo número ID criado pelo sistema e que o mesmo está configurado como Vlan Allowed na porta Trunk respectivamente do Switch conectado via Telnet.

Conclui-se que o software executa o processo de criação de LSPs e permite a comunicação entre máquinas distintas satisfatoriamente.

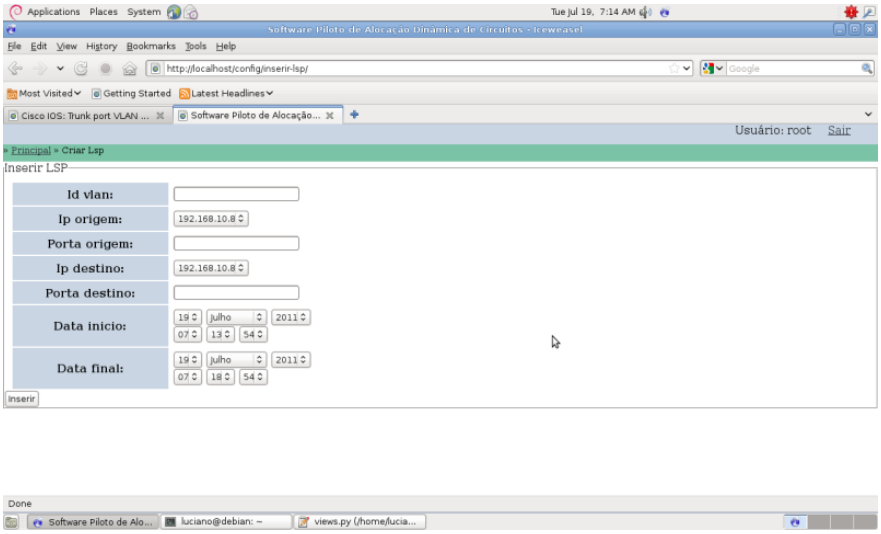

Figura 5.13: Tela com Formulário de Criação do LSP.

5.7.3 Terceiro Teste

O objetivo do terceiro teste é o de verificação dos processos de agendamento de LSPs e desativação de LSPs.

Para isso são seguidos os passos:

- São inseridos no sistema os nós do: Switch Catalyst 2960 com endereço IP 192.168.10.8, Porta Trunk 23 e senha 'private'; e Switch Catalyst 3560E com endereço IP 192.168.10.9, Porta Trunk 19 e senha 'cipo'
- Os Switches são interconectados através de um cabo de rede respectivamente nas portas trunks citadas, 23 e 19
- São criados duas máquinas virtuais com endereços IP distintos dentro da mesma sub-rede: 192.168.0.20 e 192.168.0.19
- Cada máquina virtual tem sua porta conectada à um switch distinto: 192.168.0.20 é conectado ao Switch Catalyst 2960 na porta 11 e 192.168.0.19 é conectado ao Switch Catalyst 3560E na porta 14

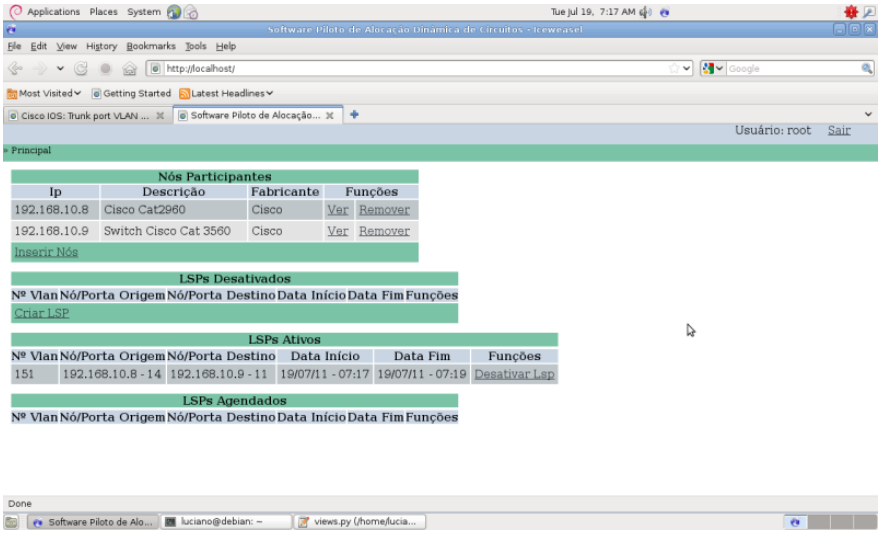

Figura 5.14: Tela com Informações do LSP Ativo.

- Dentro de cada máquina virtual executa-se o comando Ping com o endereço da outra máquina virtual
- No sistema, clica-se no link “Criar LSP”
- É apresentado um formulário com campos sobre a ID da Vlan, Nó de Origem, Nó de Destino, Horário de Início e Fim de funcionamento do LSP
- É informado a ID da VLAN como 151, o nó de origem sendo o Switch Catalyst 2960 e Porta 11, o nó de destino sendo o Switch Catalyst 3560E e Porta 14, Horário de Início para cinco minutos à frente e Horário de Finalização para dez minutos à frente
- Clica-se no link “Salvar LSP”
- O Sistema apresenta a tela inicial com as informações do Novo LSP criado
- Clica-se no link “Ativar LSP”
- O Sistema apresenta a tela inicial com as informações do Novo LSP criado, acusando-o de estar agendado - (Fig 5.10)

- Após os cinco minutos de acordo com o Horário de Início para o LSP, o sistema apresenta a tela inicial com as informações do Novo LSP criado, acusando-o de estar em funcionamento - (Fig 5.14)

- Após os dez minutos de acordo com o Horário de Término do LSP, o sistema apresenta a tela inicial com as informações do Novo LSP criado, acusando-o de estar desativado - (Fig 5.15)

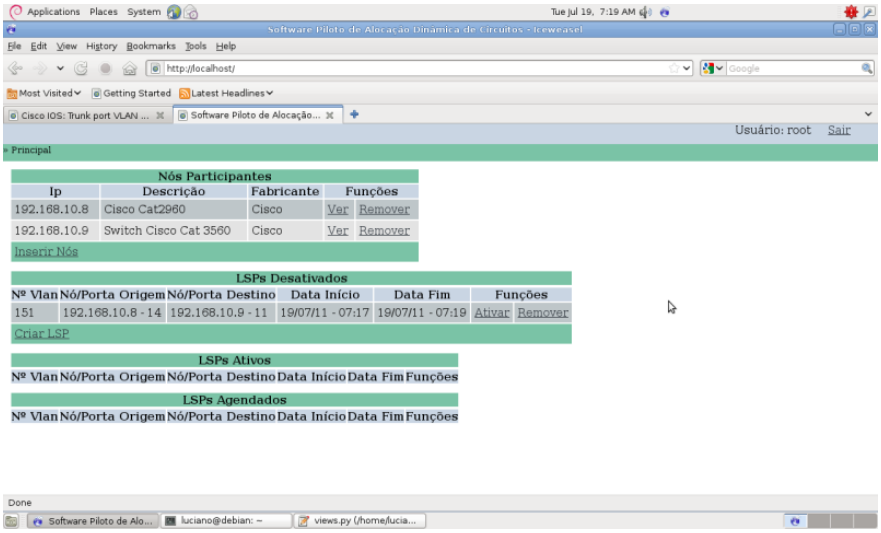

Figura 5.15: Tela com Informações do LSP Desativado.

Para verificação do Terceiro Teste é necessário verificar o resultado do comando Ping dentro das máquinas virtuais, que durante todo o processo de criação do LSP, agendamento e término, é possível perceber que não existe comunicação entre elas. Após o período de agendamento configurado para início do LSP, nota-se que as máquinas virtuais começam a receber respostas através do comando Ping, indicando que nesse momento elas possuem comunicação. Quando o LSP é acusado pelo sistema de já estar desativado, nota-se que a comunicação entre as máquinas virtuais cessa.

É possível também verificar o Terceiro Teste através de conexão Telnet em um dos dois Switches, ao se conectar em um deles e executar o comando “show running-

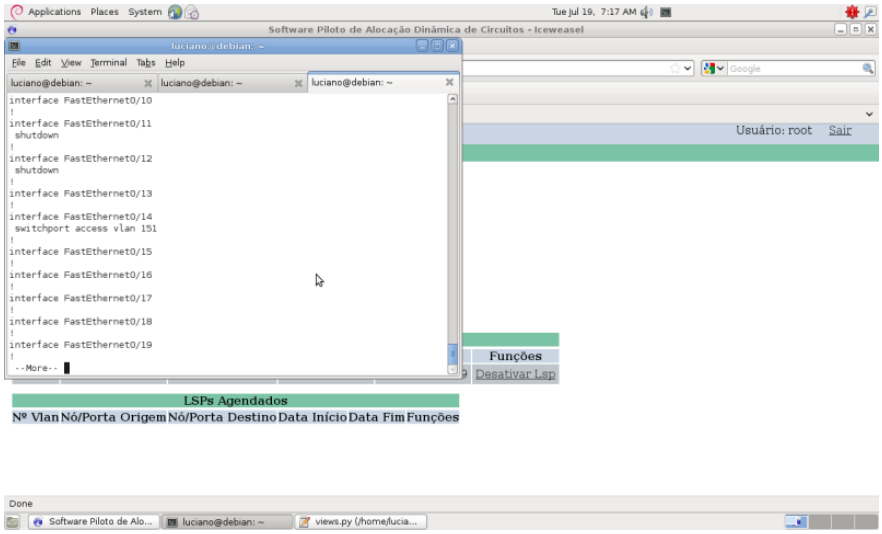

Figura 5.16: Tela com Configuração do Switch.

configuration" nota-se que, durante o tempo em que o LSP é acusado de estar ativo, existe uma Vlan com o mesmo número ID criado pelo sistema e que o mesmo está configurado como Vlan Allowed na porta Trunk respectivamente do Switch conectado via Telnet - (Fig. 5.16 e 5.17).

Conclui-se que o software executa os processos de agendamento e desativação de LSPs satisfatoriamente.

5.8 Conclusão

Neste capítulo é possível entender como o artefato de software foi desenvolvido e também as tecnologias que facilitaram a criação do mesmo. É possível entender a facilidade adicionada de abstração dos códigos específicos para switches, o que gera uma grande adaptabilidade de uso do artefato em diversos equipamentos e outros que poderão ser desenvolvidos no futuro. É possível estabelecer um comparativo entre as soluções estudadas e o artefato proposto, assim como perceber que os objetivos desta dissertação em relação ao artefato foram atingidos.

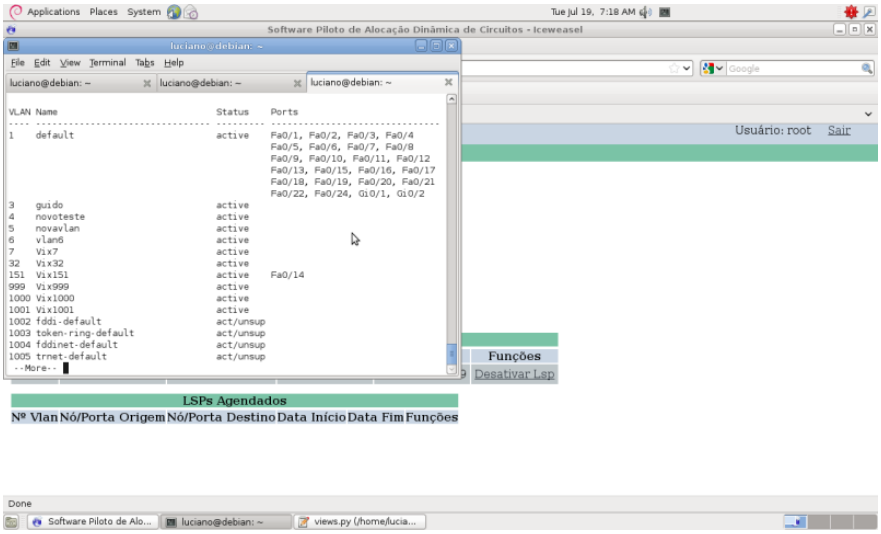

Figura 5.17: Tela com Configuração do Switch.

Também é possível constatar que os testes apresentados verificam que o artefato é capaz de executar satisfatoriamente os processos necessários levantados no momento de planejamento do mesmo, o que se conclui de que o artefato é operacional.

Capítulo 6

Conclusões e Sugestões de Trabalhos Futuros

6.1 Conclusão

Há uma demanda crescente por redes que possam prover à comunidade científica uma capacidade de uma alta taxa de transferência de dados por um curto período de tempo através de aplicações chamadas *e-science*, ao mesmo tempo em que a mesma infraestrutura consegue manter uma rede de comutação por pacotes como é hoje a *internet*. Porém, essa infraestrutura torna-se inviável pelo fato da necessidade da troca de praticamente todos os equipamentos para prover essas novas funcionalidades.

Por isso, novas iniciativas e soluções desenvolvidas em software surgiram ao redor do mundo. Esses softwares vêm sendo muito pesquisados, pelo fato de proverem as mesmas funcionalidades das redes *e-science* utilizando-se da mesma infraestrutura de rede em uso atualmente. Prover um plano de controle capaz de alocar dinamicamente os recursos tornou-se meta de pesquisadores de rede ao redor do mundo. Algumas iniciativas mais concretas podem ser vistas, como no caso do DRAGON/OSCARS e AutoBAHN, explicadas nesta dissertação.

Porém, é possível perceber que estas iniciativas ainda devem percorrer um longo caminho de desenvolvimento para que se tornem robustas o suficiente para serem utilizadas em redes de produção, dado a alta complexidade de interação entre os protocolos envolvidos para que o plano de controle permita a Alocação Dinâmica de Recursos. Os protocolos utilizados nestas soluções foram explicados sucintamente nesta dissertação.

Foi possível perceber também algumas deficiências e dificuldades em alguns pontos em relação ao DRAGON/OSCARS e AutoBAHN. Para suprir deficiências quanto à instalação, utilização e manutenção em comparação às soluções, é proposto o artefato de software. Uma grande contribuição que o artefato trouxe, é a melhoria da adaptabilidade quanto aos equipamentos de rede de diferentes fabricantes, bastando para isso a busca pelos OID específicos e a criação de arquivos para a configuração dos mesmos e a rápida inserção dentro do artefato já em uso.

O artefato possui também um ponto positivo em sua capacidade de configurar os equipamentos de rede através do protocolo SNMP, ação que nenhuma das outras soluções executa. Outros pontos interessantes são o ambiente de desenvolvimento ágil, através de frameworks, e a capacidade de prover uma interface mais amigável com o usuário.

Através de vários testes executados e explicados nesta dissertação, é possível concluir que o artefato proposto é operacional e é capaz de estabelecer LSPs entre equipamentos distintos sem maiores dificuldades.

Conclui-se então que o objetivo proposto nesta dissertação, que foi o de desenvolver um artefato com melhorias em relação às soluções estudadas, foi alcançado.

6.2 Trabalhos Futuros

Acredita-se que o artefato possa ser utilizado como base para soluções futuras de Alocação Dinâmica de Recursos, bastando que a comunidade o adote e desenvolva módulos

de interação com protocolos mais complexos como o OSPF-TE e RSVP-TE. Assim, o artefato poderá se desenvolver em mais funcionalidades provendo serviços que garantam a Qualidade de Serviço, descrição de topologias, descoberta de novos recursos, etc (Fig. 6.1).

O artefato, do modo como foi desenvolvido, é capaz de integrar facilmente outros módulos que no futuro, possam ser desenvolvidos. Assim, módulos a parte podem se somar às funcionalidades já existentes. Um módulo que provavelmente agrega valor ao artefato é o AAA - Módulo de autorização, autenticação e contabilidade de serviços oferecidos ao usuário, por exemplo.

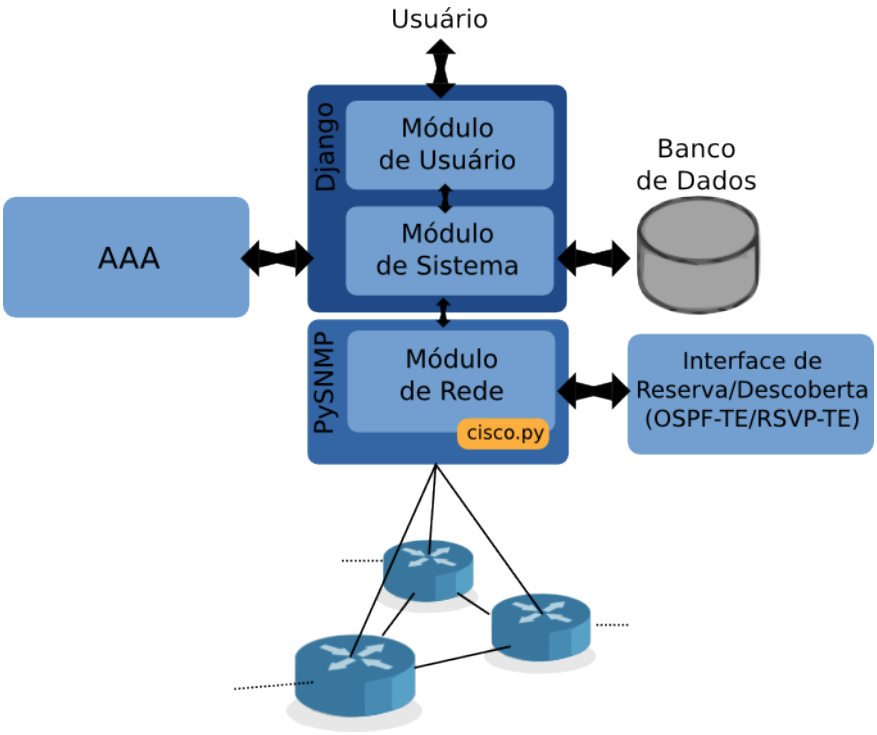

Figura 6.1: Diagrama com novos módulos.

Por outro lado, o mesmo artefato pode servir como base para o aprimoramento dos softwares já existentes. Basta que haja um aprimoramento nos códigos das soluções estudadas. Adotando-se a mesma estratégia de desenvolvimento do artefato em mó-

dulos e trazendo a mesma funcionalidade de adaptabilidade, através de arquivos de configuração dos equipamentos de rede, via SNMP.

Desse modo, as dificuldades nos softwares DRAGON/OSCARS e AutoBAHN, quanto a instalação, manutenção e utilização, enfrentadas durante as pesquisas do Projeto TI-AMHAT, podem ser solucionadas.

Referências Bibliográficas

[1] A. Jukan and G. Karmous-Edwards, "Optical Control Plane for the Grid Community," *Communications Surveys and Tutorials, IEEE*, vol. 9, no. 3, pp. 30–44, Third Quarter 2007.

[2] I. Habib, Q. Song, Z. Li, and N. Rao, "Deployment of the GMPLS control plane for grid applications in experimental high-performance networks," *Communications Surveys and Tutorials, IEEE*, vol. 44, no. 3, pp. 65–73, March 2006.

[3] A. Farrel and I. Bryskin, *GMPLS: Architecture and Applications (The Morgan Kaufmann Series in Networking)*. Morgan Kaufmann, 2005.

[4] RNP, "Wiki Projeto Futura RNP." http://wiki.rnp.br/display/futura.

[5] "Dragon project." http://dragon.maxgigapop.net/.

[6] "Autobahn." http://www.geant2.net/server/show/nav.756.

[7] J. Khan and O. Y. Tahboub, "DCN@MPLS: A Network Architectural Model for Dynamic Circuit Networking at Multiple Protocol Label Switching," *2009 Ninth Annual International Symposium on Applications and the Internet*, 2009.

[8] T. e. a. Lehman, "Control Plane Architecture and Design Considerations for Multi-Service, Multi-Layer, Multi-Domain Hybrid Networks," *2007 IEEE Infocom*, 2007.

[9] J. F. Szegedi, Peter Riera and J. A. García-Espín, "Enabling Future Internet Research: The FEDERICA Case," *2011 IEEE Communications Magazine*, July 2011.

[10] P. Angu and B. Ramamurthy, "Experiences with Dynamic Circuit Creation in a Regional Network Testbed," *2011 IEEE Infocom*, 2011.

[11] C. P. G. and, "A User Driven Dynamic Circuit Network Implementation,"

[12] "RFC 3473 - Generalized Multi-Protocol Label Switching (GMPLS) Si." http://www.faqs.org/rfcs/rfc3473.html.

[13] "Internet2." https://wiki.internet2.edu/confluence/display/abti2/Home.

[14] "IDC-Message-draft." https://wiki.internet2.edu/confluence/download/attachments/19074/IDC-Messaging-draft.pdf?version=1&modificationDate=1216238419210.

[15] "NARB RCE Architecture v.2.1b," April 2008.

[16] "ESNet." http://www.es.net/.

[17] "OSCARS." https://oscars.es.net/OSCARS.

[18] "GÉANT2." http://www.geant2.net/.

[19] "RFC 2205 - Resource ReSerVation Protocol (RSVP) – Version 1 Fun." http://www.faqs.org/rfcs/rfc2205.html.

[20] "RFC 3209 - RSVP-TE: Extensions to RSVP for LSP Tunnels." http://www.faqs.org/rfcs/rfc3209.html.

[21] "RFC 2328 - OSPF Version 2." http://www.faqs.org/rfcs/rfc2328.html.

[22] D. N. Blank-Edelman, "The 20-Minute SNMP Tutorial - Automating System Administration with Perl." http://oreilly.com/perl/excerpts/system-admin-with-perl/twenty-minute-snmp-tutorial.html, 2009.

[23] "Snmp library for python." http://pysnmp.sourceforge.net/docs/4.x/index.html.

[24] O'Reilly, "The Twenty-Minute SNMP Tutorial." http://docstore.mik.ua/orelly/perl/sysadmin/appe_01.htm.

[25] N. Technologies. http://www.ndt-inc.com/SNMP/HelpFiles/v3ConfigTutorial/v3ConfigTutorial.html.

[26] Cisco, "Cisco SNMP Object Navigator." http://tools.cisco.com/Support/SNMP/do/BrowseOID.do.

[27] M. Brandão, "Aprendendo Django no Planeta Terra." http://www.aprendendodjango.com/.

[28] S. L. Pfleeger, *Engenharia de Software: Teoria e Prática*. Prentice Hall, 2004.

Apêndice A

Desenvolvimento do Artefato de Software

Com o intuito de se atingir uma qualidade mínima na produção do artefato de software, mesmo considerando-se ainda um protótipo, buscou-se seguir as bases da Engenharia de Software. As descrições formais do artefato são descritas a seguir com base em (28).

É chamado de Nó, um equipamento com endereço IP (switch) que o sistema tem a capacidade de administrar e executar funções de configurações. É chamado de LSP um caminho configurado por uma vlan, que atravessa switches administrados pelo sistema.

A.0.1 Requisitos do Software

O artefato deve seguir as orientações abaixo:

- Facilitar a adaptação para novos equipamentos (switches)
- Criar LSP entre Switches distintos
- Desabilitar/Habilitar LSPs de acordo com agendamento

A.0.2 Diagrama de Caso de Uso

O usuário deve acessar o sistema através de uma conta autorizada e autenticada pelo sistema. Quando autorizado, o usuário tem acesso a uma tela com uma lista com informações de nós criados e administrados pelo sistema. O usuário, quando autorizado, tem permissão para: criar e remover nós, criar e remover Portas Trunk, criar e remover LSPs, ativar e desativar LSPs e agendar ou retirar LSPs do agendamento (Figura A.1). O sistema deve automaticamente configurar ou desconfigurar LSPs de acordo com o horário do agendamento.

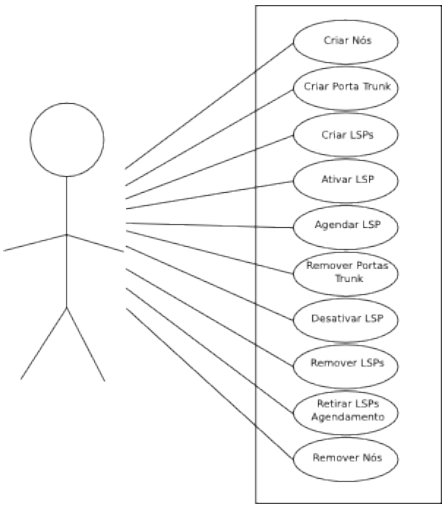

Figura A.1: Diagrama de Caso de Uso.

A.0.3 Casos de Uso

Caso: Criar Nós - C01

Ator: Usuário

Descrição: Esse caso de uso serve para descrever como são inseridos os Nós

Fluxo Principal:

1. O Usuário se loga no sistema

2. O Sistema apresenta uma tela com informações sobre Nós inseridos no sistema, LSPs Desativados, LSPs Ativos e LSPs Agendados

3. O Usuário clica no link “Inserir Nós” da área de Nós

4. O Sistema apresenta um formulário com campos sobre o endereço IP, Descrição e Senha SNMP do dispositivo

5. O Usuário preenche os campos adequadamente e clica no botão “Inserir”

6. O Sistema apresenta uma tela com informações sobre Nós inseridos no sistema, LSPs Desativados, LSPs Ativos e LSPs Agendados

Caso: Criar Porta Trunk - C02

Descrição: Esse caso de uso serve para descrever como são inseridas as Portas Trunks nos Nós

Ator: Usuário

Pré Condição: É necessário que o sistema possua pelo menos um Nó criado através da ação C01 - Criação de Nós

Fluxo Principal:

1. O Usuário se loga no sistema

2. O Sistema apresenta uma tela com informações sobre Nós inseridos no sistema, LSPs Desativados, LSPs Ativos e LSPs Agendados

3. O Usuário clica no link “Ver Nó” de algum Nó inserido no sistema

4. O Sistema apresenta uma tela com informações sobre o endereço IP, a descrição, a senha SNMP e as portas Trunks já inseridas no Nó em questão

5. O Usuário clica no link “Inserir Porta Trunk”

6. O Sistema apresenta um formulário com um campo para preenchimento do número da porta Trunk

7. O Usuário preenche o campo adequadamente e clica no botão “Inserir”

8. O Sistema apresenta uma tela com informações sobre o endereço IP, a descrição, a senha SNMP e as portas Trunks já inseridas no Nó em questão

Caso: Criar LSP - C03

Descrição: Esse caso de uso serve para descrever como é criado um LSP

Ator: Usuário

Pré Condição: É necessário que o Usuário tenha efetuado os casos C01 e C02 - Criação de Nós e Criação de Porta Trunk

Fluxo Principal:

1. O Usuário se loga no sistema

2. O Sistema apresenta uma tela com informações sobre Nós inseridos no sistema, LSPs Desativados, LSPs Ativos e LSPs Agendados

3. O Usuário clica no link “Criar LSP”

4. O Sistema apresenta um formulário com campos sobre a ID da Vlan, Nó de Origem, Nó de Destino, Horário de Início e Fim de funcionamento do LSP

5. O Usuário preenche adequadamente os campos, deixando para que o Horário de Início do LSP seja o atual e clica no botão “Salvar LSP”(A1)

6. O Sistema apresenta uma tela com informações sobre Nós inseridos no sistema, LSPs Desativados, LSPs Ativos e LSPs Agendados

Fluxo Alternativo:

A1:[Criando LSP com Horário de Início adiantado]

1. O Usuário preenche adequadamente os campos, deixando para que o Horário de Início do LSP seja adiantado em relação ao horário atual e clica no botão “Salvar LSP”

2. O Sistema apresenta uma tela com informações sobre Nós inseridos no sistema, LSPs Desativados, LSPs Ativos e LSPs Agendados

Caso: Ativar LSP/Agendar LSP - C04

Descrição: Esse caso de uso serve para descrever como é ativado um LSP

Ator: Usuário

Pré Condição: É necessário que o Usuário tenha efetuado o caso C03 - Criar LSP

Fluxo Principal:

1. O Sistema apresenta uma tela com informações sobre Nós inseridos no sistema, LSPs Desativados, LSPs Ativos e LSPs Agendados

2. Na área de LSPs criados, o Usuário clica no link “Ativar LSP”

3. O Sistema apresenta uma tela com informações sobre Nós inseridos no sistema, LSPs Desativados, LSPs Ativos e LSPs Agendados

4. O Sistema apresenta o LSP criado na área de LSPs Ativos (A1)

Fluxo Alternativo:

A1:[LSP Agendado]

1. O Sistema apresenta o LSP criado na área de LSPs Agendados (A1)

Caso: Remover Portas Trunk - C05

Descrição: Esse caso de uso serve para descrever como são removidas as Portas Trunks

Pré Condição: É necessário que o Usuário tenha efetuado o caso C02 - Criar Porta Trunk

Fluxo Principal:

1. O Sistema apresenta uma tela com informações sobre o endereço IP, a descrição, a senha SNMP e as portas Trunks já inseridas no Nó em questão

2. O Usuário clica no link "Remover Portas Trunks"

3. O Sistema apresenta uma tela com informações sobre o endereço IP, a descrição, a senha SNMP e nenhuma Porta Trunk inserida no Nó em questão

Caso: Desativar LSP - C06

Descrição: Esse caso de uso serve para descrever como desativar o LSP

Ator: Usuário

Pré Condição: É necessário que o Usuário tenha efetuado o caso C04 - Ativar LSP

Fluxo Principal:

1. O Sistema apresenta uma tela com informações sobre Nós inseridos no sistema, LSPs Desativados, LSPs Ativos e LSPs Agendados

2. O Sistema apresenta o LSP criado na área de LSPs Ativos

3. O Usuário clica no link "Desativar LSP" (A1)

4. O Sistema apresenta uma tela com informações sobre Nós inseridos no sistema, LSPs Desativados, LSPs Ativos e LSPs Agendados

5. O Sistema apresenta o LSP desativado na área de LSPs Desativados

Fluxo Alternativo:

A1:[LSP Desativado pelo Sistema]

1. O Sistema analisa que o LSP está com seu horário terminado

2. O Sistema apresenta uma tela com informações sobre Nós inseridos no sistema, LSPs Desativados, LSPs Ativos e LSPs Agendados

3. O Sistema apresenta o LSP desativado na área de LSPs Desativados

Caso: Retirar LSP do Agendamento - C07

Descrição: Esse caso de uso serve para descrever como retirar o LSP do agendamento

Ator: Usuário

Pré Condição: É necessário que o Usuário tenha efetuado o caso C04 - Agendar LSP

Fluxo Principal:

1. O Sistema apresenta uma tela com informações sobre Nós inseridos no sistema, LSPs Desativados, LSPs Ativos e LSPs Agendados

2. O Sistema apresenta o LSP criado na área de LSPs Agendados

3. O Usuário clica no link “Desativar”

4. O Sistema apresenta uma tela com informações sobre Nós inseridos no sistema, LSPs Desativados, LSPs Ativos e LSPs Agendados

5. O Sistema apresenta o LSP retirado na área de LSPs Desativados

Caso: Remover Nós - C08

Descrição: Esse caso de uso serve para descrever como remover um Nó

Ator: Usuário

Fluxo Principal:

1. O Sistema apresenta uma tela com informações sobre Nós inseridos no sistema, LSPs Desativados, LSPs Ativos e LSPs Agendados

2. O Usuário clica no link de “Remover Nó“

3. O Sistema apresenta uma tela com informações sobre Nós inseridos no sistema sem a informação do Nó removido, LSPs Desativados, LSPs Ativos e LSPs Agendados

Caso: Remover LSPs - C09

Descrição: Esse Caso de uso serve para descrever como remover um LSP

Ator: Usuário

Pré Condição: É necessário que o Usuário tenha efetuado o caso C03 - Criar LSP

Fluxo Principal:

1. O Sistema apresenta uma tela com informações sobre Nós inseridos no sistema, LSPs Desativados, LSPs Ativos e LSPs Agendados

2. O Usuário clica no link ”Remover“ na área de LSPs Desativados

3. O Sistema apresenta uma tela com informações sobre Nós inseridos no sistema, LSPs Desativados sem a informação do LSP removido, LSPs Ativos e LSPs Agendados

A.1 Eventos

Os eventos que o software deve satisfatoriamente executar são os seguintes:

- Criar Nós - (Figura A.2)
- Criar Porta Trunk - (Figura A.3)
- Criar LSP/Ativar LSP/Agendar LSP - (Figura A.4)
- Remover Portas Trunk - (Figura A.5)
- Desativar LSP - (Figura A.6)

- Retirar LSP do Agendamento - (Figura A.7)
- Remover Nós - (Figura A.8)
- Remover LSPs - (Figura A.9)

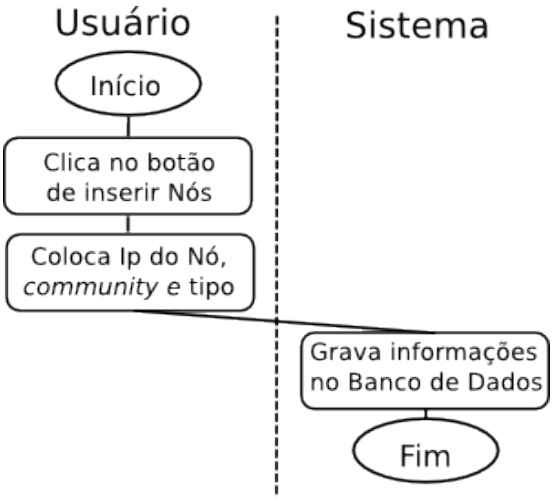

Figura A.2: Fluxograma de criação de Nós.

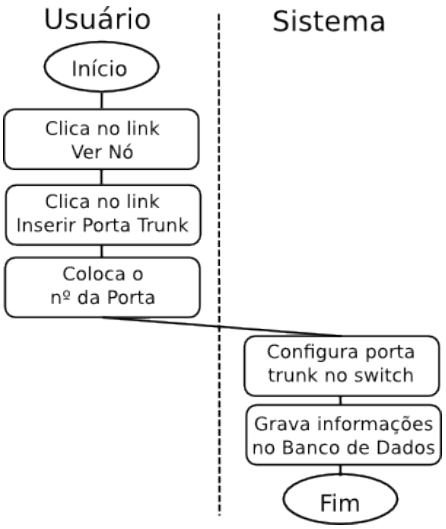

Figura A.3: Fluxograma de criação de Porta Trunk.

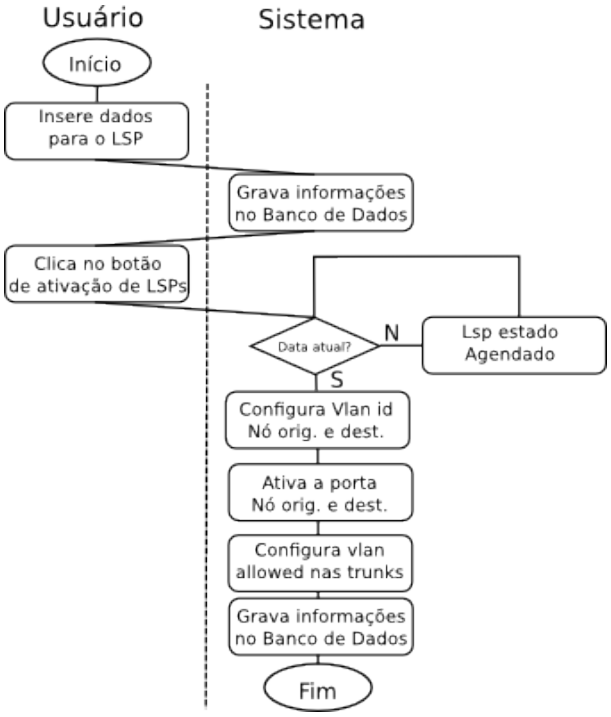

Figura A.4: Fluxograma de criação e agendamento de LSPs.

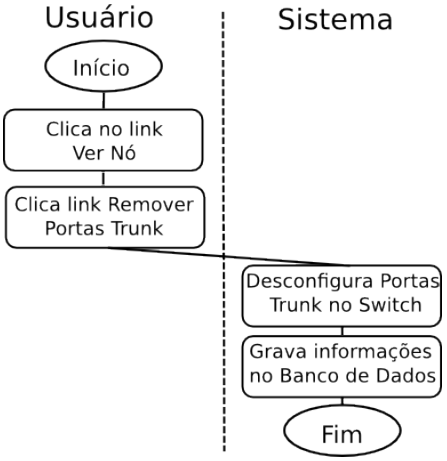

Figura A.5: Fluxograma de remoção de Portas Trunk.

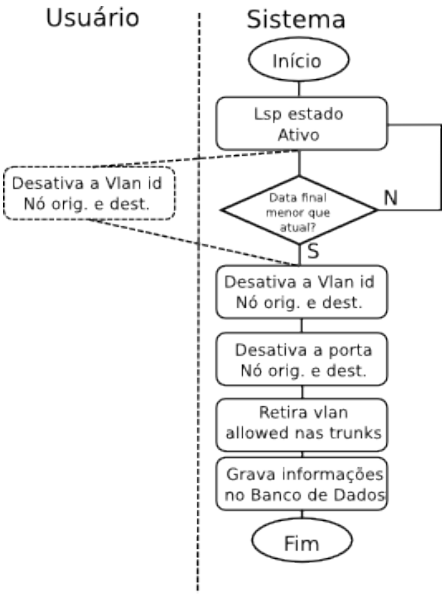

Figura A.6: Fluxograma de Desativação de LSPs.

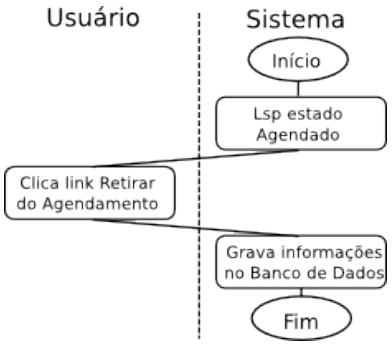

Figura A.7: Fluxograma de Desativação de Agendamento.

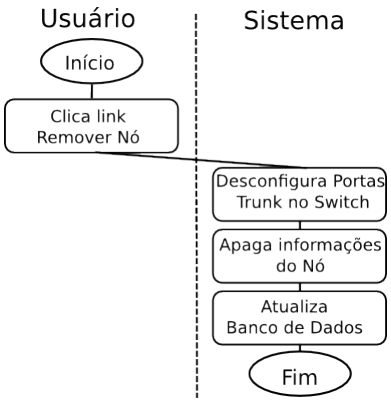

Figura A.8: Fluxograma de Remoção de Nó.

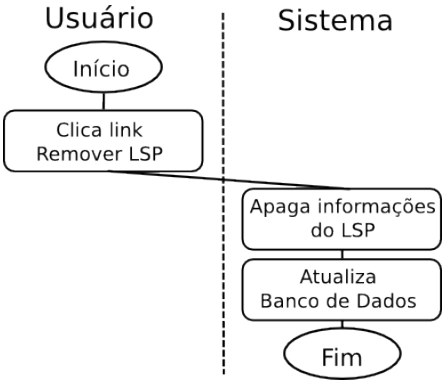

Figura A.9: Fluxograma de Remoção de LSPs.

A.2 Diagrama UML

De acordo com (28) foi gerado através da Linguagem UML - *Unified Modeling Language* o diagrama de classes com seus atributos e funções de acordo com a figura A.10.

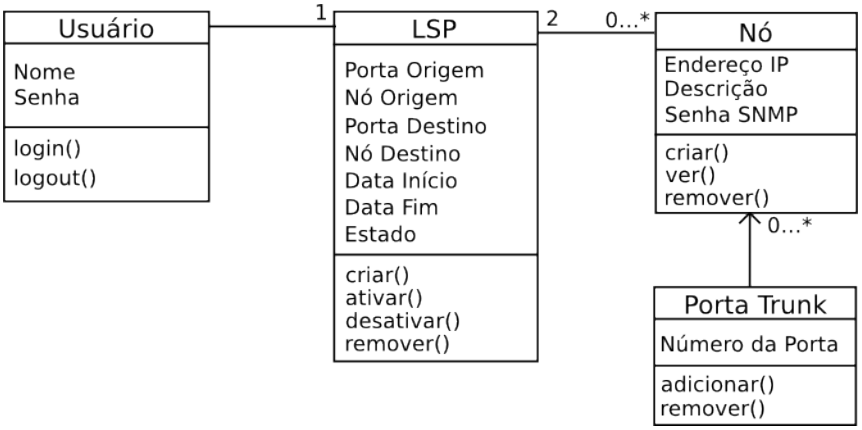

Figura A.10: Diagrama UML do Artefato.

O usuário, através de um programa browser, tem total controle do sistema, podendo inserir e retirar nós do artefato.

Para que seja estabelecido um LSP é necessário primeiro, que o usuário cadastre no sistema os switches (nós) que o artefato irá controlar, feito isso, é necessário também criar as portas trunks por onde os switches se comunicam (Figura A.3).

Com os switches e suas portas trunks devidamente inseridos no sistema, para se estabelecer um LSP, basta criar um nó com sua porta de origem e um nó com sua porta de destino que o artefato irá configurar os switches automaticamente e estabelecerá um LSP (Figura A.4).

Apêndice B

Manual de Instalação

B.1 Pacotes

O artefato foi desenvolvido inteiramente no Linux (Debian 4.0) e funciona com o suporte dos pacotes: python, apache, mod_python, mysql e cron. Esses pacotes foram instalados através do programa apt-get, devido à sua facilidade de instalação são abordadas as configurações dos pacotes que precisaram de algumas informações adicionais para funcionarem.

O pacote Django (versão 1.3) possui um excelente manual de instalação e configuração da base de dados (em inglês) e por isso também não será abordado. Seu manual pode ser encontrado em http://www.djangoproject.com. O mesmo ocorre com o manual de instalação do PySNMP cujo manual (em inglês) pode ser encontrado no endereço http://pysnmp.sourceforge.net/.

B.2 Configuração do Artefato

B.2.1 Apache

Após baixar o pacote mod_python e depois de configurados os pacotes do Django e PySNMP, é necessário configurar corretamente o arquivo do servidor web Apache. Este arquivo encontra-se em /etc/apache2/httpd.conf (Linux Debian).

```
 <Location "/">
    SetHandler python-program
    PythonHandler django.core.handlers.modpython
    SetEnv DJANGO_SETTINGS_MODULE settings
    #PythonOption django.root /
    PythonDebug On
    PythonPath "['/ENDEREÇO/DO/ARTEFATO'] + sys.path"
</Location>

Alias /admin_media /ENDEREÇO/DO/DJANGO-ACRESCIDO-DE/django/contrib/admin/media
<Directory "/ENDEREÇO/DO/DJANGO-ACRESCIDO-DE/django/contrib/admin/media">
    Order allow,deny
    Allow from all
</Directory>

<Location "/admin_media">
    SetHandler None
</Location>


Alias /media /ENDEREÇO-DO-ARTEFATO-ACRESCIDO-DE/media
```

```
<Directory "/ENDEREÇO-DO-ARTEFATO-ACRESCIDO-DE/media">
    Order allow,deny
    Allow from all
</Directory>

<Location "/media">
    SetHandler None
</Location>
```

Esses comandos devem ser escritos nesse arquivo trocando os nomes em maiúsculo para os respectivos endereços adequados.

B.2.2 Cron

O programa Cron é utilizado para que o sistema execute corretamente o agendamento de LSPs, ele é facilmente instalado e configurado através do programa apt-get. Após a sua instalação, no endereço onde o artefato foi instalado, é necessário aplicar o seguinte comando:

```
sudo crontab script.cron
```

B.3 Adaptação para novos equipamentos

Como visto nesta dissertação, o artefato criado propõe uma maneira mais simples de se adaptar códigos específicos dos equipamentos de diversos fabricantes. Para se fazer isso é necessário criar no mesmo diretório '../marlim/core' (a partir de onde o artefato está instalado) em nome minúsculo o arquivo com o nome do fabricante. Esse arquivo deve possuir extensão '.py' e deve necessariamente conter funções estruturadas de maneira que funcionem através do framework PySNMP.

O arquivo deve necessariamente possuir as seguintes funções gerais: 'criar_vlan_', 'deletar_vlan_', 'criar_interface_mode_trunk_', 'retirar_interface_mode_trunk_'. Essas são as funções gerais que o artefato se utiliza para configurar os equipamentos. Sendo que mais funções para tratar as especificidades do equipamento podem ser adicionadas sem nenhum problema no arquivo.

Atentar que necessariamente as funções gerais devem ter seu último nome com o nome do Fabricante. Exemplo é a função geral 'criar_vlan_', no arquivo cisco.py, fica como 'criar_vlan_Cisco'.

Para que o sistema possa acessar o novo arquivo é necessário a inclusão no arquivo ../marlim/teste/views.py do comando (exemplo: figura B.1):

```
from core.nome_do_arquivo import *
```

O nome_do_arquivo deve ser trocado pelo nome exato do nome do arquivo incluído no diretório retirando-se a extensão '.py'.

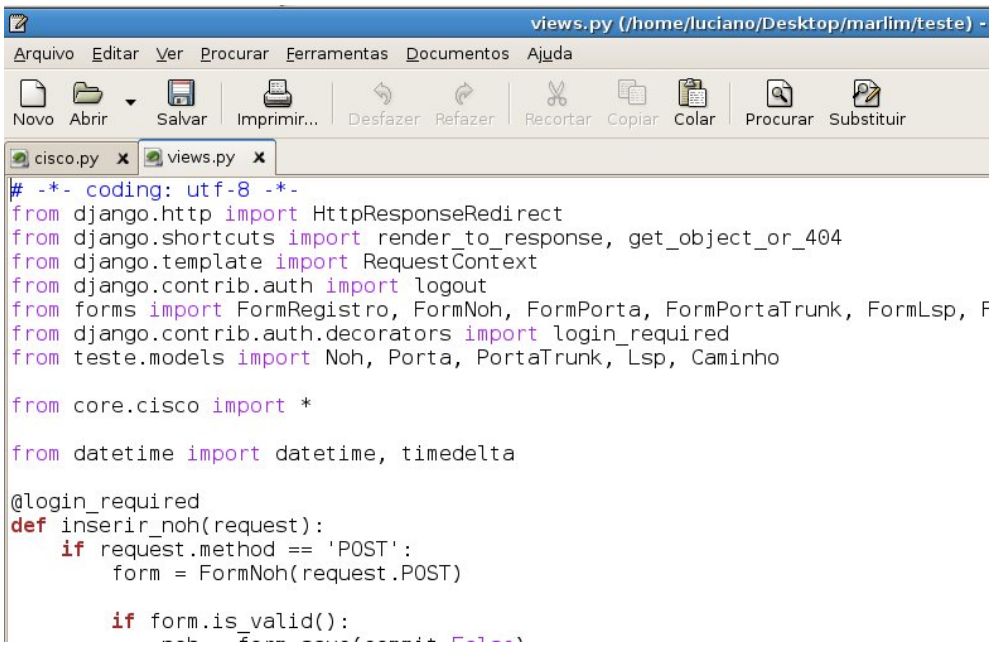

Figura B.1: Inserção da adaptação para Cisco.

Para que a opção do novo fabricante apareça no momento de inserção de um novo Nó, é necessário o último passo: acessar o endereço, via browser, adicionando o '/admin', logar na página com algum usuário que tenha acesso à página de administração e em seguida clicar no link 'Tipo nohs' da tabela chamada 'Teste' (figura B.2). Será exibida a tela com os tipos de Nós existentes, após clicar no link 'Adicionar tipo noh' será gerado um formulário em que será necessário colocar o nome com a primeira letra em maiúscula do fabricante. Ao clicar em 'Salvar' o sistema automaticamente oferecerá a nova opção para os usuários do artefato, sem necessidade de acréscimo de nenhuma outra modificação no código do sistema.

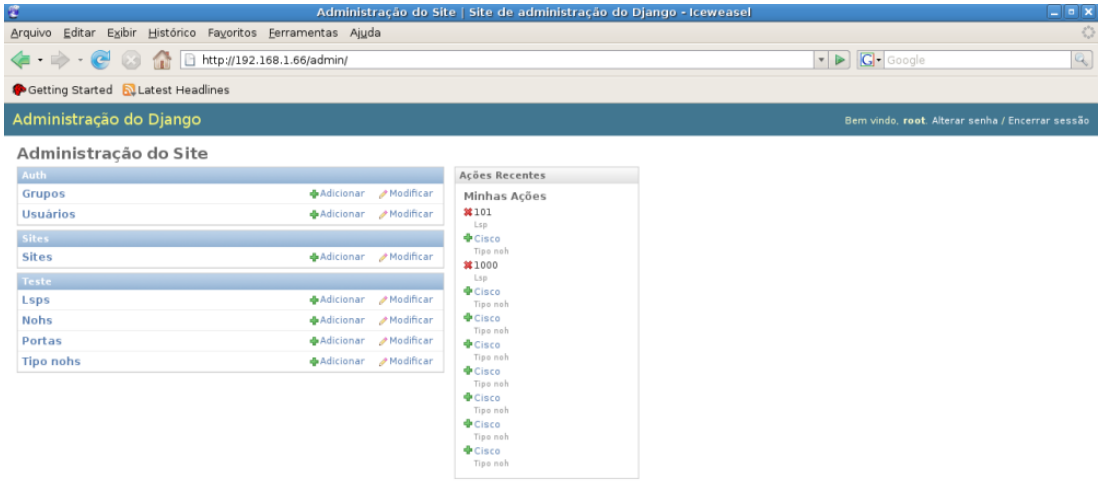

Figura B.2: Inserção de nova opção de Fabricante.

Apêndice C

Manual de Uso

O artefato funciona através de um programa *browser* de internet, no endereço do dispositivo onde foi instalado.

Ao acessar o endereço apropriado, uma tela com Usuário e Senha é apresentada (Figura C.1). Após executar o login, a tela principal do artefato é exibida (Figura C.2). Nesta tela é possível encontrar informações sobre: Nós criados, LSPs Desativados, LSPs Ativos e LSPS Agendados.

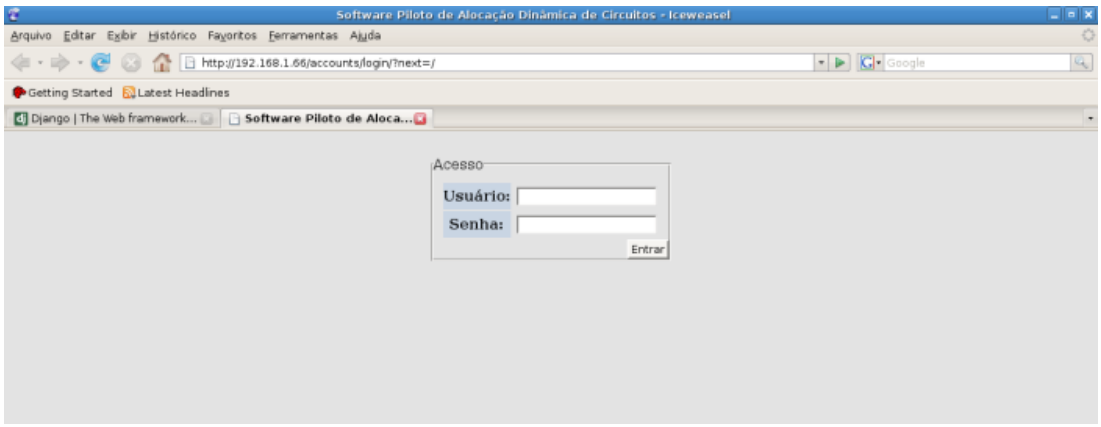

Figura C.1: Imagem da Tela de Login do Artefato.

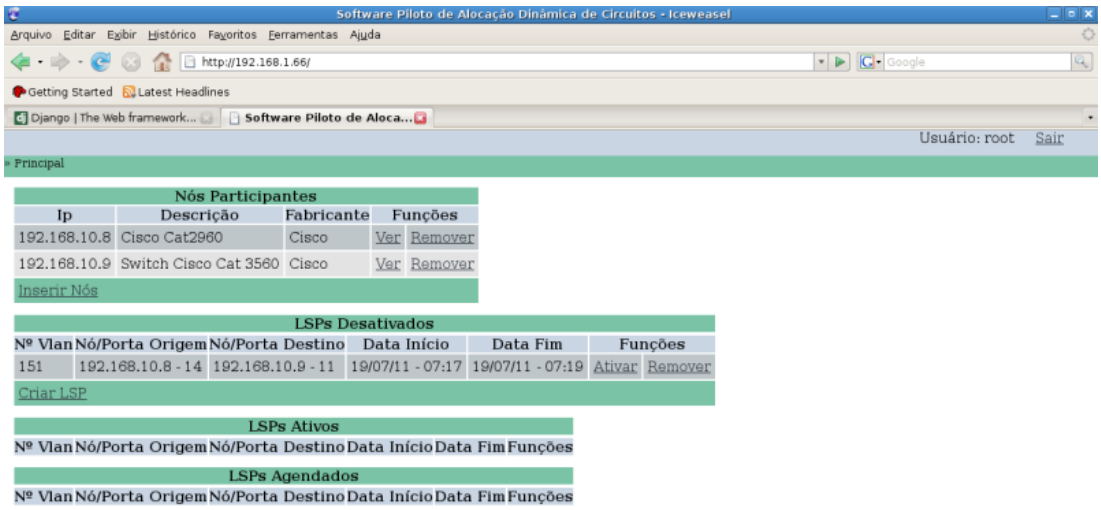

Figura C.2: Imagem da Tela Principal do Artefato.

C.1 Criação de Nós

Na tela principal na área de exibição dos Nós Participantes, é possível ver um link chamado “Inserir Nós”. Ao clicar nesse link, é gerado um formulário (Figura C.3) para inserir as seguintes informações:

Endereço IP - Endereço que o equipamento possui dentro da rede em que é utilizado.

Descrição - Campo de informação adicional sobre o dispositivo.

Senha SNMP - Campo com a senha para permissão de escrita e leitura no dispositivo.

Tipo - Menu com opções de tipos de dispositivos que o artefato é capaz de configurar.

Para salvar as informações inseridas no formulário clique no botão de “Inserir”.

C.1.1 Inserção de Portas Trunks

A inserção de Portas Trunks é necessária para a correta configuração do dispositivo quando este necessita criar uma VLAN que atravessa dispositivos distintos.

Para isso é necessário na tela principal na área de informações dos Nós, clicar no link “Ver”. Será exibida uma tela (Figura C.4) com a informação de Endereço IP, Descrição,

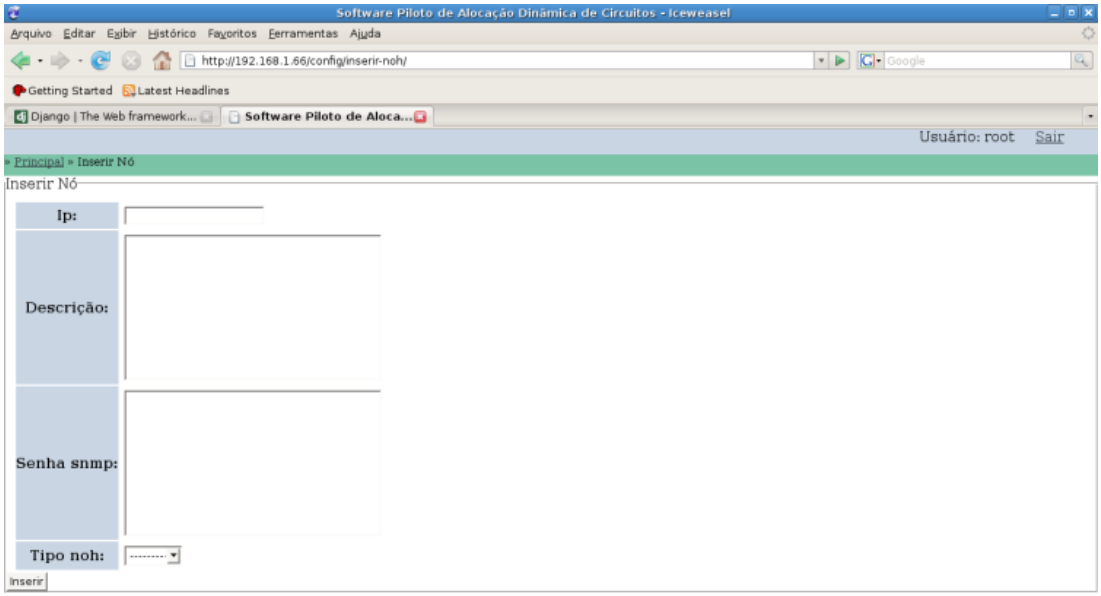

Figura C.3: Imagem do Formulário de Inserção de Nós.

Senha SNMP, Fabricante e também um link para Inserir Porta Trunk. Para criar uma Porta Trunk basta clicar neste link, que um formulário será exibido para que o número da porta seja inserido. Ao clicar no "Inserir" o artefato automaticamente configurará a porta em questão no equipamento.

Para deletar as portas trunks de um dispositivo, basta que na tela de informação do Nó, seja clicado o link "Remover Portas Trunk". Todas as Portas Trunk adicionados à este Nó serão apagados e desconfigurados no equipamento.

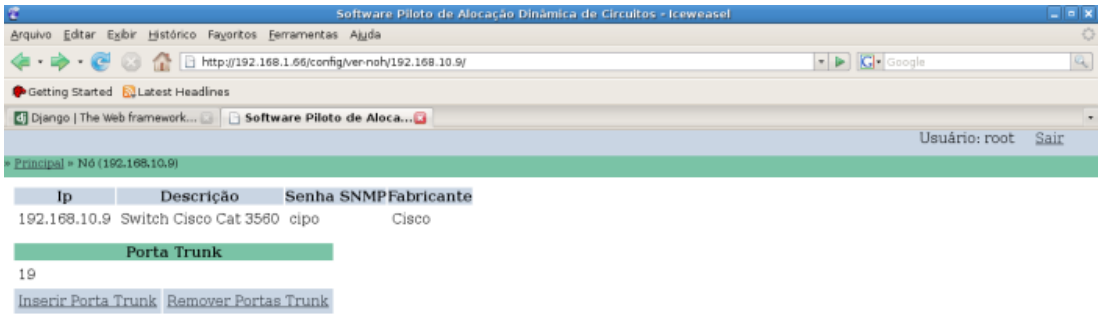

Figura C.4: Tela com informações sobre o Nó.

C.2 Criação de LSPs

Para criar um LSP, basta na tela principal, na área de LSPs Desativados, clicar no link “Criar LSPs”. Será exibido um formulário com as informações de Nós com seus respectivos endereços de origem e destino, assim como a porta de origem e destino, o número do LSP a ser criado e também o horário em que ele deverá ser configurado e desconfigurado (figura C.5). Ao clicar em “Salvar LSP” as informações serão salvas no artefato, porém somente quando se é clicado no link “Ativar LSP” é que serão configuradas adequadamente as informações nos equipamentos envolvidos.

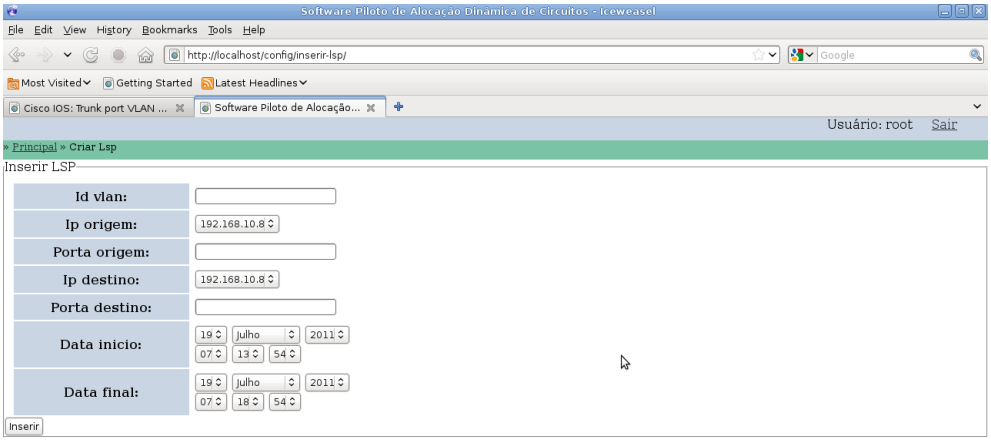

Figura C.5: Formulário para criação do LSP.

C.2.1 Agendamento

O agendamento de LSPs é feito quando, no momento da criação do LSP, é informado a data de início do LSP em uma data futura. Para que o agendamento seja executado pelo programa, ainda é necessário clicar no link “Ativar o LSP”. O LSP, em questão será considerado como agendado e ficará exposto na tabela de LSPs Agendados (figura C.6).

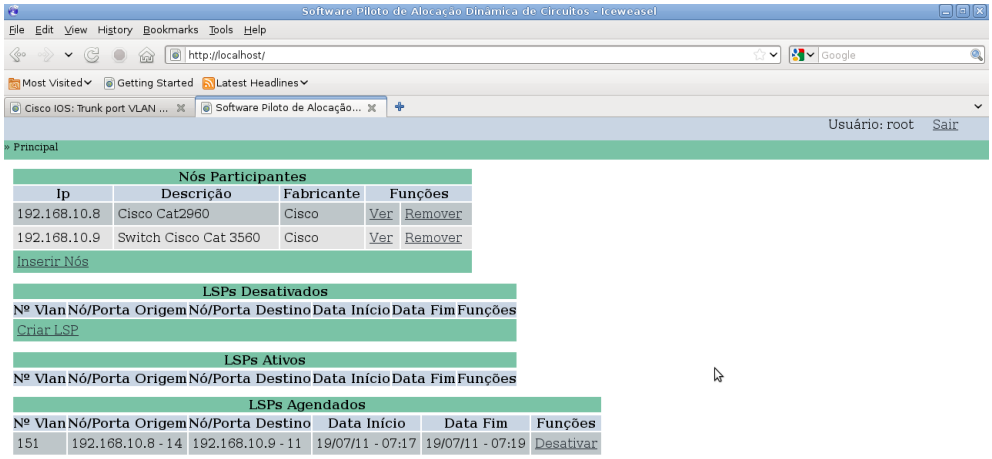

Figura C.6: Tela do Artefato mostrando o LSP Agendado.

Apêndice D

Código do artefato

Este apêndice apresenta o código do artefato criado com o intuito de recriação dos testes descritos e também como referência de estudos para um melhor entendimento da alocação dinâmica de recursos. Uma cópia do artefato com seus códigos e comentários, encontra-se disponível no CD junto ao arquivo com a cópia virtual desta dissertação.

D.1 Organização dos arquivos

O artefato e seus arquivos foram organizados da seguinte maneira:

```
/marlim - Pasta raiz onde o artefato foi instalado
 |-/core - Subpasta onde os arquivos de configuração específicos para equipamentos se encont
  |--views_snmp.py - Arquivo com as funções que se utilizam do PySNMP
  |--cisco.py - Arquivo com as funções específicas para o equipamento Cisco
 |-/media - Subpasta onde os arquivos de configuração de estilos se encontram
  |--/css
   |--estilo.css - Arquivo de configuração de estilos
 |-/templates - Subpasta onde arquivos gerais de apresentação no browser se encontram
 |-/teste - Subpasta onde a aplicação de alocação se encontra
```

```
  |--/templates - Subpasta onde arquivos da aplicação de alocação se encontram
  |--models.py - Arquivo com os modelos de interação com o banco de dados
  |--views.py - Arquivo com as funções da aplicação
  |--urls.py - Arquivo com as urls da aplicação
--settings.py - Arquivo de configuração do Django
--urls.py - Arquivo de configuração das urls gerais do sistema
--views.py - Arquivo com as funções gerais
```

D.2 Configuração do Framework

O arquivo abaixo é o de configuração do framework para seu correto funcionamento com a aplicação e interação com o banco de dados:

```
#arquivo de configuração settings.py
# Django settings for marlim project.
import os
PROJECT_ROOT_PATH = os.path.dirname(os.path.abspath(__file__))
DEBUG = True
TEMPLATE_DEBUG = DEBUG
ADMINS = (
    # ('Your Name', 'your_email@domain.com'),
)
MANAGERS = ADMINS
DATABASES = {
  'default': {
    'ENGINE': 'django.db.backends.mysql', # Add 'postgresql_psycopg2', 'postgresql', 'm
    'NAME': 'marlim',                     # Or path to database file if using sqlite3.
    'USER': 'marlim',                     # Not used with sqlite3.
```

```
    'PASSWORD': 'marlim',                  # Not used with sqlite3.
    'HOST': '',                       # Set to empty string for localhost. Not used with sqli
    'PORT': '',                       # Set to empty string for default. Not used with sqlite
  }
}
# Local time zone for this installation. Choices can be found here:
# http://en.wikipedia.org/wiki/List_of_tz_zones_by_name
# although not all choices may be available on all operating systems.
# On Unix systems, a value of None will cause Django to use the same
# timezone as the operating system.
# If running in a Windows environment this must be set to the same as your
# system time zone.
TIME_ZONE = 'America/Sao_Paulo'
# Language code for this installation. All choices can be found here:
# http://www.i18nguy.com/unicode/language-identifiers.html
LANGUAGE_CODE = 'pt-br'
SITE_ID = 1
# If you set this to False, Django will make some optimizations so as not
# to load the internationalization machinery.
USE_I18N = True
# If you set this to False, Django will not format dates, numbers and
# calendars according to the current locale
USE_L10N = True
# Absolute path to the directory that holds media.
# Example: "/home/media/media.lawrence.com/"
MEDIA_ROOT = os.path.join(PROJECT_ROOT_PATH,'media')
# URL that handles the media served from MEDIA_ROOT. Make sure to use a
```

```
# trailing slash if there is a path component (optional in other cases).
# Examples: "http://media.lawrence.com", "http://example.com/media/"
MEDIA_URL = '/media/'
# URL prefix for admin media -- CSS, JavaScript and images. Make sure to use a
# trailing slash.
# Examples: "http://foo.com/media/", "/media/".
ADMIN_MEDIA_PREFIX = '/admin_media/'
# Make this unique, and don't share it with anybody.
SECRET_KEY = 'ckvlc=ldyfo^4_=(3autj#yboo9m*4@*p8ru4of7y)0f&0z09u'
# List of callables that know how to import templates from various sources.
TEMPLATE_LOADERS = (
  'django.template.loaders.filesystem.Loader',
  'django.template.loaders.app_directories.Loader',
# 'django.template.loaders.eggs.Loader',
)
MIDDLEWARE_CLASSES = (
  'django.middleware.common.CommonMiddleware',
  'django.contrib.sessions.middleware.SessionMiddleware',
  'django.middleware.csrf.CsrfViewMiddleware',
  'django.contrib.auth.middleware.AuthenticationMiddleware',
  'django.contrib.messages.middleware.MessageMiddleware',
)
ROOT_URLCONF = 'urls'
TEMPLATE_DIRS = (
  # Put strings here, like "/home/html/django_templates" or "C:/www/django/templates".
  # Always use forward slashes, even on Windows.
  # Don't forget to use absolute paths, not relative paths.
```

```
  os.path.join(PROJECT_ROOT_PATH,"templates"),
)
INSTALLED_APPS = (
  'django.contrib.auth',
  'django.contrib.contenttypes',
  'django.contrib.sessions',
  'django.contrib.sites',
  'django.contrib.messages',
  # Uncomment the next line to enable the admin:
  'django.contrib.admin',
  'teste',
)
```

O arquivo abaixo é o de configuração do framework para o uso correto das urls da aplicação:

```
#Codigo de configuração urls.py
from django.conf.urls.defaults import *
from django.conf import settings
# Uncomment the next two lines to enable the admin:
from django.contrib import admin
admin.autodiscover()
urlpatterns = patterns('',
    # Example:
    # (r'^marlim/', include('marlim.foo.urls')),
    # Uncomment the admin/doc line below and add 'django.contrib.admindocs'
    # to INSTALLED_APPS to enable admin documentation:
    # (r'^admin/doc/', include('django.contrib.admindocs.urls')),
```

```
    # Uncomment the next line to enable the admin:
    (r'^admin/', include(admin.site.urls)),
    (r'^accounts/login/$', 'django.contrib.auth.views.login',
{'template_name': 'acesso.html'}, 'entrar'),
    (r'^sair/$', 'django.contrib.auth.views.logout',
{'next_page':'/'}, 'sair'),
    (r'^$', 'views.inicial', {}, 'inicial'),
    (r'^config/', include('teste.urls')),
    (r'^media/(.*)$', 'django.views.static.serve',
 {'document_root': settings.MEDIA_ROOT}),
)
```

O arquivo abaixo é o de configuração da view do framework:

```
#Codigo de configuração views.py
# -*- coding: utf-8 -*-
from django.http import HttpResponseRedirect
from django.shortcuts import render_to_response, get_object_or_404
from django.template import RequestContext
from django.contrib.auth import logout
from django.contrib.auth.decorators import login_required
from teste.models import Noh, Lsp

@login_required
def inicial(request):
    nohs = Noh.objects.filter()
    lsps_desativados = Lsp.objects.filter(estado='AE').order_by('id_vlan')
    lsps_ativos = Lsp.objects.filter(estado='AN').order_by('id_vlan')
```

```
    lsps_agendados = Lsp.objects.filter(estado='AG').order_by('id_vlan')


    return render_to_response(
        'tela_principal.html',
        locals(),
        context_instance=RequestContext(request),
        )
```

D.3 Configuração do arquivo core

Dentro da subpasta core, ficam armazenados os arquivos específicos de configuração dos switches que o software é capaz de configurar assim como o arquivo que executa funções do framework PySNMP.

O arquivo abaixo é o de configuração do PySNMP dentro da pasta core:

```
#Código arquivo views_snmp.py
 # -*- coding: utf-8 -*-
import string
from pysnmp.entity.rfc3413.oneliner import cmdgen
from pysnmp.proto import rfc1902


def func_string_tuple(numero):
    '''
    Função que recebe uma string de oid e retorna uma tupla.
Ex: "1.6.1.3.2.2" -> (1,6,1,3,2,2)
    '''
    aux = numero.split('.')
    return tuple(map(int,aux))
```

```
def get_cmd(address,oid):
    '''
    Função que recebe duas strings, a primeira do endereco do
dispositivo e outra do oid e retorna o valor
    '''
    errorIndication, errorStatus, errorIndex, varBinds =
 cmdgen.CommandGenerator().getCmd(
    cmdgen.CommunityData('my-agent', 'public', 0),
    cmdgen.UdpTransportTarget((address, 161)),
    func_string_tuple(oid)
    )
    return varBinds

def next_cmd(address,oid, comunidade_escrita):
    '''
    Função que recebe duas strings, a primeira do endereco
do dispositivo e outra do oid e retorna uma tabela de valores
    '''
    errorIndication, errorStatus, errorIndex, varBindTable =
cmdgen.CommandGenerator().nextCmd(
    cmdgen.CommunityData('my-agent', comunidade_escrita, 1),
    cmdgen.UdpTransportTarget((address, 161)),
    func_string_tuple(oid)
    )
    print errorIndication
    print errorStatus
```

```
    print errorIndex
    return varBindTable


def set_cmd(address,oid,tipo,parametro,comunidade_escrita):
    '''
    Função que pega o endereço, oid junto com o tipo do parametro
 e mais o parametro que se quer configurar e a comunidade
 de escrita do switch
    '''
    errorIndication, errorStatus, errorIndex, varBinds =
 cmdgen.CommandGenerator().setCmd(
    cmdgen.CommunityData('my-agent', comunidade_escrita, 1),
    cmdgen.UdpTransportTarget((address, 161)),
    (func_string_tuple(oid), getattr(rfc1902, tipo)(parametro))
    )
    print errorIndication
    print errorStatus.prettyPrint()
    print errorIndex
    return varBinds
```

O arquivo abaixo é o de configuração em switches Cisco:

```
#Código do arquivo cisco.py
# -*- coding: utf-8 -*-
import string
from pysnmp.entity.rfc3413.oneliner import cmdgen
from pysnmp.proto import rfc1902
from core import views_snmp
```

```
from binascii import unhexlify #Faz a conversão do endereço
do Octet String hexadecimal
from operator import itemgetter #Grupamento de dados
from itertools import groupby

def criar_vlan_Cisco(address,numero_vlan,comunidade_escrita):
    '''
    Função que cria uma vlan no switch cisco a partir do endereço, o número da vlan
    '''
    views_snmp.set_cmd(address,'1.3.6.1.4.1.9.9.46.1.4.1.1.1.1',
'Integer',2, comunidade_escrita)#Permitindo a criação da VLan
    views_snmp.set_cmd(address,'1.3.6.1.4.1.9.9.46.1.4.1.1.3.1',
'OctetString','Vix', comunidade_escrita)#Fazendo a edição do dono atual ser visível
    #Editando a VLAN
    views_snmp.set_cmd(address,'1.3.6.1.4.1.9.9.46.1.4.2.1.11.1.'
 + str(numero_vlan),'Integer',4, comunidade_escrita)
    views_snmp.set_cmd(address,'1.3.6.1.4.1.9.9.46.1.4.2.1.3.1.'
 + str(numero_vlan),'Integer',1, comunidade_escrita)
    views_snmp.set_cmd(address,'1.3.6.1.4.1.9.9.46.1.4.2.1.4.1.'
 + str(numero_vlan),'OctetString','Vix' + str(numero_vlan), comunidade_escrita)
    #Cisco sempre faz 100000 + numero da vlan = resultado
transforma em octeto hexadecimal
    hexadecimal = 100000 + numero_vlan
    hexadecimal = hex(hexadecimal)
    while len(hexadecimal)<9:#Acrescenta os zeros necessários para
 completar oito bits e retira o x
        hexadecimal = '0'+hexadecimal
```

```
    hexadecimal = hexadecimal.replace('x','')
    hexadecimal = unhexlify(hexadecimal)#Converte para Octet String
    views_snmp.set_cmd(address,'1.3.6.1.4.1.9.9.46.1.4.2.1.6.1.'
 + str(numero_vlan),'OctetString',hexadecimal, comunidade_escrita)
    views_snmp.set_cmd(address,'1.3.6.1.4.1.9.9.46.1.4.1.1.1.1',
'Integer',3, comunidade_escrita)
    views_snmp.set_cmd(address,'1.3.6.1.4.1.9.9.46.1.4.1.1.1.1',
'Integer',4, comunidade_escrita)

def deletar_vlan_Cisco(address,numero_vlan, comunidade_escrita):
    '''
    Função que deleta uma vlan no switch cisco a partir do endereço
 e o número da vlan
    '''
    views_snmp.set_cmd(address,'1.3.6.1.4.1.9.9.46.1.4.1.1.1.1','Integer',
2,comunidade_escrita)#Permitindo a criação da Vlan
    views_snmp.set_cmd(address,'1.3.6.1.4.1.9.9.46.1.4.1.1.3.1','OctetString',
'Vix', comunidade_escrita)#Fazendo a edição do dono atual ser visível
    #Deletando a VLAN
    views_snmp.set_cmd(address,'1.3.6.1.4.1.9.9.46.1.4.2.1.11.1.'
 + str(numero_vlan),'Integer',6, comunidade_escrita)
    views_snmp.set_cmd(address,'1.3.6.1.4.1.9.9.46.1.4.1.1.1.1',
'Integer',3, comunidade_escrita)
    views_snmp.set_cmd(address,'1.3.6.1.4.1.9.9.46.1.4.1.1.1.1',
'Integer',4, comunidade_escrita)

def tipo_interface_Cisco(address,comunidade_escrita):
```

```
    '''
    Função que avalia se a interface é FasEthernet, GigabitEthernet
 ou qualquer outra que possa vir
    '''
    interfaces=views_snmp.next_cmd(address,'1.3.6.1.2.1.2.2.1.2',
comunidade_escrita)
    for x in interfaces:
        for y,z in x:
            if 'FastEthernet0/1' in str(z):
                return '100'
            if 'GigabitEthernet0/1' in str(z):
                return '101'


def ligar_porta_Cisco(address,porta,comunidade_escrita):
    '''
    Função que liga a porta dada do switch assinalado
    '''
    porta = str(porta)
    while len(porta)<2:#Acrescenta os zeros necessários para completar
 dois digitos
        porta = '0'+porta
    views_snmp.set_cmd(address,'1.3.6.1.2.1.2.2.1.7.'
+ tipo_interface_Cisco(address,comunidade_escrita) + porta,'Integer',
1,comunidade_escrita)#Ligando a porta


def desligar_porta_Cisco(address,porta,comunidade_escrita):
    '''
```

```
    Função que desliga a porta dada do switch assinalado
    '''
    porta = str(porta)
    while len(porta)<2:#Acrescenta os zeros necessários para completar
 dois digitos
        porta = '0'+porta
    views_snmp.set_cmd(address,'1.3.6.1.2.1.2.2.1.7.'
+ tipo_interface_Cisco(address,comunidade_escrita) + porta,'Integer',
2,comunidade_escrita)#Ligando a porta

def vlan_existe_Cisco(address,numero_vlan,comunidade_escrita):
    '''
    Função que verifica se a vlan a ser deletada realmente existe
    '''
    vlans=views_snmp.next_cmd(address,'1.3.6.1.4.1.9.9.46.1.3.1.1.2',
comunidade_escrita)
    for x in vlans:
        for y,z in x:
            if y[-1] == numero_vlan:
                return True
    return False

def adicionar_vlan_porta_Cisco(address,numero_vlan,porta, comunidade_escrita):
    '''
    Função que adiciona a porta a uma vlan se esta existe e ainda
trata se a interface é Ethernet ou GigabitEthernet
    '''
```

```
    porta = str(porta)
    while len(porta)<2:#Acrescenta os zeros necessários para completar
 dois digitos
        porta = '0'+porta
    if vlan_existe_Cisco(address,numero_vlan,comunidade_escrita):
        views_snmp.set_cmd(address,'1.3.6.1.4.1.9.9.68.1.2.2.1.2.'
+ tipo_interface_Cisco(address,comunidade_escrita) + porta,'Integer',
str(numero_vlan), comunidade_escrita)

def retirar_vlan_porta_Cisco(address,numero_vlan,porta,comunidade_escrita):
    '''
    Faz a porta ficar com a vlan default 1
    '''
    porta = str(porta)
    while len(porta)<2:#Acrescenta os zeros necessários para completar
 dois digitos
        porta = '0'+porta
    views_snmp.set_cmd(address,'1.3.6.1.4.1.9.9.68.1.2.2.1.2.'
+ tipo_interface_Cisco(address,comunidade_escrita) + porta,'Integer',
1, comunidade_escrita)

def adicionar_native_vlan_trunk_porta_Cisco(address,numero_vlan,
porta,comunidade_escrita):
    '''
    Função que adiciona a porta a uma vlan se esta existe
    '''
    porta = str(porta)
```

```
    while len(porta)<2:#Acrescenta os zeros necessários para completar
 dois digitos
        porta = '0'+porta
    if vlan_existe_Cisco(address,numero_vlan,comunidade_escrita):
        views_snmp.set_cmd(address,'1.3.6.1.4.1.9.9.46.1.6.1.1.5.'
+ tipo_interface_Cisco(address,comunidade_escrita) + porta,'Integer',
str(numero_vlan), comunidade_escrita)
        return 'Trunk adicionado'
    else:
        return 'Vlan nao existe'

def criar_interface_mode_trunk_Cisco(address,porta,comunidade_escrita):
    '''
    Função que configura a porta para virar trunk
    '''
    porta = str(porta)
    while len(porta)<2:#Acrescenta os zeros necessários para completar
dois digitos
        porta = '0'+porta
    views_snmp.set_cmd(address,'1.3.6.1.4.1.9.9.46.1.6.1.1.3.'+
 tipo_interface_Cisco(address,comunidade_escrita) + porta,'Integer',
4, comunidade_escrita)#Criando o encapsulamento dot1q
    views_snmp.set_cmd(address,'1.3.6.1.4.1.9.9.46.1.6.1.1.13.'+
 tipo_interface_Cisco(address,comunidade_escrita) + porta,'Integer',
1, comunidade_escrita)#Criando o switchport mode trunk

def retirar_interface_mode_trunk_Cisco(address,porta,comunidade_escrita):
```

```
    ,,,
    Função que desconfigura a porta para não ser mais trunk
    ,,,
    porta = str(porta)
    while len(porta)<2:#Acrescenta os zeros necessários para completar
 dois digitos
        porta = '0'+porta
    views_snmp.set_cmd(address,'1.3.6.1.4.1.9.9.46.1.6.1.1.13.'+
tipo_interface_Cisco(address,comunidade_escrita) + porta,'Integer',
2, comunidade_escrita)#Retirando switchport mode trunk
    views_snmp.set_cmd(address,'1.3.6.1.4.1.9.9.46.1.6.1.1.3.'+
 tipo_interface_Cisco(address,comunidade_escrita) + porta,'Integer',
5, comunidade_escrita)#Tirando o encapsulamento dot1q
#------------- Funções para Trunk na Porta ------------------
def criar_vlan_trunk_porta_Cisco(address,string,chunk,porta,comunidade_escrita):
    ,,,
    Função que cria a vlans trunks que serão permitidas em uma porta trunk
    ,,,
    porta = str(porta)
    while len(porta)<2:#Acrescenta os zeros necessários para completar
 dois digitos
        porta = '0'+porta
    print type(string)
    if chunk == '1024':
        views_snmp.set_cmd(address,'1.3.6.1.4.1.9.9.46.1.6.1.1.4.'+
 tipo_interface_Cisco(address,comunidade_escrita) + porta,'OctetString',
string, comunidade_escrita)
```

```
    if chunk == '2048':
        views_snmp.set_cmd(address,'1.3.6.1.4.1.9.9.46.1.6.1.1.17.'+
 tipo_interface_Cisco(address,comunidade_escrita) + porta,'OctetString',
string, comunidade_escrita)
    if chunk == '3072':
        views_snmp.set_cmd(address,'1.3.6.1.4.1.9.9.46.1.6.1.1.18.'+
tipo_interface_Cisco(address,comunidade_escrita) + porta,'OctetString',
string, comunidade_escrita)
    if chunk == '4096':
        views_snmp.set_cmd(address,'1.3.6.1.4.1.9.9.46.1.6.1.1.19.'+
tipo_interface_Cisco(address,comunidade_escrita) + porta,'OctetString',
string, comunidade_escrita)

def vlan_trunk_octeto_Cisco(lista):
    '''
    Função que recebe uma lista com a posição e o valor do octeto e
devolve toda a string que deve ser colocada no cisco para setar as
vlans trunks
    '''
    octeto="".join(256*['0'])
    for i in lista:
        if i[0] == 0:
            octeto = i[1] + octeto[1:]
        else:
            octeto = octeto[:i[0]] + i[1] + octeto[i[0]+1:]
    return unhexlify(octeto)
```

```
def vlan_lista_posicao(lista):
    '''
    Função que recebe uma lista e devolve a posição e soma de bits no
quarteto de bits em que deve ficar e convertido para o valor hexadecimal
    obs: 3-i%4 já dá o indice dentro do bloco de 4 bits em que o bit está posicionado
    '''
    lista_aux=[]
    for i in lista:
        lista_aux.append([i,i/4,3-i%4])
    lista=groupby(lista_aux, itemgetter(1))
    lista_valor=[]
    for i,items in groupby(lista_aux,itemgetter(1)):
        valor=0
        for x in items:
            valor=valor+pow(2,x[2])#Faz a soma dos bits na posição que
se encontram dentro do quarteto de bits
        lista_valor.append([i,str(valor)])
    for i in lista_valor:
         if i[1] == '10':
             i[1] = 'a'
         if i[1] == '11':
             i[1] = 'b'
         if i[1] == '12':
             i[1] = 'c'
         if i[1] == '13':
             i[1] = 'd'
         if i[1] == '14':
```

```
            i[1] = 'e'
        if i[1] == '15':
            i[1] = 'f'
    return lista_valor

def vlan_trunk_Cisco(address,lista_vlans,porta,comunidade_escrita):
    '''
    Função que recebe uma lista de numeros de vlans e a transforma nos
octetos que o cisco permite na configuração de vlans trunks e configura as vlans
    trunks na porta do switch de endereço address e com permissão de escrita
    '''
    lista1024 =[]
    lista2048 =[]
    lista3072 =[]
    lista4096 =[]#Necessário para dividir as vlans nos quatro grupos
 do padrão da Cisco
    for numero in lista_vlans:
        if numero < 4096:
            if numero >= 3072:
                lista4096.append(numero)
            else:
                if numero >= 2048:
                    lista3072.append(numero)
                else:
                    if numero >= 1024:
                        lista2048.append(numero)
                    else:
```

```
                        lista1024.append(numero)
    lista1024 = sorted(lista1024)#Ordenando as listas
    lista2048 = sorted(lista2048)
    lista3072 = sorted(lista3072)
    lista4096 = sorted(lista4096)
    print lista1024
    lista2048 = [numero-1024 for numero in lista2048]#Trazendo para a
divisão padrão da Cisco
    lista3072 = [numero-2048 for numero in lista3072]
    lista4096 = [numero-3072 for numero in lista4096]
    lista1024 = vlan_lista_posicao(lista1024)#Criando o mapeamento
 de bits padrão da Cisco
 - http://www.ciscopress.com/articles/article.asp?p=29803&seqNum=2
    lista2048 = vlan_lista_posicao(lista2048)
    lista3072 = vlan_lista_posicao(lista3072)
    lista4096 = vlan_lista_posicao(lista4096)
    lista1024 = vlan_trunk_octeto_Cisco(lista1024)#Gerando o octeto
string que é necessário para setar as trunks vlan
    lista2048 = vlan_trunk_octeto_Cisco(lista2048)
    lista3072 = vlan_trunk_octeto_Cisco(lista3072)
    lista4096 = vlan_trunk_octeto_Cisco(lista4096)
    print lista1024
    print type(lista1024)
    criar_vlan_trunk_porta_Cisco(address,lista1024,'1024',porta,
comunidade_escrita)#Fazendo a configuração no switch
    criar_vlan_trunk_porta_Cisco(address,lista2048,'2048',porta,
comunidade_escrita)
```

```
    criar_vlan_trunk_porta_Cisco(address,lista3072,'3072',porta,
comunidade_escrita)
    criar_vlan_trunk_porta_Cisco(address,lista4096,'4096',porta,
comunidade_escrita)
    return 'OK'
```

D.4 Configuração da Aplicação

Como o artefato foi desenvolvido a partir do framework Django, os códigos apresentados aqui estão divididos nas três camadas conceituais de Model-View-Controller. Respectivamente os modelos que interagem com o banco de dados (models.py), os *templates* de apresentação do artefato no browser (extensão .html) e os controladores (chamados no Django de views.py).

O arquivo abaixo é o de modelo de interação com o banco de dados:

```
#Código do models.py
#-*- coding: utf 8 -*-
from django.db import models
class TipoNoh(models.Model):
    nome = models.TextField()
    def __unicode__(self):
        return self.nome

class Noh(models.Model):
    ip = models.IPAddressField()#xxx.xxx.xxx.xxx
    descricao = models.TextField('Descrição',null=True, blank=True)
    senha_snmp = models.TextField()
    tipo_noh = models.ForeignKey(TipoNoh)
```

```
    def __unicode__(self):
        return self.ip

ESTADOS_PORTA = (
    ('O','Ocupado'),
    ('L','Livre'),
)

class Porta(models.Model):
    numero = models.IntegerField()
    noh = models.ForeignKey(Noh, blank=True, null=True)
    estado = models.CharField(max_length=1, default='L', choices=ESTADOS_PORTA)
    def __unicode__(self):
        return "%s"%(self.numero)

class PortaTrunk(models.Model):
    numero = models.IntegerField()
    noh = models.ForeignKey(Noh, blank=True, null=True)
    def __unicode__(self):
        return "%s"%(self.numero)

ESTADOS_LSP = (
    ('AE','A ser Estabelecido'),
    ('AG','Agendado'),
    ('AN','Ativo Normal'),
    ('AA','Ativo Agendado'),
    ('FA','Falha'),
```

```
    ('JA','Jah Usado')
)

class Lsp(models.Model):
    id_vlan = models.IntegerField()#Numero da Vlan
    ip_origem = models.IPAddressField()
    porta_origem = models.IntegerField(blank=True)#Apenas para gravar o lsp
    ip_destino = models.IPAddressField()
    porta_destino = models.IntegerField(blank=True)#Na
 verdade a view terá que apresentar as portas relacionadas a cada noh
    data_inicio = models.DateTimeField(blank=True, null=True)
    data_final = models.DateTimeField(blank=True, null=True)

    estado = models.CharField(max_length=2, default='AE', choices=ESTADOS_LSP)

    def __unicode__(self):
        return "%s"%(self.id_vlan)

class Caminho(models.Model):#Modelo é indicado
 no template onde é criado o caminho de nós por onde o lsp irá passar
    lsp = models.ForeignKey(Lsp, null=True, blank=True)
    ip = models.IPAddressField(null=True,blank=True)
#Através de forms são dadas as opções de nó existentes
```

O arquivo abaixo é o de funcões gerais para a alocação de LSPs:

```
#Código da views do sistema (views.py)
# -*- coding: utf-8 -*-
```

```
from django.http import HttpResponseRedirect
from django.shortcuts import render_to_response,
 get_object_or_404
from django.template import RequestContext
from django.contrib.auth import logout
from forms import FormRegistro, FormNoh, FormPorta,
 FormPortaTrunk, FormLsp, FormCaminho
from django.contrib.auth.decorators import login_required
from teste.models import Noh, Porta, PortaTrunk, Lsp, Caminho


from core.cisco import *


from datetime import datetime, timedelta


@login_required
def inserir_noh(request):
    if request.method == 'POST':
        form = FormNoh(request.POST)


        if form.is_valid():
            noh = form.save(commit=False)


            noh.save()


            return HttpResponseRedirect('/')
    else:
        form = FormNoh()
```

```
    return render_to_response(
        'teste/inserir_noh.html',
        locals(),
        context_instance=RequestContext(request),
        )

@login_required
def remover_noh(request, noh):
    noh = get_object_or_404(Noh, ip=noh)
    portatrunks = PortaTrunk.objects.filter(noh=noh).
order_by('numero')

    for portatrunk in portatrunks:
        funcao= 'retirar_interface_mode_trunk_'+
noh.tipo_noh.nome #Primeiro chama-se o aquivo e
depois a funcao que sempre termina no Tipo do Noh
        eval(funcao)(noh.ip,int(portatrunk.numero),
noh.senha_snmp)

    noh.delete()

    return HttpResponseRedirect('/')

@login_required
def ver_noh(request, noh):
     noh = get_object_or_404(Noh, ip=noh)
```

```
     portatrunks = PortaTrunk.objects.filter(noh=noh)
.order_by('numero')

     return render_to_response(
         'teste/ver_noh.html',
         locals(),
         context_instance=RequestContext(request),
         )

@login_required
def inserir_porta(request, noh):
    if request.method == 'POST':
        form = FormPorta(request.POST)

        if form.is_valid():
            porta = form.save(commit=False)
            porta.noh = get_object_or_404(Noh, ip=noh)
            porta.save()

            return HttpResponseRedirect('/config/ver-noh/'+ "%s"%(noh))
    else:
        form = FormPorta()

    return render_to_response(
        'teste/inserir_noh.html',
        locals(),
        context_instance=RequestContext(request),
```

```
        )


@login_required
def inserir_porta_trunk(request, noh):
    if request.method == 'POST':
        form = FormPortaTrunk(request.POST)


        if form.is_valid():
            porta = form.save(commit=False)
            porta.noh = get_object_or_404(Noh, ip=noh)
            tipo_noh = porta.noh.tipo_noh#Pegando o
tipo do Noh
            funcao= 'criar_interface_mode_trunk_'+
tipo_noh.nome #Primeiro chama-se o aquivo e depois a
funcao que sempre termina no Tipo do Noh
            eval(funcao)(porta.noh.ip,porta.numero,
porta.noh.senha_snmp)
            porta.save()


            return HttpResponseRedirect('/config/ver-noh/'+ "%s"%(noh))
    else:
        form = FormPortaTrunk()


    noh=noh
    return render_to_response(
        'teste/inserir_porta_trunk.html',
        locals(),
```

```
        context_instance=RequestContext(request),
        )


@login_required
def remover_portas_trunk(request, noh):
    noh = get_object_or_404(Noh, ip=noh)
    portatrunks = PortaTrunk.objects.filter(noh=noh).
order_by('numero')

    for portatrunk in portatrunks:
        funcao= 'retirar_interface_mode_trunk_'+
 noh.tipo_noh.nome #Primeiro chama-se o aquivo e
depois a funcao que sempre termina no Tipo do Noh
        eval(funcao)(noh.ip,int(portatrunk.numero),
noh.senha_snmp)

    return HttpResponseRedirect('/config/ver-noh/'+ "%s"%(noh))



@login_required
def inserir_lsp(request):
    if request.method == 'POST':
        form = FormLsp(request.POST)

        if form.is_valid():
            form.save()
```

```
            return HttpResponseRedirect('/')
    else:
        form = FormLsp(
            initial={
                'data_inicio': datetime.now(),
                'data_final': datetime.now() +
timedelta(minutes=5),
                },
            )
    return render_to_response(
        'teste/inserir_lsp.html',
        locals(),
        context_instance=RequestContext(request),
        )


@login_required
def ver_caminho(request, lsp):
     lsp=lsp#Só para ter a variável mostrada no template
     caminhos = Caminho.objects.filter(lsp__id_vlan=lsp)
     return render_to_response(
         'teste/ver_caminho.html',
         locals(),
         context_instance=RequestContext(request),
         )


@login_required
def inserir_caminho(request, lsp):
```

```
    if request.method == 'POST':
        form = FormCaminho(request.POST)

        if form.is_valid():
            caminho = form.save(commit=False)
            caminho.lsp = get_object_or_404(Lsp,
id_vlan=lsp)
            caminho.save()

            return HttpResponseRedirect
('/config/ver-caminho/'+ "%s"%(lsp))
    else:
        form = FormCaminho()

    return render_to_response(
        'teste/inserir_caminho.html',
        locals(),
        context_instance=RequestContext(request),
        )

@login_required
def ativar_lsp(request, lsp):
    lsp_aux = Lsp.objects.get(id_vlan=lsp)#Pega
o objeto lsp

    if lsp_aux.data_inicio >  datetime.now():
        lsp_aux.estado = 'AG' #Estado agendado
```

```
        lsp_aux.save()
        return HttpResponseRedirect('/')
    else:
        configurar_lsp(lsp)
        return HttpResponseRedirect('/')


def ativar_lsp_agendados():#Roda com o Programa Cron
    lsps_agendados = Lsp.objects.filter(estado='AG')
#Pega os lsps agendados

    for lsp in lsps_agendados:
        if lsp.data_inicio.year < datetime.now().year or
lsp.data_inicio.month < datetime.now().month or
lsp.data_inicio.day < datetime.now().day or lsp.data_inicio.hour
< datetime.now().hour or lsp.data_inicio.minute <=
datetime.now().minute:#Necessário para retirar a parte
dos segundos por causa do cron
            configurar_lsp(str(lsp))

    return True

def desativar_lsps():#Roda com o Programa Cron
    lsps_ativos = Lsp.objects.filter(estado='AN')#Pega
os lsps agendados

    for lsp in lsps_ativos:
```

```
        if lsp.data_final.year < datetime.now().year
or lsp.data_final.month < datetime.now().month or
lsp.data_final.day < datetime.now().day or lsp.data_final.hour
< datetime.now().hour or lsp.data_final.minute <=
datetime.now().minute:#Necessário para retirar a parte
dos segundos por causa do cron
            desativar_lsp(str(lsp))

    return True

def configurar_lsp(lsp):
    lsp_existentes = Lsp.objects.filter(estado='AN')
    nohs_existentes = Noh.objects.filter()
    #Cria a vlan no switch de origem e agrega à porta,
 faz o mesmo com o switch de destino
    lsp = Lsp.objects.get(id_vlan=lsp)#Pega o objeto lsp

    origem = Noh.objects.get(ip=lsp.ip_origem)
    tipo_origem = origem.tipo_noh
    funcao_origem1 = 'criar_vlan_'+ tipo_origem.nome
    eval(funcao_origem1)(origem.ip,int(lsp.id_vlan),
origem.senha_snmp)#Criando a vlan na origem
    funcao_origem2 = 'ligar_porta_'+ tipo_origem.nome
    eval(funcao_origem2)(origem.ip,int(lsp.porta_origem),
origem.senha_snmp)#Ligando a porta de origem
    funcao_origem3 = 'adicionar_vlan_porta_' + tipo_origem.nome
    eval(funcao_origem3)(origem.ip,int(lsp.id_vlan),
```

```
int(lsp.porta_origem),origem.senha_snmp)#Adicionando
a vlan à porta origem

    destino = Noh.objects.get(ip=lsp.ip_destino)
    tipo_destino = destino.tipo_noh
    funcao_destino1 = 'criar_vlan_'+ tipo_destino.nome
    eval(funcao_destino1)(destino.ip,int(lsp.id_vlan),
destino.senha_snmp)#Criando a vlan no destino
    funcao_destino2 = 'ligar_porta_'+ tipo_destino.nome
    eval(funcao_destino2)(destino.ip,int(lsp.porta_destino),
destino.senha_snmp)#Ligando a porta de destino
    funcao_destino3 = 'adicionar_vlan_porta_' +
tipo_destino.nome
    eval(funcao_destino3)(destino.ip,int(lsp.id_vlan),
int(lsp.porta_destino),destino.senha_snmp)#Adicionando
a vlan à porta destino

    lsps = [int(lsp.id_vlan)]
    for lista_lsp in lsp_existentes:#Adicionando as vlans
já existentes
        lsps.append(int(lista_lsp.id_vlan))#O python guarda
os valores como long int e por isso é necessário a
conversão novamente para int

    for nohs in nohs_existentes:
        porta_trunks = PortaTrunk.objects.filter(noh=nohs)
        for porta_trunk in porta_trunks:
```

```
            funcao= 'vlan_trunk_'+ nohs.tipo_noh.nome
#Primeiro chama-se o aquivo e depois a funcao que sempre
termina no Tipo do Noh
            eval(funcao)(nohs.ip,lsps,int(porta_trunk.numero),
nohs.senha_snmp)

    lsp.estado='AN'#Colocando no estado ativo
    lsp.save()

    return HttpResponseRedirect('/')

@login_required
def desativar_lsp_agendado(request, lsp):
    lsp = Lsp.objects.get(id_vlan=lsp)#Pega o objeto lsp
    lsp.estado = 'AE' #Estado default
    lsp.save()

    return HttpResponseRedirect('/')



def desativar_lsp(lsp):
    lsp_existentes = Lsp.objects.filter(estado='AN')
    nohs_existentes = Noh.objects.filter()
    #Cria a vlan no switch de origem e agrega à porta,
faz o mesmo com o switch de destino
    lsp = Lsp.objects.get(id_vlan=lsp)#Pega o objeto lsp
```

```
    origem = Noh.objects.get(ip=lsp.ip_origem)
    tipo_origem = origem.tipo_noh
    funcao_origem1 = 'deletar_vlan_'+ tipo_origem.nome
    eval(funcao_origem1)(origem.ip,int(lsp.id_vlan),
origem.senha_snmp)#Deletando a vlan
    funcao_origem2 = 'retirar_vlan_porta_' + tipo_origem.nome
    eval(funcao_origem2)(origem.ip,int(lsp.id_vlan),
int(lsp.porta_origem),origem.senha_snmp)#Retirando a
vlan da porta
    funcao_origem3 = 'desligar_porta_'+ tipo_origem.nome
    eval(funcao_origem3)(origem.ip,int(lsp.porta_origem),
origem.senha_snmp)#Desligando a porta de origem



    destino = Noh.objects.get(ip=lsp.ip_destino)
    tipo_destino = destino.tipo_noh
    funcao_destino1 = 'deletar_vlan_'+ tipo_destino.nome
    eval(funcao_destino1)(destino.ip,int(lsp.id_vlan),
destino.senha_snmp)#Deletando a vlan
    funcao_destino2 = 'retirar_vlan_porta_' + tipo_destino.nome
    eval(funcao_destino2)(destino.ip,int(lsp.id_vlan),
int(lsp.porta_destino),destino.senha_snmp)#Retirando a
 vlan da porta
    funcao_destino3 = 'desligar_porta_'+ tipo_destino.nome
    eval(funcao_destino3)(destino.ip,int(lsp.porta_destino),
destino.senha_snmp)#Desligando a porta de destino
```

```
    lsps = []
    for lista_lsp in lsp_existentes:#Retirando a vlan das
portas trunks da lista de vlans existentes
        if not lista_lsp.id_vlan == lsp.id_vlan:
            lsps.append(int(lista_lsp.id_vlan))#O objeto do
 django guarda os valores como long int e por isso é
 necessário a conversão novamente para int

    for nohs in nohs_existentes:
        porta_trunks = PortaTrunk.objects.filter(noh=nohs)
        for porta_trunk in porta_trunks:
            funcao= 'vlan_trunk_'+ nohs.tipo_noh.nome
 #Primeiro chama-se o aquivo e depois a funcao que sempre termina no Tipo do Noh
            eval(funcao)(nohs.ip,lsps,int(porta_trunk.numero),
nohs.senha_snmp)

    lsp.estado='AE'#Colocando no estado ativo
    lsp.save()

    return HttpResponseRedirect('/')

@login_required
def remover_lsp(request, lsp):
    lsp = Lsp.objects.get(id_vlan=lsp)#Pega o objeto lsp
    lsp.delete()

    return HttpResponseRedirect('/')
```

O arquivo abaixo é o de configurações de urls da aplicação:

```
 # -*- coding: utf-8 -*-
from django.conf.urls.defaults import *

urlpatterns = patterns('teste.views',
 url(r'^inserir-noh/$', 'inserir_noh',
 {}, name='inserir_noh'),
 url(r'^remover-noh/(?P<noh>[\d.]+)/$',
 'remover_noh', {}, name='remover_noh'),
 url(r'^ver-noh/(?P<noh>[\d.]+)/$',
 'ver_noh', {}, name='ver_noh'),
 url(r'^inserir-porta/(?P<noh>[\d.]+)/$',
 'inserir_porta', {}, name='inserir_porta'),
 url(r'^inserir-porta-trunk/(?P<noh>[\d.]+)/$',
 'inserir_porta_trunk', {}, name='inserir_porta_trunk'),
 url(r'^remover-portas-trunk/(?P<noh>[\d.]+)/$',
 'remover_portas_trunk', {}, name='remover_portas_trunk'),
 url(r'^inserir-lsp/$', 'inserir_lsp',
 {}, name='inserir_lsp'),
 url(r'^ver-caminho/(?P<lsp>[\d.]+)/$',
 'ver_caminho', {}, name='ver_caminho'),
 url(r'^inserir-caminho/(?P<lsp>[\d.]+)/$',
 'inserir_caminho', {}, name='inserir_caminho'),
 url(r'^ativar-lsp/(?P<lsp>[\d.]+)/$',
 'ativar_lsp', {}, name='ativar_lsp'),
 url(r'^desativar-lsp/(?P<lsp>[\d.]+)/$',
 'desativar_lsp', {}, name='desativar_lsp'),
```

```
 url(r'^desativar-lsp-agendado/(?P<lsp>[\d.]+)/$',
 'desativar_lsp_agendado', {}, name='desativar_lsp_agendado'),
 url(r'^remover-lsp/(?P<lsp>[\d.]+)/$',
 'remover_lsp', {}, name='remover_lsp'),
)
```

D.5 Configuração de Estilo

O arquivo abaixo é o de definição de estilo para apresentação no browser:

```
*{
  padding:0;
  margin:0;
}
body{
  background: #fff; /*#e3e3e3;*/
}
a{
  color: #2c333b;
}
#acesso{
  margin: 0 auto;
  margin-top: 30px;
  width: 280px;
  height: 200px;
}
#acesso input.btn{
  float: right;
```

```
}
#usuario{
  width: 100%;
  height: 30px;
  background: #c9d5e3;
}
#usuario #mensagem{
  float:right;
  margin-right: 30px;
}
#usuario p{
  margin: 10px;
  display: inline;
}
#breadcrumb{
  width: auto;
  height: 25px;
  font-size: 13px;
  background: #7ac3a6;
}
table{
  min-width: 600px;
  margin-left: 10px;
  margin-top: 10px;
}
table td{
  padding:5px;
```

```
}
table th{
  background: #c9d5e3;
}
table tr.branco{
  background: #bec6c9;
}
table tr.cinza{
  background: #e3e3e3;
}
```