UNIVERSIDADE FEDERAL DO RIO GRANDE DO SUL
INSTITUTO DE INFORMÁTICA
CURSO DE CIÊNCIA DA COMPUTAÇÃO

DANIEL DEL SENT SOARES

SiViCS: Um Sistema para Visualização e Coleta de Switches – Proposta de ferramenta integrada ao Sistema de Gerência de Redes da UFRGS

Trabalho de Graduação.

Prof. Dr. Lisandro Zambenedetti Granville
Orientador

Prof. Dra. Renata de Matos Galante
Co-orientadora

Porto Alegre, julho de 2012.

AGRADECIMENTOS

Agradeço a todos os envolvidos durante o desenvolvimento deste trabalho, incluindo meus professores orientadores e aos funcionários do Departamento de Redes e Suporte do Centro de Processamento de Dados da Universidade Federal do Rio Grande do Sul envolvidos.

SUMÁRIO

LISTA DE ABREVIATURAS E SIGLAS

CPD	Centro de Processamento de Dados
CSS	Cascading Style Sheets
ER	Entidade-Relacional
HTML	Hypertext Markup Language
IP	Internet Protocol
IPAM	IP Address Management
JVM	Java Virtual Machine
MIB	Management Information Base
NAC	Network Access Control
OID	Object Identifier
PHP	PHP: Hypertext Preprocessor
SABRE	Sistema Aberto de Registro de Estações
SGBD	Sistema de Gerência de Banco de Dados
SNMP	Simple Network Management Protocol
UFRGS	Universidade Federal do Rio Grande do Sul
XML	Extensible Markup Language

LISTA DE FIGURAS

LISTA DE TABELAS

RESUMO

O crescimento das redes de computadores faz com que seja cada vez mais necessário o uso de sistemas de gerência de redes mais completos para a administração de dispositivos de rede. Esses sistemas buscam na coleta e uso de informações acessíveis nos dispositivos de rede um melhor funcionamento e uma melhor gerência da rede. A UFRGS não é diferente, e desenvolveu seu sistema de gerência de redes, agregando mais funcionalidades ao longo dos últimos anos.

Esse trabalho propõe uma ferramenta para a coleta e armazenamento de dados de dispositivos de rede e uma interface para visualização de grafos gerados a partir dos dados coletados. Os dispositivos selecionados para essa coleta são os switches da UFRGS, armazenando os dispositivos ligados a cada porta de cada switch da UFRGS. Ao decorrer do trabalho serão apresentados os conceitos necessários para o entendimento da ferramenta, junto com a apresentação das suas três camadas: um coletor de dados escrito em PHP, uma base de dados temporal e uma interface web para visualização de grafos.

Palavras-Chave: Redes, coleta SNMP, switches, base de dados temporais, Prefuse.

SyViCS - A System for Visualization and Collect of Switches

ABSTRACT

The growth of computer networks makes it increasingly necessary to use network management systems for more complete management of network devices. These systems seek in the collect and use of accessible information of network devices for improved performance and better network management. UFRGS is no different, and developed network management system, adding more features over the last years.

This paper proposes a tool for collect and store data from network devices and an interface to view graphs generated from the data collected. The devices selected for this collect are the switches of UFRGS, storing the devices connected to each port of each switch of UFRGS. Along the paper we present concepts necessary for understanding the tool, along with the presentation of its three layers: a data collector written in PHP, a temporal database and a web interface for viewing graphs.

Keywords: Networks, SNMP Collect, Switches, Temporal Databases, Prefuse.

1 INTRODUÇAO

As redes de computadores institucionais - como as redes de universidades e órgãos públicos - têm se tornado cada vez maiores e mais complexas. Essa complexidade pode se apresentar de diversas formas: redes extremamente grandes que não foram dimensionadas da forma correta, uma enorme heterogeneidade de tipos de dispositivos, grandes números de incidentes de segurança, etc.

Uma forma de manter uma melhor organização das redes institucionais são os sistemas de gerência de redes. Existem diversos sistemas comerciais, como, por exemplo, Men and Mice[1] e BT Diamond IP[2], que oferecem ferramentas para alocação de IPs, acompanhamento da segmentação dos registros, controle de incidentes de segurança, entre outros. Esses sistemas são denominados *IPAM*[3], e alguns também apresentam as características de *NAC*[4], oferecendo ferramentas para controle de acesso a rede. O sistema *Men and Mice* é amplamente conhecido por ser um dos maiores e mais confiável sistemas de gerência de rede.

Uma funcionalidade dificilmente encontrada nos sistemas de gerência de redes é a visualização topológica da rede. Por se tratar de dados que precisam ser coletados em um processo complexo e pesado, poucos são os sistemas se preocupam em fornecer uma visualização gráfica da rede e de suas mudanças ao longo do tempo. Perguntas como *'Por onde esteve esta máquina antes de me causar problemas aqui?'* ou *'Quais dos meus switches estão com overhead de tráfego?'* podem ser respondidas com uma coleta de dados temporais nos dispositivos de uma rede.

Dentro dos diversos casos da complexidade de redes, podemos citar a própria Universidade Federal do Rio Grande do Sul. Hoje, a UFRGS[5] conta com 12 mil máquinas espalhadas em todos os seus campi, incluindo estações de trabalho, dispositivos de rede e laboratórios de computação.

A UFRGS também desenvolve seu próprio sistema de gerência de redes desde 2008, de uma forma incremental até os dias atuais, e, seguindo esse processo incremental, iniciou-se em 2011 o desenvolvimento de um módulo para coleta de dados de switches.

1 www.menandmice.com

2 btdiamondip.com

3 IP Address Management

4 Network Access Control

5 www.ufrgs.br

O objetivo desse trabalho de conclusão é apresentar uma ferramenta de coleta e visualização de topologias de switches. Neste trabalho será apresentada uma proposta de coleta de dados temporais dos switches de uma rede, junto com uma ferramenta para a visualização dos dados coletados. Devido a complexidade do trabalho, será utilizado como caso de uso a rede da UFRGS e como base para a ferramenta o sistema de gerência de redes desenvolvida e utilizada pela UFRGS. Mesmo sendo focada na rede da UFRGS, a ideia de coleta de switches aqui apresentada pode ser expandida para outras redes, e as ferramentas aqui propostas podem ser adaptadas às estruturas de outros sistemas de gerência de redes, inclusive para fazer a coleta de dispositivos diferentes de switches.

O restante do trabalho é organizado da seguinte forma: no capítulo 2 será apresentada uma revisão de conceitos que são necessários para a compreensão da solução apresentada. No capítulo 3 a ferramenta será apresentada, explicando o funcionamento da coleta temporal dos switches da UFRGS, bem como a ferramenta para visualização gráfica desses dados. Ainda neste capítulo, serão apresentados alguns resultados do trabalho. No capítulo 4 será feita uma análise de algumas decisões tomadas durante o desenvolvimento do projeto. Por fim, no capítulo 5, será feita uma revisão e avaliação sobre os resultados do trabalho, concluindo quais metas foram atingidas durante o desenvolvimento do trabalho.

2 FUNDAMENTAÇÃO TEÓRICA

O objetivo deste capítulo é rever - ou apresentar - alguns conceitos que serão utilizados no trabalho, e que serão necessários para o entendimento do trabalho.

Além disso, como já foi citado, por se tratar de um trabalho que abrange três grandes áreas de atuação distintas, o foco da solução foi dado na rede da UFRGS. Por esse motivo, é necessário explicar a organização de rede da UFRGS, bem como observar o funcionamento do sistema que abriga os registros de dispositivos de rede da UFRGS.

2.1 Banco de dados

Um banco de dados é por definição, uma coleção de dados inter-relacionados, representando informações sobre um domínio específico. Na prática, qualquer coleção de dados que se relacione de forma a criar uma imagem de um domínio é um banco de dados (ELMASRI, 1999).

Os bancos de dados que não apresentam características temporais são chamados de *bancos de dados instantâneos.* Eles não apresentam nenhuma funcionalidade temporal e são a maioria entre os bancos convencionais. Qualquer alteração necessária sobre os dados neste tipo de banco de dados é feita sobre os dados atualmente armazenados, mantendo apenas o estado atual dos dados e perdendo a informação do passado a cada alteração.

A Figura 2.1 mostra um exemplo desse tipo de banco de dados, onde somente o presente é acessível para consultas.

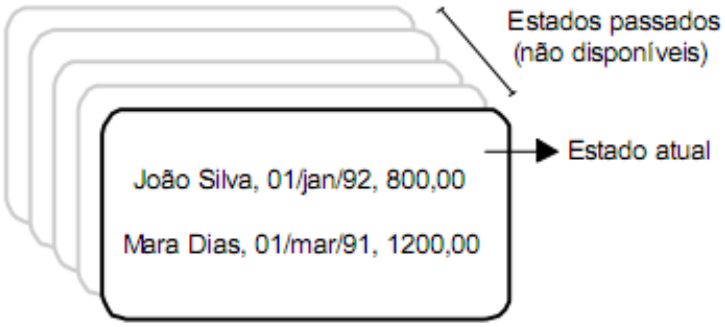

Figura 2.1: Banco de dado instantâneo (EDELWEISS, 1998)

Com a quantidade cada vez maior de dados que são armazenados em aplicações modernas, a necessidade de armazenar a informação histórica dos dados aumentou. Com o aumento da capacidade de armazenamento em todos os tipos de dispositivos, surgiu a possibilidade de armazenar mais informações, incluindo o *histórico* dos dados e permitindo assim o acesso a estados anteriores e futuros da informação.

2.1.1 Bancos de dados Temporais

Bancos de dados temporais se caracterizam por conter uma informação temporal ligada ao dado armazenado de forma a manter todos os estados - presente, passado e futuro - de uma informação, registrando sua evolução no tempo. Dentro dos principais estudos de bancos de dados temporais, são encontrados três tipos principais: bancos de dados de tempo de transação, bancos de dados de tempo de validade e bancos de dados bitemporais (EDELWEISS, 1998).

Os **bancos de dados de tempo de transação** vinculam ao dado armazenado um tempo gerado pelo SGBD - ou pela aplicação responsável pela criação/alteração do dado - que não necessariamente reflete a validade do dado no mundo real, ou seja, o dado possui uma informação temporal que pode ser aproximada ou fictícia. A Figura 2.2 apresenta um exemplo de banco de dados de tempo de transação onde cada estado do dado está vinculado o tempo em que ele foi válido no banco, mas não necessariamente com o tempo que ele foi válido no mundo. As informações temporais nos bancos de dados de tempo de transação representam o instante em que a informação foi armazenada no banco de dados, isto é, o tempo do sistema.

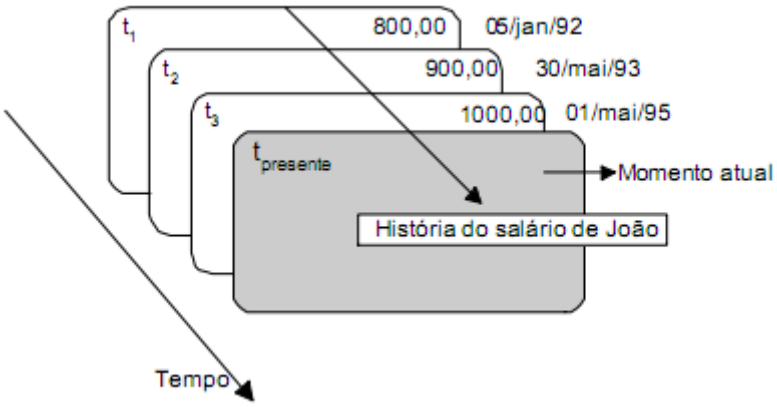

Figura 2.2: Banco de dados de tempo de transação (EDELWEISS, 1998)

Já os **bancos de dados de tempo de validade** atrelam ao dado uma informação temporal que reflete fielmente a validade do dado no mundo real. Dessa forma, é garantido que a vida do dado armazenado é reflexo da validade dele no mundo. Na Figura 2.3 a seguir, temos um exemplo para a mesma situação da Figura 2.2, porém agora a informação temporal é a data onde o dado existiu no mundo real.

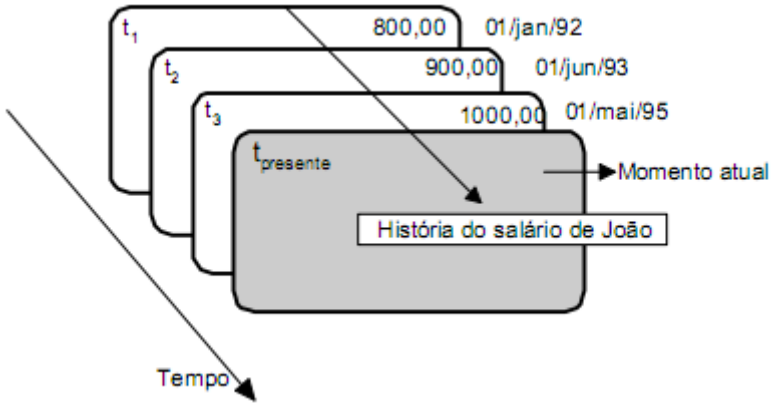

Figura 2.3: Banco de dados de tempo de validade (EDELWEISS, 1998)

Por fim, os **bancos de dados bitemporais** possuem ambas as características temporais: além de possuir um tempo gerado pelo sistema, eles também possuem um dado temporal que reflete a realidade da informação armazenada. Na Figura 2.4 a seguir, é exemplificado um banco de dados bitemporal com as características temporais dos dois bancos citados anteriormente.

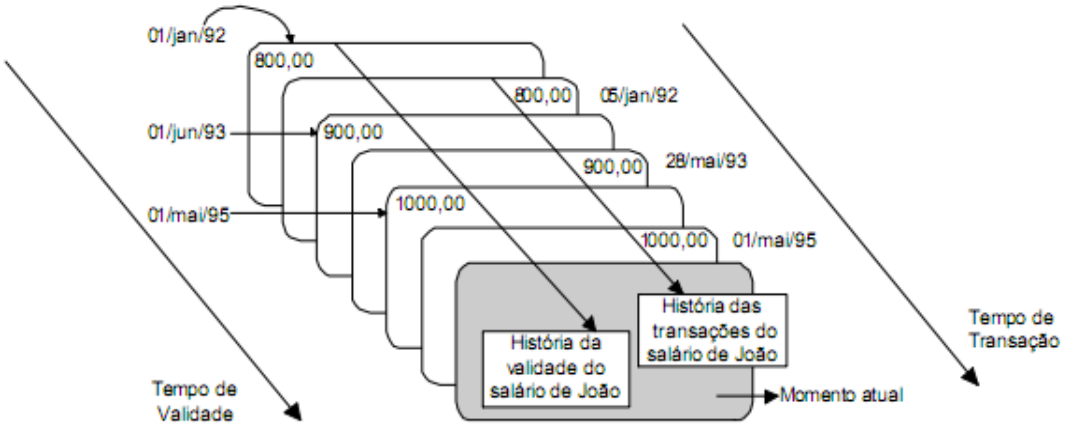

Figura 2.4: Banco de dados bitemporal (EDELWEISS, 1998)

Outro exemplo que pode ser aplicado aos três tipos de bancos de dados temporais utilizando o meio acadêmico é o de um registro de vínculo de aluno. Se armazenado em um *banco de dados de tempo de transação*, a validade desse aluno seria marcada pelo seu registro por um sistema acadêmico e se encerraria por uma atualização nesse sistema, podendo assim não ser reflexo do dado real, mas sim do instante em que a base e dados foi atualizada com os dados do aluno em questão. Se armazenado em um *banco de dados de tempo de validade*, as datas de início e fim devem ser informadas pelo usuário, de forma a refletir o tempo real em que o aluno possuiu um vínculo. Caso esse registro fosse armazenado em um *banco de dados bitemporal*, ambos os tempos deveriam existir: tanto a data gerada pelo uso do sistema, quanto uma informação temporal dada por um usuário.

Para o desenvolvimento da ferramenta que será descrita neste trabalho, o banco de dados utilizado é o de ***tempo de transação***. Esse tipo de base temporal mantém a temporalidade sem necessariamente refletir a validade do dado no mundo real, algo que é necessário para a ferramenta, sendo essa escolha analisada mais adiante.

2.1.2 Trabalhos estudados sobre bancos de dados temporais

Além do trabalho citado na definição de bases temporais, que apresentou boa parte dos conceitos temporais utilizados na ferramenta, outro trabalho analisado foi o de HUBLER (1999), que apresenta uma proposta de implementação de banco de dados temporais utilizando SGBDs convencionais, abordando tanto os SGBDs orientados a objetos, quanto os SGBDs relacionais, que são mais frequentemente encontrados no mercado e tem como exemplo o P**ostgre,** SGBD utilizado neste trabalho.

2.2 Redes

Dentro da área de redes de computadores, vários conceitos que não são tão conhecidos precisam ser vistos para um entendimento do resultado final deste trabalho. Dentre esses conceitos estão os conceitos de switch, de topologia de rede, do SNMP e também uma série de conceitos importantes para definir o que é o sistema de gerência de redes da UFRGS, sistema esse que serviu como base para o desenvolvimento da ferramenta deste trabalho de conclusão.

2.2.1 Switches

Antes de explicar o conceito de um switch, dois outros conceitos devem ser explicados: domínio de colisão e pontes.

Domínio de colisão é uma região *lógica* dentro de uma subrede onde pacotes de dados podem colidir durante sua transmissão (CARISSIMI, 1999). Cada colisão de pacote gera uma retransmissão de dados, diminuindo assim a eficiência da rede.

Considerando uma pequena rede constituída de 5 computadores dentro de uma sala sem acesso a redes externas, temos um domínio de colisão constituído por apenas esses 5 computadores já que pacotes dentro dessa rede podem apenas colidir com outros pacotes que sejam transmitidos por um dos 5 computadores. Ao conectar essa pequena rede a uma rede maior, os seus domínios de colisão são somados, gerando um domínio de colisão maior que é igual a soma de todos os domínios de colisões das redes separadas.

Com o crescimento de uma rede, o domínio de colisão desta cresce de forma a comprometer o funcionamento da mesma. As **pontes** são dispositivos que tem o intuito de diminuir o problema do crescimento do domínio de colisão (CARISSIMI, 1999). As pontes (*ou bridges)* são dispositivos que conectam duas redes *físicas* sem unificar as redes *lógicas*, mantendo assim dois domínios de colisão separados. A unificação *lógica* das redes só ocorre quando um pacote precisa ser transmitido de uma rede para outra, ou quando um pacote é transmitido em *broadcast*, isto é, para todos os dispositivos que fazem parte da rede completa.

Um switch (*ou comutador*) nada mais é do que um conjunto de pontes que conecta **N** redes físicas sem necessariamente conectar seus domínios de colisão (CARISSIMI, 1999). Um switch é basicamente constituído por um conjunto de portas (normalmente 24 ou 48) e por um microprocessador responsável por fazer a análise dos pacotes e fazer o bridging entre os domínios, fazendo a retransmissão do pacote para a porta correspondente. A análise dos pacotes é feita através do endereço MAC que identifica de forma única o pacote. O switch mantém informações de endereços aprendidos, e caso não possua a informação necessária para retransmissão do pacote, utiliza protocolos específicos para aprendizagem de endereço. A Figura 2.5 mostra uma típica organização

de rede, onde os switches são utilizados para conectar regiões menores, mantendo os domínios de colisão separados.

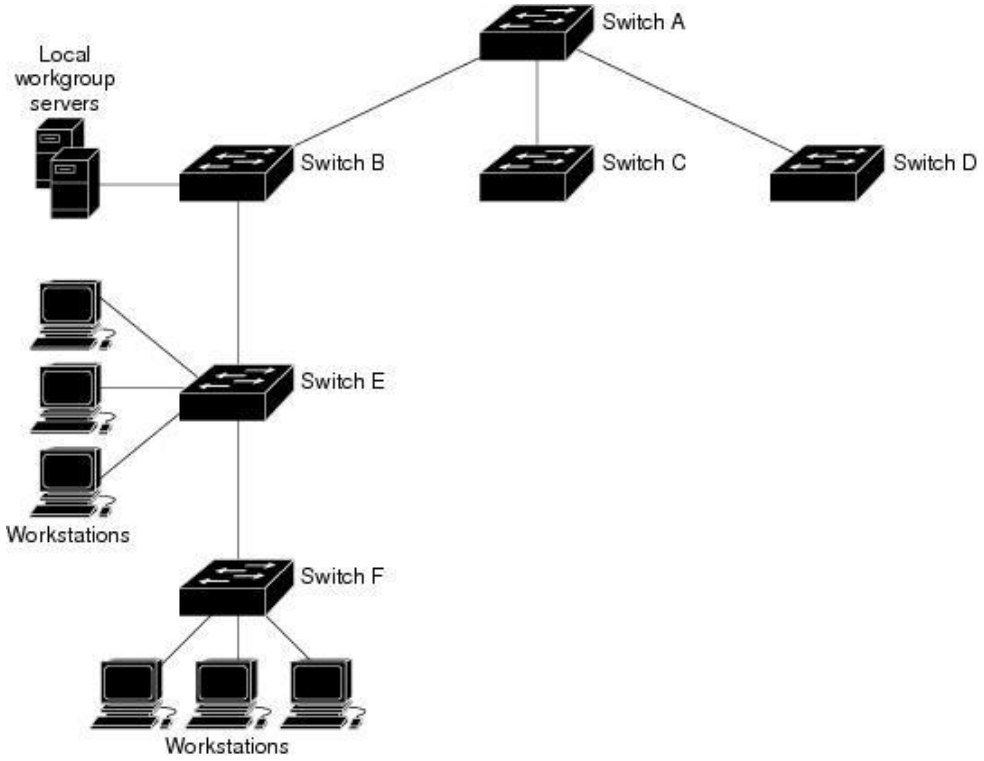

Figura 2.5: Estrutura de rede com switches

Na Figura 2.5, o switch A faz o bridging de todos os pacotes que precisam ser transmitidos do domínio do switch D para os domínios de outros switches, sem afetar a transmissão dentro de cada domínio. Por não ser objetivo deste trabalho, não será discutido em mais detalhes o funcionamento interno dos switches.

2.2.2 Topologia de Redes

Conceitualmente, a topologia de uma rede descreve a organização de uma rede de computadores. Uma topologia de rede pode representar a organização do tráfego de dados dentro da rede, sendo assim uma topologia *lógica*, ou pode representar a organização das conexões dos dispositivos dentro da rede, recebendo o nome de topologia *física* (CARISSIMI, 1999).

Física ou lógica, dentre os tipos de topologias destacam-se 4 tipos básicos que podem formar redes mais complexas. Esses tipos são descritos a seguir e são visíveis na Figura 2.6:

- **Barramento**: neste tipo de topologia todos os dispositivos de rede estão ligados em um único barramento de dados, que é compartilhado por todos ao se executar qualquer transmissão de dados. Por esse motivo, somente um dispositivo pode usar o barramento por vez, e todos os dispositivos recebem todos os dados enviados ao barramento, descartando informações que não são destinadas a eles.
- **Anel**: nesta topologia os dispositivos são ligados entre si em pares formando um circuito fechado em forma de anel. Todo tráfego flui em apenas uma direção, e

os dados são transmitidos de nó em nó, o que pode tornar a transmissão onerosa em anéis muito extensos.

- **Estrela**: numa topologia em estrela os dispositivos estão todos ligados a um dispositivo central chamado *concentrador*. Este dispositivo é responsável por receber as transmissões e enviar a todos os nodos da rede, de forma semelhante à transmissão em um barramento.
- **Árvore**: em uma topologia em árvore geralmente encontramos uma hierarquia de dispositivos, geralmente com funções diferentes a cada nível. Assim, cada nível da árvore normalmente apresenta um tipo diferente de dispositivo.

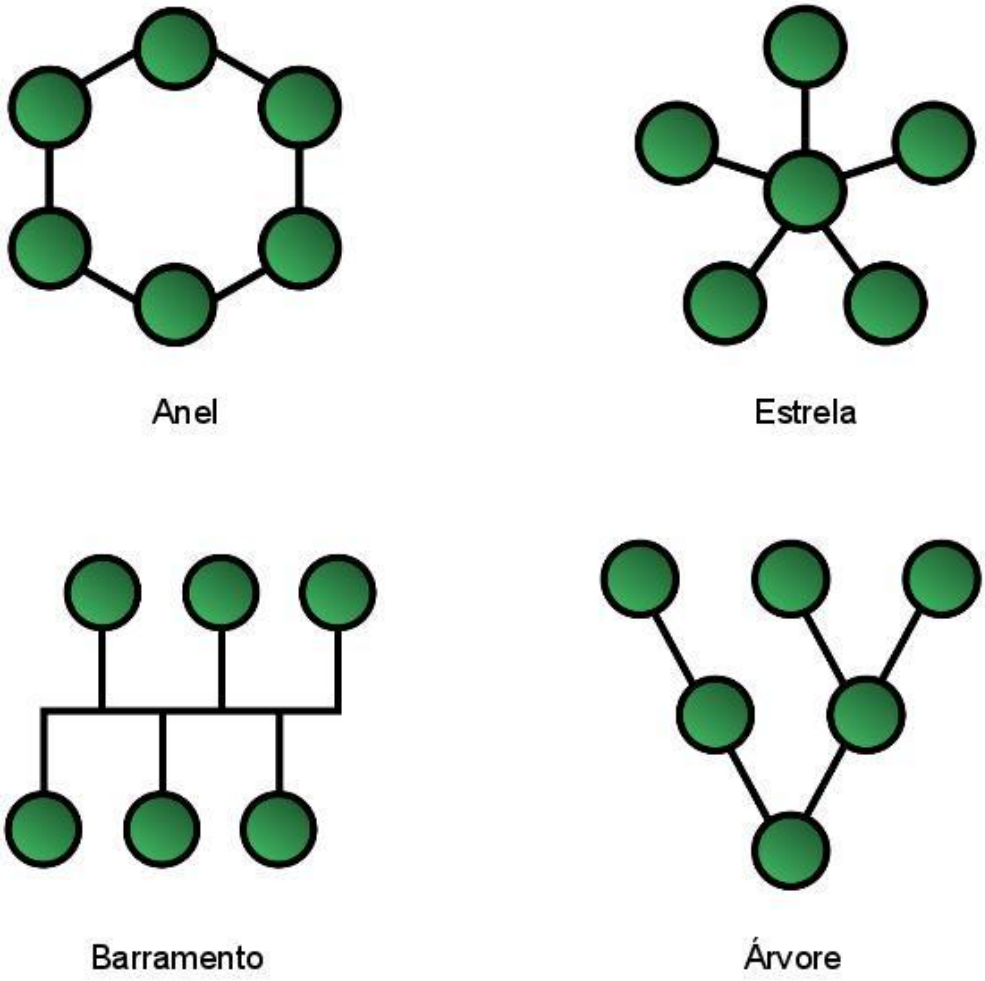

Figura 2.6: Topologias de rede

Neste trabalho, o que é chamado de *topologia da rede* não se encaixa totalmente em nenhum dos tipos básicos vistos. O que será visualizado pelo usuário na interface final é uma organização hierárquica da informação coletada dos switches e das entidades lógicas utilizadas pelo sistema de gerência da UFRGS, como o *bloco* e o *registro de IP*, que serão mais bem descritas adiante.

Assim, a visualização da ferramenta descrita no trabalho exibe uma topologia de *árvore*. Porém, isso não representa necessariamente a topologia física dos dispositivos dentro da UFRGS.

SNMP

O *Simple Network Management Protocol* (SNMP), é um protocolo padrão para gerência de dispositivos em redes IP (STALLING, 1999). Entre os diversos dispositivos que suportam SNMP estão roteadores, switches, servidores, estações de trabalho e impressoras.

O uso mais comum do SNMP é no monitoramento de dispositivos ligados à rede, de forma a garantir seu bom funcionamento e alertar sobre falhas os administradores da rede. O SNMP acessa os dados gerenciados através de duas primitivas básicas: GET e SET. Nem todos os dispositivos aceitam a primitiva SET, que permite a alteração de dados.

Neste uso mais comum, um ou mais dispositivos recebem a tarefa de monitorar e gerenciar um grupo de dispositivos dentro de uma rede. Os dispositivos que recebem essa tarefa são chamados de *gerentes SNMP*. Cada *dispositivo gerenciado*, ou seja, que é monitorado por um *gerente*, executa uma aplicação que é responsável por receber requisições de GET ou SET, e retornar os dados necessários para atender a solicitação. Essa aplicação é chamada de *agente*. As variáveis acessadas pelo agente estão organizadas hierarquicamente em forma de árvore e são chamadas de MIBs (*Managament Information Bases*). Em resumo, uma rede gerenciada por SNMP contém os seguintes componentes:

- **Dispositivos gerenciados:** qualquer tipo de dispositivo que faça parte da rede, podendo ser um roteador, switch, servidor, telefone IP, estação de trabalho, etc. As informações que podem estar sendo armazenadas nas variáveis do dispositivo podem ser específicas por tipo de dispositivo, ou informações que são comuns a vários tipos;
- **Agentes**: software que roda em cada dispositivo gerenciado e é responsável por acessar as variáveis do dispositivo para atender às solicitações de GET e SET. Um agente tem conhecimento local das variáveis armazenadas por seu dispositivo, e por esse motivo pode traduzir requisições de dados que existem nas variáveis de seu dispositivo;
- **Gerentes:** máquinas que monitoram os dispositivos gerenciados da rede;
- **NMS** (*Network Managament System*): software executado nos gerentes que monitora os dispositivos gerenciados. Como a quantidade de dispositivos gerenciados pode ser enorme, bem como a variedade dos dados existentes em cada dispositivo, um ou mais NMS diferentes podem estar executando simultaneamente em cada dispositivo gerenciador.

Na Figura 2.7 temos um caso de uso com uma rede gerenciada mista, com gerentes e dispositivos gerenciados.

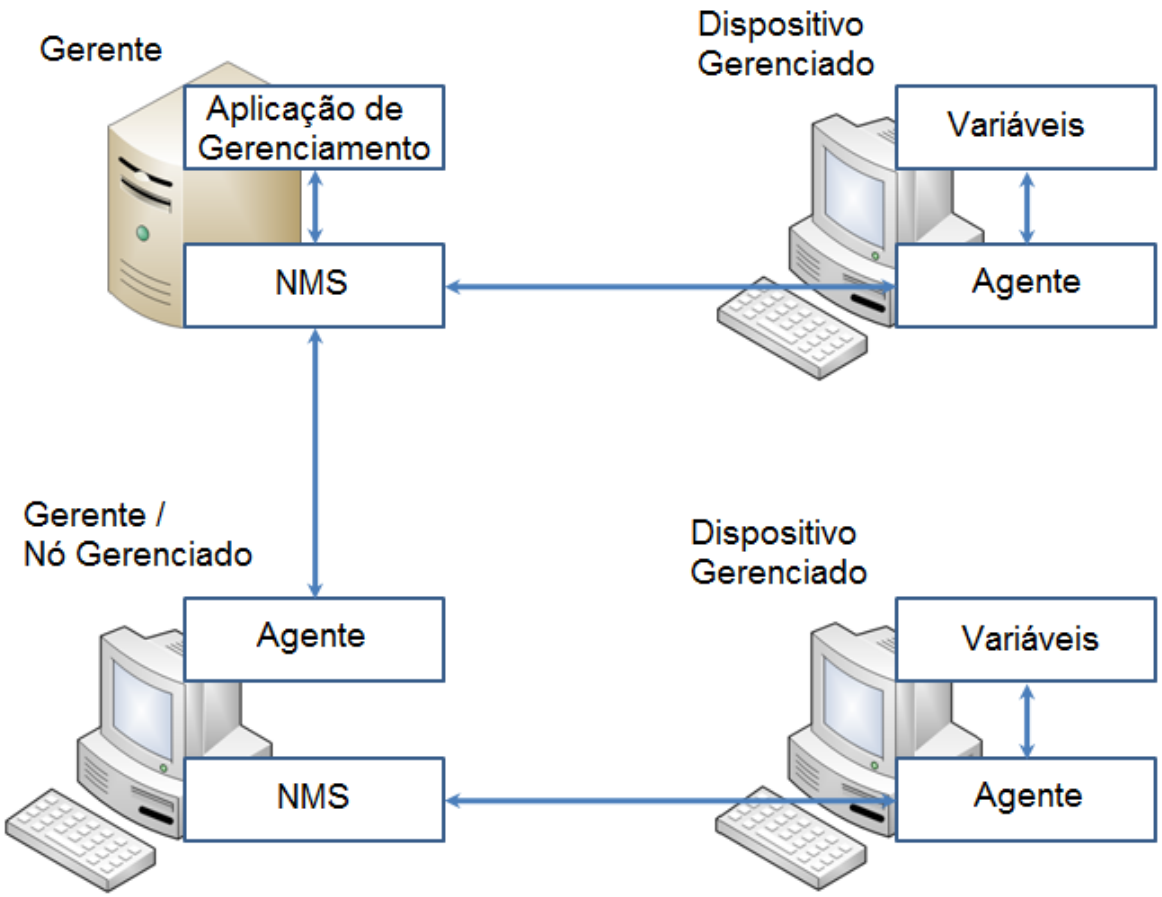

Figura 2.7: Exemplo de funcionamento de rede gerenciada

O protocolo SNMP opera na camada de aplicação (camada 7 do modelo OSI). Os agentes responsáveis por receber e responder requisições SNMP escutam a porta 161 UDP. A organização de cada pacote SNMP não será abordada nessa revisão.

2.2.3 MIBs

Managament Information Base (MIB) é uma base de dados composta pelas variáveis que um dispositivo oferece para gerenciamento, sendo que cada dispositivo de rede possui sua própria MIB. Uma MIB é organizada de forma hierárquica (árvore), onde cada nodo dessa árvore é um objeto que pode ser identificado de forma única por uma string de números separados por pontos, chamada de ***oid***, ou *object idenfier* (identificador de objeto).

Apesar de cada dispositivo possuir sua própria MIB, existem padrões que definem conjuntos de variáveis, chamados de *módulos*. Os módulos podem representar conjuntos de variáveis definidos para tipos específicos de dispositivo (como o módulo utilizado no trabalho, que é válido apenas para *bridges*) ou conjuntos de variáveis definidos e implementados para dispositivos de uma determinada organização (o que acontece com dispositivos de redes de fabricantes renomadas, como a CISCO).

Cada nodo armazena um *objeto gerenciado*, que é um objeto que contém informações sobre um dispositivo gerenciado. Cada objeto gerenciado pode ser constituído de um ou mais objetos de um dos seguintes tipos:

- *Escalar*: o objeto define uma instância única de um objeto, ou seja, ele armazena seu próprio valor.
- *Tabular*: o objeto define múltiplas instâncias de um determinado objeto, que por sua vez, pode ser tabular ou escalar.

As tabelas e atributos da MIB utilizadas na ferramenta serão vistas com mais detalhes adiante, na descrição completa da ferramenta.

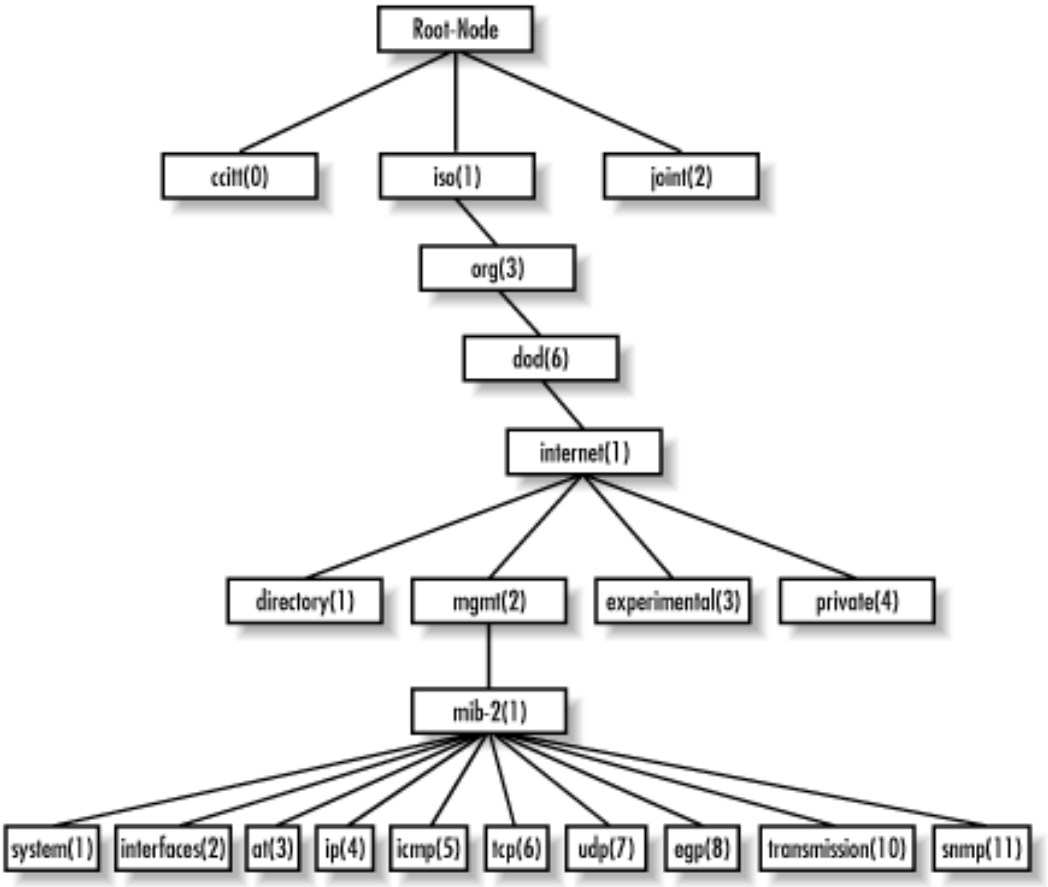

Figura 2.8: Parte da organização das MIBs

Na Figura 2.8 é possível ver parte da estrutura da MIB. Os níveis mais altos possuem informações padrão de organização, sendo que as informações de gerência se encontram abaixo do quinto nível, prefixados pelo OID **.1.3.6.1.2**. Por exemplo, a MIB interfaces (*.1.3.6.1.2.1.2*) armazena a informações das interfaces de um dispositivo, sendo assim padrão e implementada por todo dispositivo de rede.

2.2.4 Trabalhos estudados sobre coleta SNMP

O primeiro trabalho relacionado à coleta de dados utilizando SNMP foi o de BEJERANO (2003), que apresenta um algoritmo para descoberta da topologia de uma rede heterogênea. O algoritmo parte de um esqueleto base composto por alguns dos switches/roteadores que fazem parte da topologia, expande o esqueleto da topologia até um ponto em que ele não cresça mais e parte para uma etapa de refinamento do esqueleto montado. No trabalho, o algoritmo não é aplicado com o uso do protocolo SNMP, mas as provas formais comprovam seu funcionamento.

O segundo trabalho desta área foi o de BREITBART (2004)**,** que propõe uma ferramenta para coleta de dados de topologia a partir do algoritmo citado anteriormente. A descoberta é feita a partir de um roteador semente. Dele são descobertos os vizinhos e o algoritmo é aplicado recursivamente até o ponto em que a topologia não aumenta mais. A ferramenta proposta por ele serviu como base para os elementos da BRIDGE MIB que vieram a ser utilizados na construção da coleta de dados deste trabalho, o que será apresentado nos capítulos posteriores.

2.3 Sistema de gerência da UFRGS

Em 2008 a UFRGS iniciou o desenvolvimento de um *IPAM* (IP Address Manager), com uma interface web para gerência da rede e alteração de diversas configurações sem uma interação direta com os dispositivos, bem como um NAC (*Network Access Control*) com uma interface de registro de dispositivos para o usuário final. O desenvolvimento desse software se estende até a presente data, de forma incremental com funcionalidades adaptadas de sistemas existentes e outras originais moldadas para suprir as necessidades da UFRGS.

No final de 2010, se iniciou o projeto de conversão do *IPAM* utilizado na UFRGS para uma versão desvinculada da sua estrutura organizacional. Esse sistema recebeu o nome de *SABRE (Sistema ABerto de Registros)* e já foi apresentado no *Workshop de TI das Instituições Federais de Ensino Superior* (WTIIFES) de Santa Catarina em 2011 (MACHADO, 2011). Atualmente ele passa pelos trâmites para o registro de software em meados de 2012.

Como o problema que esse trabalho analisa é extremamente amplo e complexo, e também é uma funcionalidade muito interessante para o sistema de registros da UFRGS, o escopo foi reduzido para a rede da UFRGS. Dessa forma, todo o trabalho é baseado em apresentar uma solução interessante para visualização da topologia da UFRGS.

Por esse motivo, os dois próximos subcapítulos apresentam a organização dos registros de dispositivos de rede dentro da UFRGS, e também apresentam as tabelas da base de dados da UFRGS utilizadas para auxiliar o desenvolvimento da ferramenta.

2.3.1 Organização de rede da UFRGS

A rede da UFRGS é uma rede de **classe B** endereçada por **143.54.0.0**. Isso dá a UFRGS **256^2** IPs para gerenciar. Hoje a UFRGS conta com aproximadamente *12.000* IPs em atividade, sendo 8.000 deles *gerenciados* pelo sistema de registro de estações da UFRGS.

A diferença entre os IPs *gerenciados* e os *não gerenciados* está na configuração da máquina: os registros gerenciados pelo sistema possuem uma configuração semelhante a de computadores domésticos, onde os servidores de DHCP da UFRGS são responsáveis por fornecer um IP válido para o computador.

Dentro das subredes gerenciadas pelo sistema, existem duas formas de conceder um IP válido a um novo dispositivo. A primeira envolve um cadastro manual no sistema, onde cada dado do dispositivo é informado por um gerente, e ao término do cadastro, o dispositivo passa a receber do servidor de DHCP do sistema um IP válido. A segunda forma envolve um cadastro para o qual o usuário é redirecionado ao acessar pela primeira vez a internet com um dispositivo recém conectado na rede. Poucos dados são solicitados, e por esse motivo, esse cadastro é realizado pelo próprio usuário do

dispositivo. Após o término do registro, o dispositivo também passa a receber um IP válido do servidor de DHCP da UFRGS.

A rede da UFRGS é dividida em N **subredes**. Cada subrede pode ser vista como uma rede *classless*, que possui um IP de identificação e uma máscara de subrede livre, permitindo assim a criação de subredes de tamanho 4 até uma subrede de tamanho 65536 que englobaria toda a UFRGS. Cada subrede pode possuir configurações próprias ou herdar as que foram criadas para toda a UFRGS.

Depois das subredes, temos outro nível de configuração: **o bloco**. O bloco é uma faixa de IPs contida dentro de uma subrede, sendo que uma subrede pode conter de 1 a N blocos contíguos. Novamente, aqui temos um nível de configuração, pois os blocos podem herdar as configurações de suas subredes ou ter novas configurações.

Abaixo dos blocos estão os **IPs** que foram registrados de uma das duas formas descritas anteriormente. Um IP sozinho não representa nada dentro do sistema. Para ter significância como registro, ele é atrelado a uma **interface** e a um **dispositivo**, formando assim a *tupla* que identifica um IP dentro da UFRGS. O registro de um *dispositivo* representa o dispositivo material, ou seja: um computador, um roteador, um notebook, um telefone, etc. Já uma *interface* representa uma forma de se conectar a rede, como, por exemplo, uma placa de rede ou um adaptador wireless. Cada IP está vinculado a uma Interface e cada dispositivo pode possuir N interfaces, e o exemplo prático disso é um notebook com 2 IPs: um com fio e outro sem fio.

Entre os dispositivos que fazem parte da UFRGS estão cerca de 300 switches. Dentro do sistema de registro, não existe a informação de quais registros estão ligados a quais switches, e muito menos a informação de hierarquia de switches.

2.3.2 Base de dados da UFRGS

As entidades a seguir são representações das tabelas da base de dados da UFRGS, e não retratam a base real da UFRGS por critérios de anonimação dos dados. Na Figura 2.9, temos o diagrama onde estão as tabelas que armazenam os dispositivos de rede - sem atribuir um IP a eles - e também as tabelas auxiliares utilizadas para especificar todos os dados de um switch. No caso da UFRGS, devido a heterogeneidade dos dispositivos, existe uma tabela auxiliar para modelos de switches, e é nela que se localiza o identificador do objeto da MIB que deve ser consultado no dispositivo.

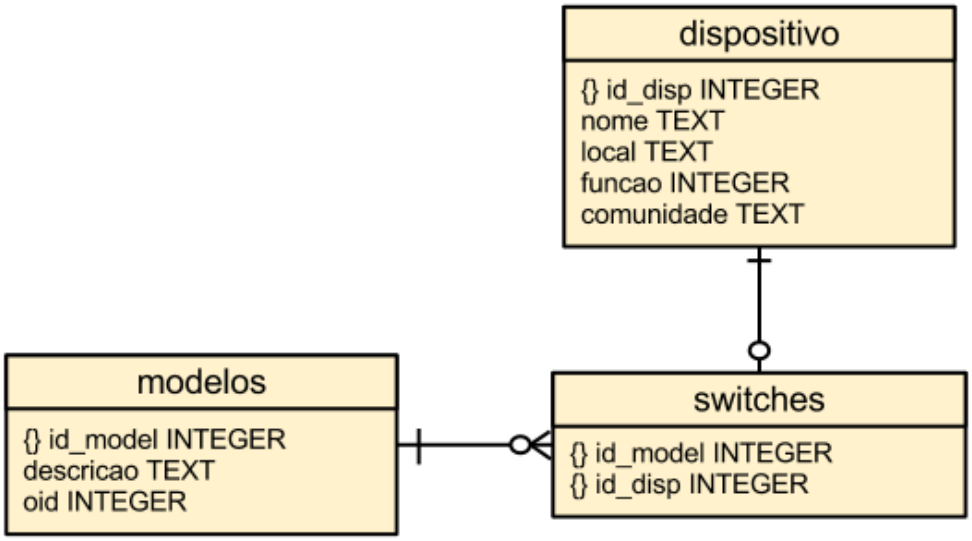

Figura 2.9: Tabelas que fazem parte do sistema de gerência e auxiliam a coleta

No segundo diagrama, na Figura 2.10, estão as tabelas que armazenam de fato os registros da UFRGS. Como dito anteriormente, existem vários níveis para os conjuntos de registros.

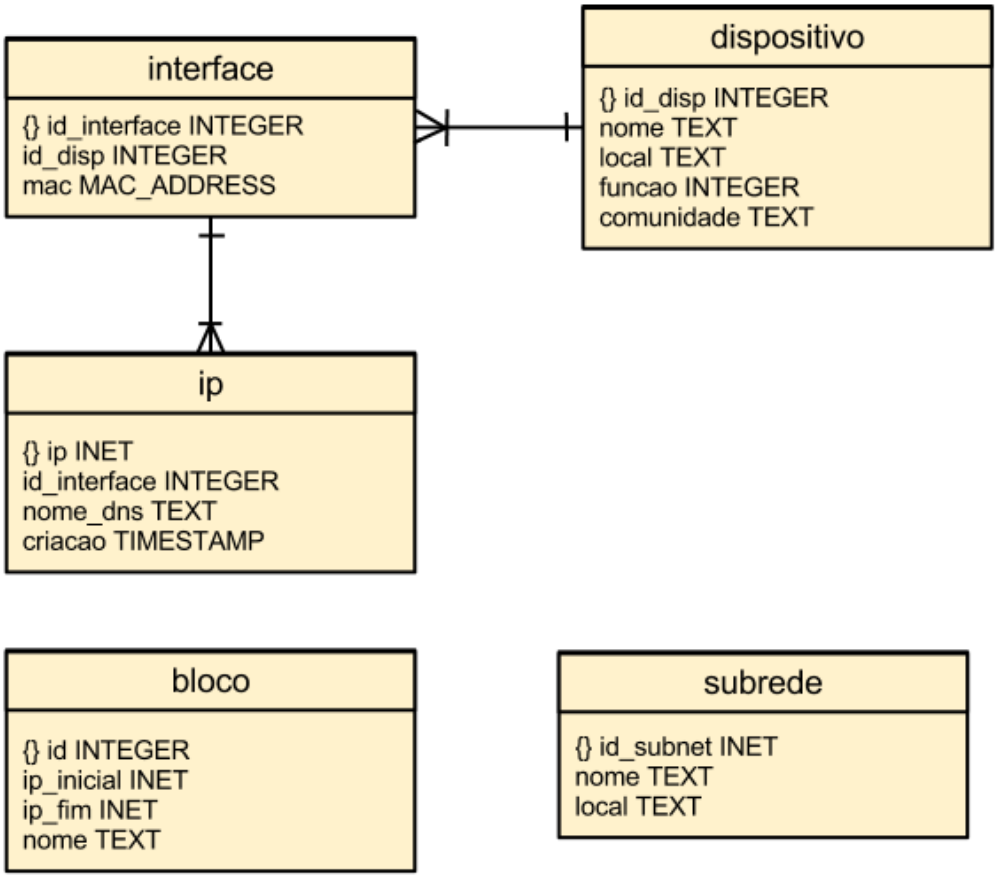

Figura 2.10: Tabelas que armazenam os registros de IPs no sistema da UFRGS

Não existe uma chave que ligue o registro de um *bloco* e valide sua existência dentro de uma *subrede*, sendo assim, toda a validação envolvendo a consistência de dados dessas tabelas é feita dentro do sistema de registros. Abaixo dos *blocos* estão os *registros de IPs*, representados por um trio de registros nas tabelas **ip - interface - dispositivos**.

2.4 Visualização de Grafos

A última parte do trabalho é composta por uma interface gráfica para visualização de grafos. A interface web segue muito do padrão visual definido para ferramentas da UFRGS, porém, ocorreu um grande estudo para a definição do toolkit que seria utilizado para a renderização desse grafo.

2.4.1 Prefuse

O Prefuse é um toolkit extensível para visualização de grafos em Java (HEER, 2005). Ele fornece estruturas otimizadas para visualização de dados em tabelas, grafos e árvores, incluindo uma vasta gama de layouts e técnicas, tais como animações (força e movimento) dos elementos gráficos. Ele é totalmente escrito em Java, utilizando bibliotecas gráficas 2D e é facilmente integrado a aplicações Java que utilizam Swing, ou como applets para web.

Em um primeiro momento, o Prefuse foi escolhido para ser utilizado em uma aplicação Java desktop, porém, após estudos da usabilidade da aplicação e da integração do trabalho com a ferramenta utilizada pela UFRGS, a aplicação foi alterada para uma interface em PHP, com o uso de um applet em Java para visualização de grafos.

No caso da ferramenta desenvolvida neste trabalho, o Prefuse foi escolhido por se tratar de um toolkit capaz de lidar com grandes quantidades de dados e com grandes quantidades de registros. Outras possibilidades existem tanto em Java quanto em HTML 5, porém, o Prefuse foi escolhido por ser o toolkit com mais trabalhos relacionados envolvendo quantidades grandes de nodos.

2.4.2 Java Applets

Applets são elementos HTML que permitem a execução de uma aplicação java em um web browser. Para sua execução, basta que a aplicação tenha sido desenvolvida como applet, e que o cliente possua uma *JVM (Java Virtual Machine)* instalada. Com esses pré-requisitos, o cliente será capaz de executar no browser a aplicação de forma completa.

A sintaxe básica de um applet é a seguinte:

```
<APPLET CODE="JavaApplet.class" WIDTH="200" HEIGHT="50">
   <PARAM NAME="text" VALUE="I am a parameter!">
</APPLET>
```

Como um applet aceita qualquer quantidade de parâmetros, todos os dados para a geração do grafo são passados dessa forma.

O tag **<APPLET>** ainda é suportado, mas ele está em estado depreciado, e para novas versões da aplicação será utilizado o tag **<OBJECT>**, que ainda não é 100% suportado para java applets em **HTML**.

Na aplicação desenvolvida para o trabalho, o uso de applet reduziu a interação do grafo com a base de dados da UFRGS, porém garantiu que qualquer usuário tivesse acesso a aplicação sem a necessidade de um instalador separado, bastando possuir a JVM em sua máquina.

2.4.3 Grafos com Prefuse

O Prefuse é um toolkit poderoso, capaz de renderizar diversos tipos de estruturas de dados gráficas: tabelas, árvores, grafos, gráficos, etc. Para o trabalho, o único funcionamento interessante do Prefuse é a geração de grafos.

O Prefuse tem a capacidade de gerar visualizações gráficas a partir de arquivos XML bem estruturados. Essa capacidade é extremamente interessante quando estamos trabalhando com grafos bem definidos e com aplicações desktop. No caso da ferramenta desenvolvida para o trabalho, a outra possibilidade é mais interessante: especificar a estrutura e os componentes do grafo em tempo de execução, utilizando parâmetros que podem ser passados através do elemento **<APPLET>**.

Para criar um grafo com Prefuse dentro de uma classe java, é necessário criar um elemento do tipo *Graph* e especificar a estrutura dos nodos, ou seja, quais informações serão armazenadas em cada nodo, como no código a seguir:

```
Graph graph = new Graph();
graph.addColumn("type", String.class);
graph.addColumn("title", String.class);
```

Após isso, os nodos são criados com o tipo *Node*, adicionando-se a cada nodo os valores dos atributos especificados na estrutura, como no código a seguir:

```
Node node1 = graph.addNode();
node1.set("type", "ip");
node1.set("title", "100.0.1.1");

Node node2 = graph.addNode();
node2.set("type", "rede");
node2.set("title", "100.0.1.0/24");
```

Tendo os nodos que são ligados por uma aresta criados, é possível adicionar as arestas a estrutura do grafo, com uma chamada do método *addEdge* do grafo, como no código a seguir:

```
graph.addEdge(node1, node2);
```

Esses três trechos de código resumem a funcionalidade básica do prefuse de especificar nodos e arestas, e por consequência, grafos *não-direcionados* de qualquer tipo, bastando estender a estrutura de forma a aceitar **N** nodos e **M** arestas.

Além da funcionalidade básica, o prefuse fornece diversas formas de alterar a aparência dos nodos e arestas em função dos dados armazenados no grafo, bem como animações diferentes para visualização dos grafos. Essas funcionalidades mais poderosas serão vistas rapidamente durante a descrição completa da ferramenta.

3 SIVICS

O objetivo deste capítulo é descrever a proposta e a implementação de uma ferramenta denominada SiViCS - Sistema para Visualização e Coleta de Switches - que tem por objetivo ser uma ferramenta para coleta, armazenamento e visualização da topologia das portas dos switches de uma rede. A proposta e a ferramenta estão definidas para o funcionamento em conjunto com o Sistema de Registros de Estações da UFRGS. A ferramenta SiViCS está dividida em três camadas distintas, que tiveram o seu desenvolvimento de forma independente, mas que são consolidadas no SiViCS de forma integrada, a saber: (i) um script coletor de informações de dispositivos de rede por snmp; (ii) uma base temporal para informações topológicas das portas dos switches; e (iii) uma interface *web* para visualização de grafos topológicos da rede.

Este capítulo está organizado da seguinte forma. Primeiramente, é apresentada uma visão geral do SiViCS Em seguida, cada uma das três camadas da ferramenta SiViCS é explicada detalhadamente. O capítulo é finalizado com uma análise geral dos resultados alcançados.

3.1 Visão Geral

O esquema da Figura 3.1 mostra a organização das três principais camadas existentes na ferramenta. O objetivo principal do SiViCS é ser uma ferramenta para coleta, armazenamento temporal e visualização dos registros ligados a cada porta de um switch, tendo como caso de uso a rede da UFRGS. A primeira camada da ferramenta é constituída por um script coletor definido para ser executado periodicamente em uma máquina coletora dedicada. Ao ser executado, esse script coleta dados dos switches da UFRGS, fazendo a análise de quais dados devem ser atualizados em uma base temporal. A segunda camada é esta base temporal, sendo essa camada o ponto central do SiViCS. Nesta base de dados estão todas as tabelas utilizadas pela ferramenta, bem como as tabelas do sistema de gerência de redes da UFRGS. A terceira camada é formada por uma interface gráfica permite consultas a base temporal, gerando um grafo referente à coleta dos switches.

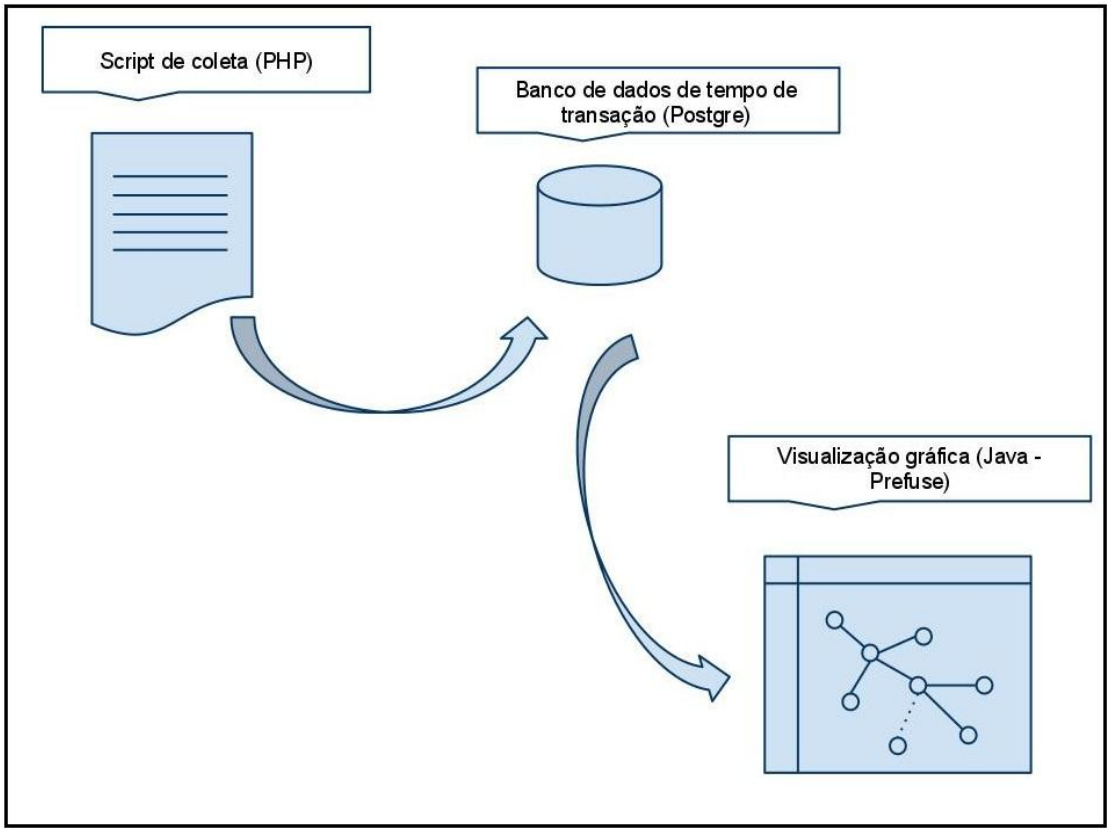

Figura 3.1: Esquema geral da ferramenta de coleta e visualização de redes

Os recursos computacionais utilizados para o desenvolvimento e testes do SiViCS são os seguintes:

- Processador Intel Xeon 5110 [6] @ 1.60 GHz;
- Máquina virtual executando Ubuntu 10.04 [7];
- 1 GB de memória dedicada a máquina virtual;
- Postgre 8.4 [8]
- PHP 5.3.2 [9]

As subseções a seguir descrevem cada uma das camadas detalhadamente.

6 http://ark.intel.com/products/27214/Intel-Xeon-Processor-5110

7 http://releases.ubuntu.com/lucid/

8 http://www.postgresql.org/about/news/1108/

9 http://php.net/releases/5_3_2.php

3.2 Coletor SNMP

A primeira camada da ferramenta é constituída por um script coletor. O objetivo do coletor é executar uma rotina que busca em cada switch coletado os registros encontrados em cada uma de suas portas, além de atualizar o estado anterior da topologia de cada switch.

O script coletor foi escrito utilizando a linguagem **PHP**. Além de possuir uma sintaxe simples e um desempenho relativamente bom, o PHP possui uma abstração na manipulação de arrays que é muito útil durante o processo de coleta de dados. Para muitas aplicações, a abstração de dados do PHP pode ser considerada incorreta ou indesejada, mas, para o desenvolvimento do script, características como fraca tipagem do PHP e índices de qualquer tipo nos arrays são extremamente interessantes. Esta última característica permite, por exemplo, a criação de estruturas de dados com índices de qualquer tipo, incluindo dados que são muitos utilizados em toda extensão da ferramenta: IPs e endereços físicos (*mac address*).

Para executar a coleta, o PHP utiliza o módulo Net-SNMP que fornece todas as funções para acesso as *mibs* de dispositivos de rede. O coletor é executado a cada **30** minutos em uma máquina dedicada a este processamento, varrendo os switches da rede da UFRGS, coletando dados específicos das mibs dos switches. Esse intervalo de tempo foi definido pelo departamento de redes do CPD da UFRGS, para garantir que este processo não sobrecarregasse a rede. A definição inicial de 15 minutos, feita em função do tempo de vida dos dados coletados foi descartada por esse motivo.

Antes de descrever como os dados são coletados (seção 3.2.2), a subseção 3.2.1 a seguir descreve quais são e de onde vêm os dados coletados durante a execução do script.

3.2.1 BRIDGE-MIB

O módulo da MIB utilizado nessa ferramenta é o módulo BRIDGE-MIB, que define as variáveis que podem estar presentes em bridges, e por consequência, em switches. A BRIDGE-MIB é identificada pelo *oid* 1.3.6.1.2.1.17, e este *oid* é caracterizado pela subárvore da Figura 3.2.

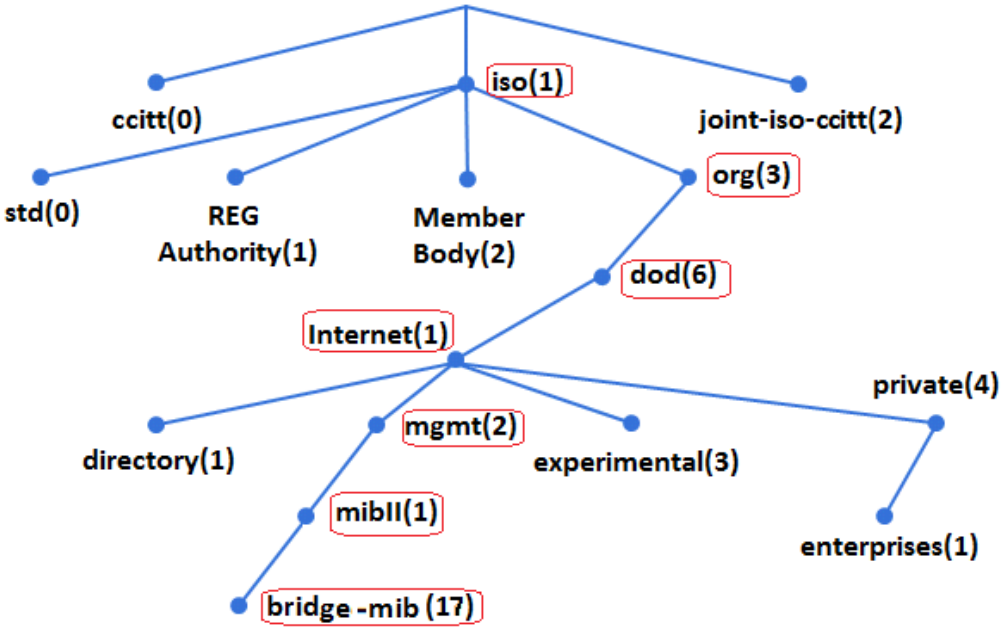

Figura 3.2: Subárvore que identifica a BRIDGE-MIB

Dentro da BRIDGE-MIB[10], estamos interessados em apenas um de seus objetos: o objeto *dot1dTpFdbEntry*, uma tabela de objetos representada pelo oid *1.3.6.1.2.1.17.4.3.1*. Este objeto é responsável por armazenar a tabela de MACs (endereços físicos) descobertos durante o funcionamento em modo de *transparent bridging* (chaveamento transparente) e as portas para as quais os pacotes devem ser encaminhados.

Sempre que um switch que implementa esse objeto recebe um pacote, ele registra o endereço físico do pacote te a porta pela qual ele foi recebido nesta tabela de sua MIB. Nem todos os switches comerciais possuem esse objeto na sua MIB, porém, no caso da rede da UFRGS foi identificado que todos os switches possuíam as variáveis necessárias para o funcionamento da coleta, e que todos operavam em *transparent bridging.*

Exemplificando esse modo de operação, considere um switch com 3 portas e com um dispositivo conectado em cada porta: o dispositivo X está conectado a porta 1, o dispositivo Y está conectado a porta 2 e o dispositivo Z está conectado a porta 3. Considere também o caso de uma transmissão simples, em que X transmite para Z. Na situação inicial, a tabela de encaminhamento está vazia, como pode ser visto na Tabela 3.1.

Tabela 3.1: Estado inicial da tabela de encaminhamento

MAC (dispositivo)	Porta
-	-

10 http://www.oidview.com/mibs/0/BRIDGE-MIB.html

Ao receber o pacote de *X* na porta *1*, o switch examina o pacote e gera um registro com o endereço de origem e a porta pela qual o pacote foi recebido. A nova situação da tabela de encaminhamento pode ser vista na Tabela 3.2:

Tabela 3.2: Tabela de encaminhamento após envio de X para Z

MAC (dispositivo)	Porta
X	1
-	-

O switch examina o pacote e encontra o endereço de destino *Z*. Como ele não encontra *Z* na sua tabela, ele inicia um processo de inundação (*flooding),* enviando o pacote por todas as suas outras portas. Tanto *Y* quanto *Z* recebem o pacote. Como o pacote é destinado ao dispositivo *Z, Y* o descarta. Enquanto isso, *Z* recebe e responde o pacote. O switch, então, recebe um pacote de *Z* na porta *3* e gera um registro na sua tabela com o endereço de origem do pacote e a porta pela qual ele foi recebido, alterando o estado da tabela de encaminhamento para a situação visível na Tabela 3.3:

Tabela 3.3: Tabela de encaminhamento após envio de Z para X

MAC (dispositivo)	Porta
X	1
Z	3
-	-

Em seguida, o switch examina o endereço de destino do pacote, no caso, o endereço de *X*. Ao ler sua tabela de encaminhamento, o switch encontra a porta pela qual o pacote deve ser enviado para chegar até *X*. Dessa forma, não é necessário efetuar a inundação e o pacote é enviado para *X* pela porta *1* do switch.

Considerando o funcionamento normal de uma rede, após o envio de *N* pacotes entre os dispositivos, cada switch deve ter uma tabela completa o suficiente para enviar pacotes originados de qualquer dispositivo que seja acessível em outra de suas portas. Cabe ressaltar que o *N* é variável em função do tamanho e da complexidade da rede. Cada entrada nesta tabela, é representada por 3 objetos, como se fossem 3 colunas na tabela:

- **dot1dTpFdbAddress** (oid *1.3.6.1.2.1.17.4.3.1.1)*
  - Endereço mac do registro;
- **dot1dTpFdbPort** (oid *1.3.6.1.2.1.17.4.3.1.2)*
  - Porta que foi registrada para o endereço mac durante o processo de transparent bridging;

- **dot1dTpFdbStatus** (oid *1.3.6.1.2.1.17.4.3.1.3)*
    - Informação de status da entrada na tabela. Possui 5 valores válidos, sendo que apenas registros de status 3 são salvos
        - **other** (1): indica que o registro foi gerado por outro processo e não o transparent bridging;
        - **invalid** (2): indica que o registro não é mais válido, ou seja, já passou do tempo de vida, mas ainda não foi removido da tabela;
        - **learned** (3): indica um registro válido e atualmente usado;
        - **self** (4): indica um endereço mac do próprio switch;
        - **mgmt** (5): indica um endereço estático armazenado no objeto *dot1dTpFdbAddress.*

Pelo fato da memória interna dos switches ser extremamente limitada, essas tabelas não são mantidas sem controle e indefinidamente. As entradas nessa tabela são limpas de tempos em tempos, e passam a ser consideradas inválidas após um tempo de validade, um intervalo entre 1 e 15 minutos, valor este que depende do dispositivo e da sua configuração.

3.2.2 Coletando dados da rede da UFRGS

A coleta de dados dos switches é extremamente baseada na base de dados da UFRGS, onde estão registrados todos os dispositivos da rede da UFRGS. Estes dispositivos são agrupados em entidades chamadas de blocos que, por sua vez, são agrupados em subredes.

Dentro da estrutura do sistema da UFRGS, os switches são cadastrados de forma diferente dos outros dispositivos: além dos dados de qualquer computador, como, por exemplo, nome, localização e usuário, eles recebem uma *string de comunidade* e um *identificador de objeto*, sendo esses dois dados necessários para a consulta a dados específicos da **MIB**.

Todos os dispositivos usam o identificador de objeto padrão para consultas sobre dados de switches. Porém, permitindo a especificação de um *oid* diferente do padrão abrimos a possibilidade do uso de MIBs proprietárias, que por sua vez podem trazer mais informações ao sistema.

A string de comunidade é uma *senha* para acesso aos dados da MIB. Sem a informação correta de comunidade, o dispositivo consultado não responde a consulta, tornando inviável a coleta de dados. Na maioria dos dispositivos, as strings de comunidade configuradas como padrão são a string *public* ou *private.* Esse fato é considerado uma das maiores ameaças à segurança de redes[11]. Na UFRGS, todas as comunidades de switches e roteadores foram configuradas com senhas reais, dando mais segurança à rede. Por esse motivo, no nosso caso de estudo da UFRGS, foi necessário acrescentar essa informação ao cadastro dos switches.

O algoritmo de coleta pode ser descrito da seguinte forma, sendo **Main** o script de coleta e **Collect** um processo criado por **Main** para a coleta de um switch específico:

11 http://www.sans.org/top20/2000

Main

Entrada: -

Saída: log dos tempos de início e fim da coleta de cada switch

1. Cria registro de início de coleta
2. Busca todos os switches cadastrados na base da UFRGS
3. Executa uma coleta teste, inserindo os switches acessíveis em **$switch**
4. Para cada switch **x** acessível, executa *Collect (***$switch[x]***)*
5. Para cada processo *Collect* encerrado, busca da memória compartilhada o indicador de sucesso e os tempos de início e fim da coleta.

Collect

Entrada: switch a ser coletado

Saída: boolean indicando sucesso da coleta e tempos da coleta em uma área de memória compartilhada

1. Busca todos os registros que fazem parte da mesma subrede do *switch* em questão e insere em **$coletar[]**
2. Busca a topologia válida para o switch no momento da coleta e mantém em **$topologia[]**, onde **$topologia[$mac][$porta]** = *false*
3. Inicializa **$nova_topologia[]** para armazenar novos registros
4. Para cada endereço **$mac** em **$coletar[]**
    a. Executa um *snmp get* com a comunidade do *switch* buscando a **$porta**.
    b. Se **$porta** é válida e existe um registro em **$topologia[]** para o **$mac** e **$porta,** então **$topologia[$mac][$porta]** = *true*
    c. Se **$porta** é válida mas não existe um registro em **$topologia[]**, então **$nova_topologia[$mac][$porta]** = *true*
5. Para cada endereço *mac* salvo em **$nova_topologia** um novo registro inserido no banco.
6. Para cada endereço mac em **$topologia**
    a. Se **$topologia[$mac][$porta]** == *false*, fecha o registro indicando final da sua validade.
    b. Caso contrário, nada é feito.
7. Salva em uma área compartilhada da memória o tempo de início e fim da coleta, e um indicador de coleta bem sucedida.

O algoritmo Main é responsável por dar início a coleta. Main gera o registro de início de coleta e consulta cada um dos switches cadastrados no banco de dados da UFRGS, validando quais são atingíveis em uma consulta SNMP. Essa etapa de validação (etapa 3 de Main) reflete a escolha por um algoritmo otimista para a execução da coleta. Ao encontrar um switch inalcançável, o algoritmo é otimista e considera que o estado daquele switch se manteve inalterado, ao invés de considerar que todos os registros deixaram de ser válidos. Em seguida, Main dispara N processos, um para cada switch válido, iniciando a coleta propriamente dita, coleta essa representada aqui pelo algoritmo **Collect**.

O algoritmo **Collect** lida apenas com dados de um switch específico. Ele busca por todos os endereços físicos que fazem parte da mesma subrede do switch na base da UFRGS, e executa uma consulta para cada endereço físico encontrado. Como resultado desta consulta, temos a porta na qual o endereço físico foi detectado ou um retorno nulo caso o endereço não tenha tido pacotes encaminhados por aquele switch.

O resultado de todas essas consultas é comparado ao estado anterior do switch, encontrando assim novos registros e registros que deixaram de ser válidos. Todos os novos registros e todas as atualizações de validade são feitas como uma transação, de forma que a topologia de um switch não é atualizada de forma parcial. Após a confirmação dessa transação, passamos a ter um novo estado atual para cada switch, podendo ele ser igual ao estado anterior.

3.3 Banco de Dados Temporal

O objetivo do banco de dados temporal desenvolvido neste trabalho é especificamente armazenar os dados coletados pelo script coletor (seção 3.2), de forma a manter um histórico de quais dispositivos foram detectados em cada porta de cada switch coletado. Com essas informações, é possível inferir uma topologia dos switches de uma rede, bem como as mudanças ocorridas nas interconexões de uma rede, já que estas podem mudar ao longo do tempo.

Para manter as informações do coletor, foi utilizado um banco de dados por **tempo de transação**. A escolha de um banco de dados por tempo de transação foi feita a partir da análise do funcionamento da coleta de dados da rede, onde foram levantados 3 fatos que definiram essa escolha:

1. *A coleta não necessariamente reflete a validade real dos dados.* A coleta de dados é feita com base em tabelas internas mantidas pelos dispositivos de rede. Essas tabelas mantém apenas dados recentes do funcionamento da rede, com uma vida média de 5~15 minutos. Considerando essa vida útil, o dado coletado frequentemente será um dado de um passado recente, sendo assim um dado que não reflete fielmente a sua validade no mundo real, mas sim o instante em que foi registrado na tabela interna do dispositivo.
2. *Alguma parte da rede pode estar inacessível.* Esta é outra questão detectada que pode ser causada por instabilidades que não podem ser previstas pelo coletor. Qualquer equipamento de rede pode ter uma falha e tornar o resultado da coleta *parcialmente incorreto.* Assim, temos mais uma situação em que os dados coletados e a informação temporal associada a eles não é totalmente fiel a validade do registro no mundo real.

3. *Pode ocorrer uma falha na máquina de coleta.* Mais um tipo de falha que pode ser *tolerada* com uma boa prática de tolerância a falhas, mas ainda assim pode vir a gerar registros que não refletam a validade do dado no mundo real. A tolerância a falhas poderia executar a coleta novamente após uma tentativa falha, ou disparar o processo em outra máquina, porém, isso não evitaria que a coleta deixasse de ser totalmente fiel a realidade dos dados.

Por esses três motivos foi feita a escolha por um banco de dados de tempo de transação. A Figura 3.3, através de um diagrama ER, mostra a estrutura de tabelas criadas para armazenar a coleta.

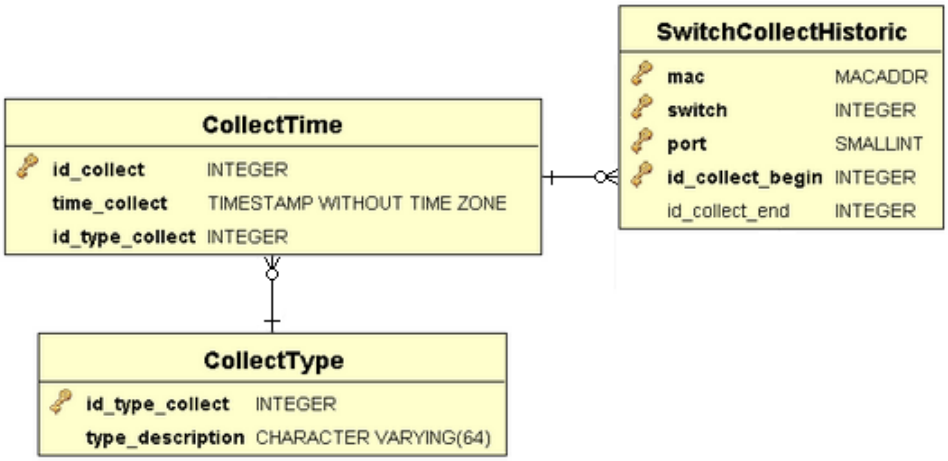

Figura 3.3: Tabelas utilizadas para manter os registros temporais da coleta

Por se tratar de um sistema intimamente ligado ao sistema de registro de estações da UFRGS, é necessário armazenar poucos dados sobre os registros coletados nos switches. Dessa forma, apenas três tabelas foram necessárias:

- ***CollectType.*** Todo o trabalho foi desenvolvido em torno de apenas um tipo de coleta, que é a coleta temporal da topologia dos switches. Analisando as possibilidades que uma coleta temporal nos dá, poderíamos desenvolver outros tipos de coleta: uma coleta para roteadores, uma coleta para switches de determinado campus da UFRGS, etc. Especializando o tipo de coleta, podemos utilizar a mesma tabela de histórico de coleta para outros tipos de coletas de switches e também utilizar a mesma tabela de registros de início de coleta para processos que não envolvessem switches. Essa possibilidade foi coberta na estrutura de dados especificada para este trabalho, porém a criação e a implementação de outros tipos de coletas não foram cobertas pela ferramenta SiViCS.
- ***CollectTime.*** Para facilitar o armazenamento e reduzir o tamanho do banco, decidiu-se usar uma referência a uma tabela de *tempos* para armazenar a informação temporal do dado coletado. Dessa forma, a informação de tempo é armazenada como um inteiro que aponta para um timestamp na tabela *CollectTime*. A cada início de execução do script coletor, um novo registro é gerado em *CollectTime*, indicando o início de uma nova coleta e salvando o

timestamp que será utilizado em todos os registros gerados ou atualizados naquela execução.

- ***SwitchCollectHistoric.*** Nesta tabela é feito todo o armazenamento dos dados coletados nos switches. Em resumo, aqui ficam os macs que foram detectados nos switches, bem como a porta na qual foram lidos e o período de validade deste registro. São armazenados apenas 5 dados: o identificador do switch, a porta do switch, o mac coletado, e os identificadores de início e fim de validade do registro. O *identificador do switch* é um numérico utilizado pelo sistema de gerência da UFRGS que aponta para a tabela de dispositivos do sistema, e através desse identificador, é possível buscar todos os dados de um switch dentro do sistema, tais como, nome, localização, ip, mac e responsável pelo registro. O *mac* vem do próprio sistema de gerência da UFRGS. Como foi descrito no funcionamento do script coletor, o script busca informações somente dos registros que fazem parte da rede do switch coletado. Para cada mac, ele verifica se ele está ligado ao switch, retornando o número da *porta* em caso positivo. Por fim, são armazenadas duas informações temporais em cada registro desta tabela, representando o *início* e o *fim da validade de um registro*. Um registro com fim de validade vazio representa um registro ainda válido, ou seja, que está sendo repetidamente coletado na rede da UFRGS.

3.4 Visualização Topológica

Esta camada da ferramenta SiViCS tem o objetivo de exibir para o usuário um retorno visual dos dados salvos na base de dados temporal durante a etapa de coleta de dados pelo script coletor. Essa visualização gráfica se dá por meio de um grafo, onde cada nodo pode representar um dispositivo da rede da UFRGS, uma porta de um switch, um bloco ou subrede existente no sistema de gerência de redes da UFRGS.

A visualização foi implementada como uma aplicação web mista de PHP com um Applet escrito em Java, e possui duas partes bem definidas: a primeira para especificação do grafo e de seus parâmetros, e uma segunda parte onde o grafo é visível e manipulável, conforme detalhadamente descrito nas subseções a seguir.

3.4.1 Ferramentas e Parâmetros da Interface de Visualização

A interface de visualização foi implementada utilizando a mesma linguagem e o mesmo ambiente de desenvolvimento do sistema de gerência de redes da UFRGS: o PHP e o ambiente de desenvolvimento do CPD/UFRGS[12].

Dessa forma, seu código segue os novos padrões de desenvolvimento da UFRGS, que incluem uma identidade visual definida para todos os sistemas da UFRGS, o uso do framework *Yii*[13] e do framework de javascript *jQuery*[14].

12 http://www.ufrgs.br/cpd/

13 http://www.yiiframework.com/

14 http://jqueryui.com/

A identidade visual dos sistemas é dada através dos arquivos de estilo, ou *arquivos CSS*, criados por designers do CPD para serem utilizados e adaptados para todos os novos sistemas da UFRGS.

O framework Yii garante o bom uso da arquitetura MVC - model/view/controller - no sistema, garantindo uma melhor organização de código, além de padrões para uma boa códificação e diversas abstrações para facilitar acesso a banco de dados.

O uso da jQuery fornece uma abstração no uso de javascript do sistema, tornando o código mais legível e otimizado, e abre uma gama imensa de widgets, interações e efeitos visuais para o sistema, tais como calendários personalizáveis, barras de seleção (*sliders*) e efeitos de transparência/deslocamento. Além disso, o uso da jQuery garante o funcionamento correto entre browsers, já que todas suas ferramentas são testadas e funcionais em um grande conjunto de browsers: Internet Explorer 6.0+, Firefox 3.6+, Safari 5.0+, Opera, Chrome).

Definidas as ferramentas utilizadas, podemos partir para a definição e funcionamento da interface. Na Figura 3.4 temos a tela inicial da ferramenta.

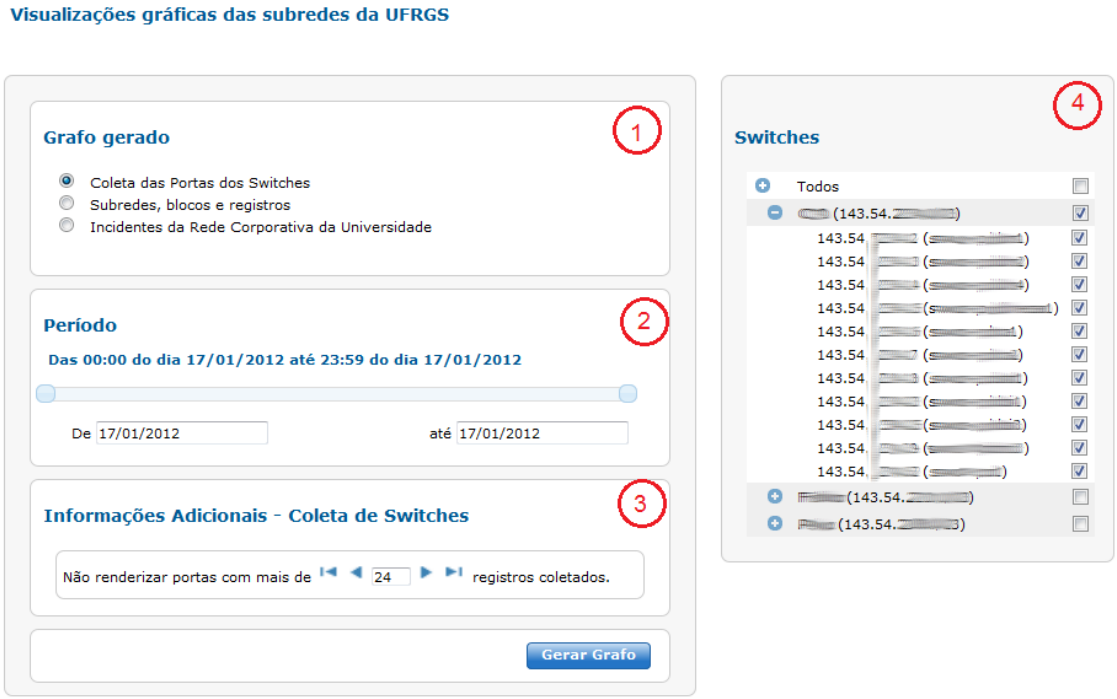

Figura 3.4: Página Inicial da Ferramenta de Visualização

O primeiro parâmetro visível na tela é um seletor do tipo de grafo a ser gerado **(1)**. Para garantir uma maior utilização da ferramenta, foram definidos dois tipos de grafos adicionais. Além do grafo de *coleta de portas dos switches*, existem os grafos de *registros do sistema* e de *incidentes do sistema*, descritos da seguinte forma:

- **Coleta das Portas dos Switches.** É o grafo gerado como parte principal do trabalho e será descrito detalhadamente na sequencia do texto.
- **Subredes, blocos e registros.** É o grafo criado para renderizar as entidades de *subredes*, *blocos* e *registros* cadastrados no sistema de registro de estações da UFRGS. Para este tipo de grafo, o único parâmetro definido pelo usuário é a lista de subredes e blocos que devem ser renderizadas pelo grafo.

Apesar de ser capaz de manipular quantidades enormes de dados, o Prefuse não foi capaz de renderizar com rapidez o grafo de toda UFRGS, e por esse motivo, este tipo de grafo é aconselhado para visualizações de pequenas partes da rede da UFRGS.

- **Incidentes da Rede Corporativa da Universidade.** É o grafo criado para acompanhar os incidentes de segurança de rede registrados na rede da UFRGS. Neste grafo, apenas incidentes da rede corporativa são cobertos. Isto significa que, incidentes em dispositivo que não são patrimônio da UFRGS não são cobertos. Incidentes que ainda estão atualmente ativos, ou seja, ainda não foram resolvidos pelo *time de resposta a incidentes*[15] da UFRGS, aparecem em vermelho, enquanto que os incidentes que já foram resolvidos aparecem em cinza.

Além do tipo de grafo, existem mais três outros parâmetros que podem ser definidos nesta tela:

- **Período (2).** O usuário deve definir o período que será utilizado como filtro na consulta que busca os registros que farão parte do grafo. Este parâmetro é utilizado pelo grafo de incidentes e pelo grafo da coleta de switches.

  Ao se selecionar duas datas iguais, ou seja, definir que o filtro será para um dia específico, a barra de seleção de horário abaixo das datas se comporta como na Figura 3.5. Isto significa que existe apenas uma única barra para a seleção do horário de início e do horário de fim. Caso duas datas diferentes sejam selecionadas, a barra de seleção de horário se divide em duas, uma para seleção de horário de início e outra para seleção do horário de fim.

Figura 3.5: Seletor de período para datas diferentes

- **Switches / Blocos (3).** Na direita da página está localizada uma área para definição da área de consulta dos grafos. Nesta área, para os grafos de registros e de incidentes, o usuário define quais subredes e blocos terão seus registros retornados. No grafo de coleta de switches, o usuário define quais subredes e quais switches dentro de cada subrede serão renderizados no grafo gerado.
- **Informações Adicionais - Coleta de Switches (4).** Esta área de parâmetros foi definida para parâmetros específicos do grafo de coleta dos switches, sendo que até o final do desenvolvimento da ferramenta, apenas um parâmetro adicional foi necessário: limite de registros por porta. Este parâmetro foi definido para gerar uma visualização mais limpa do grafo. Como o script coletor busca registros a partir da tabela de encaminhamento

15 http://www.ufrgs.br/tri/

dos switches, portas com outros switches conectados podem ter dezenas de registros coletados, já que todos os pacotes encaminhados para endereços do switch filho irão passar por aquela porta. Dessa forma, utilizando este parâmetro, pode-se remover do grafo portas com mais do que N registros pendurados, retirando assim as portas que possivelmente têm um switch conectado a elas. Cabe ressaltar que, esse parâmetro é experimental, sendo definido apenas para gerar grafos mais limpos e legíveis.

Definidos os parâmetros da interface de visualização e o tipo de grafo a ser gerado, avança-se para a visualização final, onde o grafo é renderizado e pode ser visualizado pelo usuário do sistema.

3.4.2 Grafo e manipulação de dados

Esta seção descreve a etapa de exibição (visualização) dos grafos. O foco da explicação está voltado para o tipo de grafo *coleta das portas dos switches da* UFRGS porque representa o escopo principal deste trabalho.

Definidos os parâmetros do grafo (datas de início e fim da coleta, switches que devem ser renderizados, e ponto de corte das portas), o grafo pode ser gerado clicando em **Gerar Grafo.** O grafo é renderizado em uma nova aba do browser, o que permite ao usuário alterar algum parâmetro e gerar outro grafo semelhante mais rapidamente. Na Figura 3.6 podemos ver um exemplo do resultado final do grafo, com o applet na parte superior, e a legenda das cores do grafo na parte inferior.

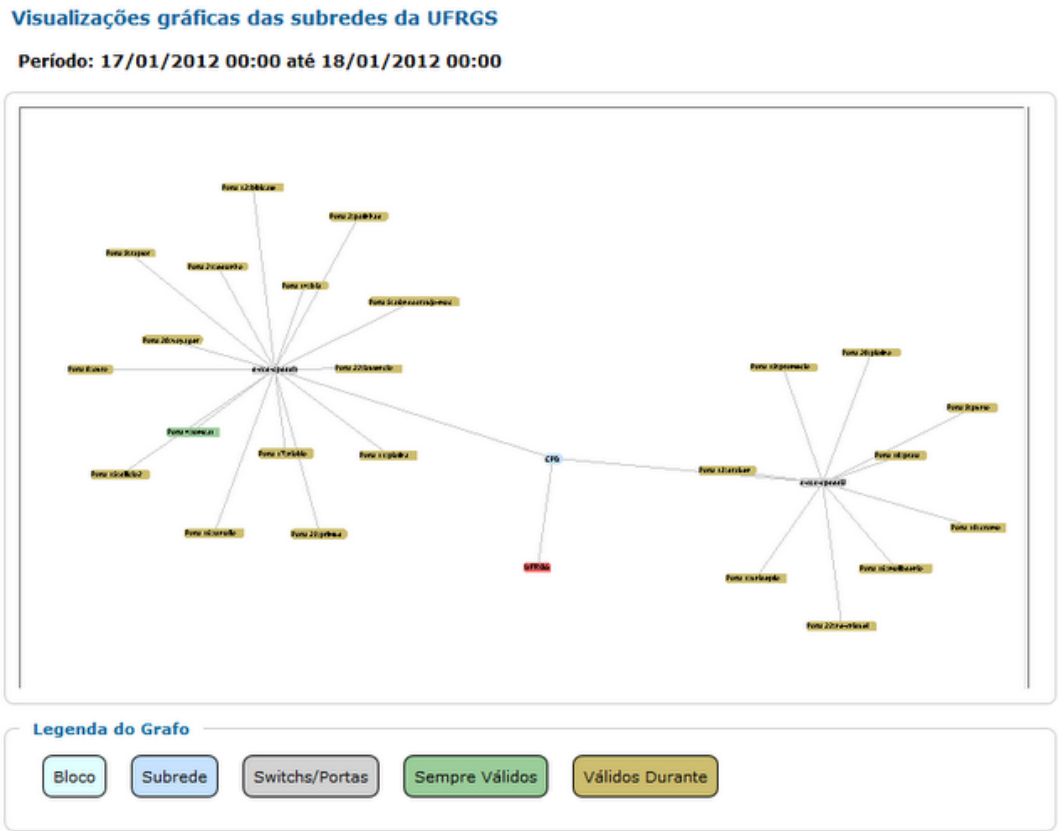

Figura 3.6: Grafo gerado pela ferramenta

Ao carregar a página web, é feita a consulta sobre as tabelas da base de dados temporal definida no trabalho e da base da UFRGS, buscando os dados coletados nas portas dos switches. Como os parâmetros de período na página web são timestamps, é feita uma consulta a tabela **CollectTime**, buscando os registros que representam a data de início e a data de fim selecionadas pelo usuário. Caso as datas estejam foram do período de coleta, a ferramenta dispara uma exceção com o erro apropriado.

Com o período da coleta validado, pode-se efetuar a consulta para construção do grafo. A consulta da Figura 3.7 ilustra a consulta que permite retornar os dados do sistema de gerência de redes da UFRGS.

```
SELECT DISTINCT
        S.switch,
        S.port,
        S.mac,
        CASE
       WHEN COALESCE(S.id_collect_end, $Fim +1)  > $Fim AND
                S.id_collect_begin < $Inicio THEN  '0'
       WHEN COALESCE(S.id_collect_end, $Fim +1)  <= $Fim AND
                S.id_collect_begin >= $Inicio THEN '1'
       WHEN COALESCE(S.id_collect_end, $Fim +1)  > $Fim AND
                S.id_collect_begin >= $Inicio THEN '2'
       WHEN COALESCE(S.id_collect_end, $Fim +1)  <= $Fim AND
                S.id_collect_begin < $Inicio THEN '3'
          END AS interval
FROM
        "SwitchCollectHistoric" S
WHERE
        S.switch in ($Switches)
        AND ( S.id_collect_begin BETWEEN $Inicio AND $Fim
            OR S.id_collect_end BETWEEN $Inicio AND $Fim )
ORDER BY
        S.switch,
        S.port,
        S.mac
```

Figura 3.7: Consulta na base temporal

Esta consulta utiliza as variáveis *$Inicio* e *$Fim*, que são os inteiros que representam o período selecionado pelo usuário na interface. Além disso, a variável *$Switches*

contém a lista dos ids numéricos que identificam os switches que serão retornados pela consulta.

A consulta garante que os registros retornados foram válidos em algum momento dentro do período selecionado e que são registros de um dos switches selecionados pelo usuário. Além dos dados retornados na consulta, também são retornados todos os dados necessários para se identificar o registro da máquina correspondente ao endereço físico dentro da UFRGS.

O comando **CASE** definido na seleção retorna um atributo chamado *interval* que define o tipo de validade do registro. Em cada uma das possibilidades é utilizado o comando **COALESCE** que garante que o teste será feito com um valor válido no caso do registro ainda ser válido, ou seja, ter um *id_collect_end* vazio. Este atributo *interval* pode ter os seguintes valores e significados:

- ***interval == 0:*** o registro começou a ser válido antes da data de início definida pelo usuário, e ainda continua válido;
- ***interval == 1:*** o registro passou a ser válido após a data de início definida pelo usuário, e terminou antes da data de fim especificada;
- ***interval == 2:*** o registro começou a ser válido após a data de início, e ainda continua válido;
- ***interval == 3:*** o registro passou a ser válido antes da data de início, e já não é mais válido na data de fim selecionada pelo usuário.

Após a execução da consulta, tem-se uma quantidade de dados que ainda precisa ser manipulada no PHP, pois se pode ter uma máquina que teve mais de um tipo de validade dentro do período, ou seja, gera-se uma linha para cada valor do atributo *interval* retornado.

Para garantir que cada nodo será desenhado apenas uma vez, o retorno dessa consulta é varrido, sendo os registros classificados em dois tipos:

- **Sempre válidos:** registros que retornaram o atributo *interval igual a 0* **ou** que retornaram no mínimo dois registros, um com *interval igual a 2* e outro com *interval igual 3*, ou seja, já eram válidos antes do início do período e também foram válidos no final do período definido; e
- **Parcialmente válidos:** por exclusão, todos os outros registros.

Após essa manipulação que classifica a validade dos registros, os dados são manipulados de forma a gerar os nodos que serão passados como parâmetros para o applet. Com os dados da consulta são gerados os seguintes nodos:

- Um nodo **raiz** intitulado *UFRGS*;
- Um nodo para cada entidade **subrede**;
- Um nodo para cada **switch**;
- Um nodo para cada **porta** retornada, sendo que:
    - Caso a porta tenha tido apenas um registro coletado, o nome da máquina no sistema de registro da UFRGS estará no mesmo nodo da porta;

- Caso a porta tenha dois ou mais registros coletados, ela terá tantos nodos filhos quantos forem os dispositivos retornados.

Cada nodo recebe um identificador numérico único. Esse identificador é utilizado para gerar uma string com as ligações do grafo, com a sintaxe *A-B@B-C-D-E@C-F-G...*, onde:

- **A** possui o filho **B**;
- **B** possui os filhos **C**, **D** e **E**;
- **C** possui os filhos **F** e **G**, e assim por diante.

Assim, somente são descritas as ligações de pai para filho, que geram uma descrição menor para definir todas as ligações do grafo.

Definidos os parâmetros do applet, o PHP gera o HTML necessário para carregar o applet no browser e passar todos os parâmetros para código java. Um exemplo de grafo gerado pelo sistema pode ser visto na Figura 3.8, com a coleta dos switches da subrede do CPD, entre os dias 17 e 19 de janeiro de 2012.

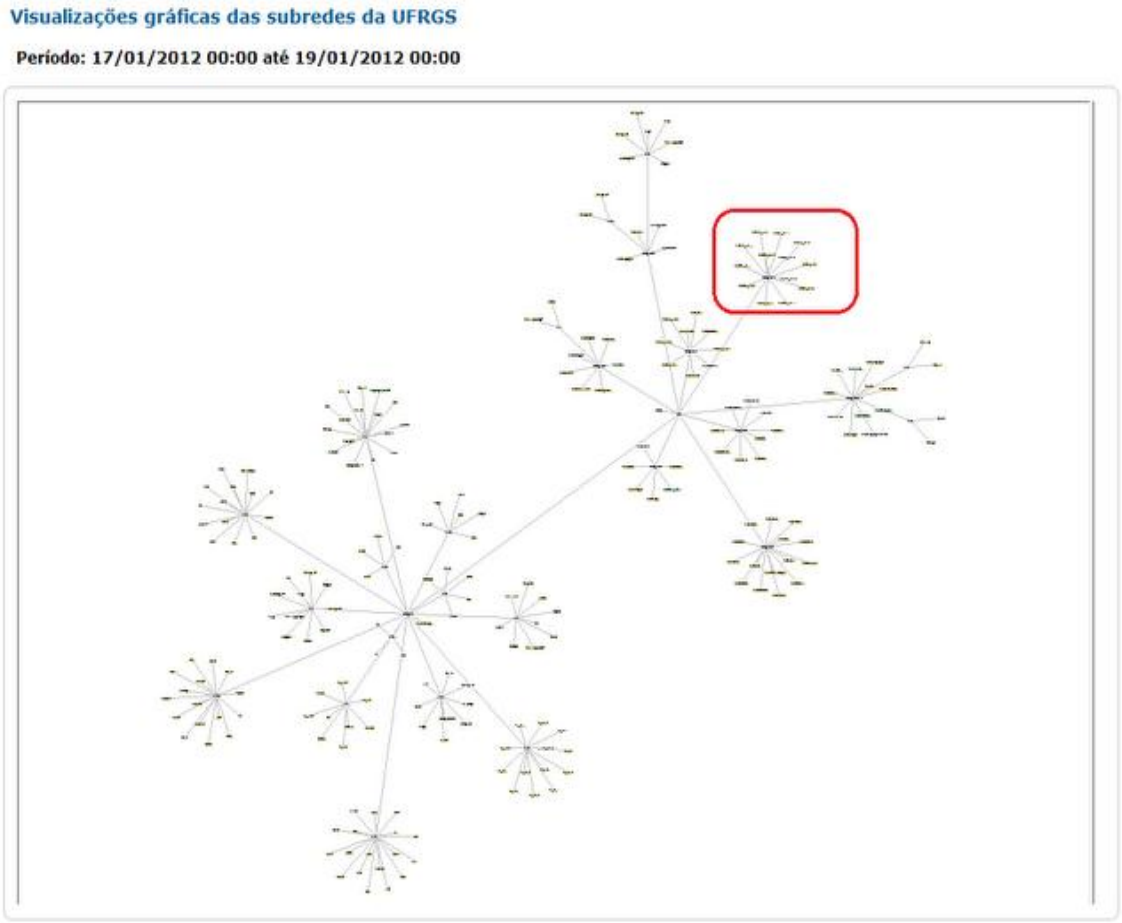

Figura 3.8: Grafo de coleta de portas de switch

O Prefuse fornece diversas interações básicas, como efeitos de força e movimentação ao arrastar os nodos, escala do grafo para que fique totalmente visível na área destinada ao applet, *zoom in* e *zoom out* utilizando o mouse. Na Figura 3.9 podemos ver a área destacada da Figura 3.8 com mais clareza:

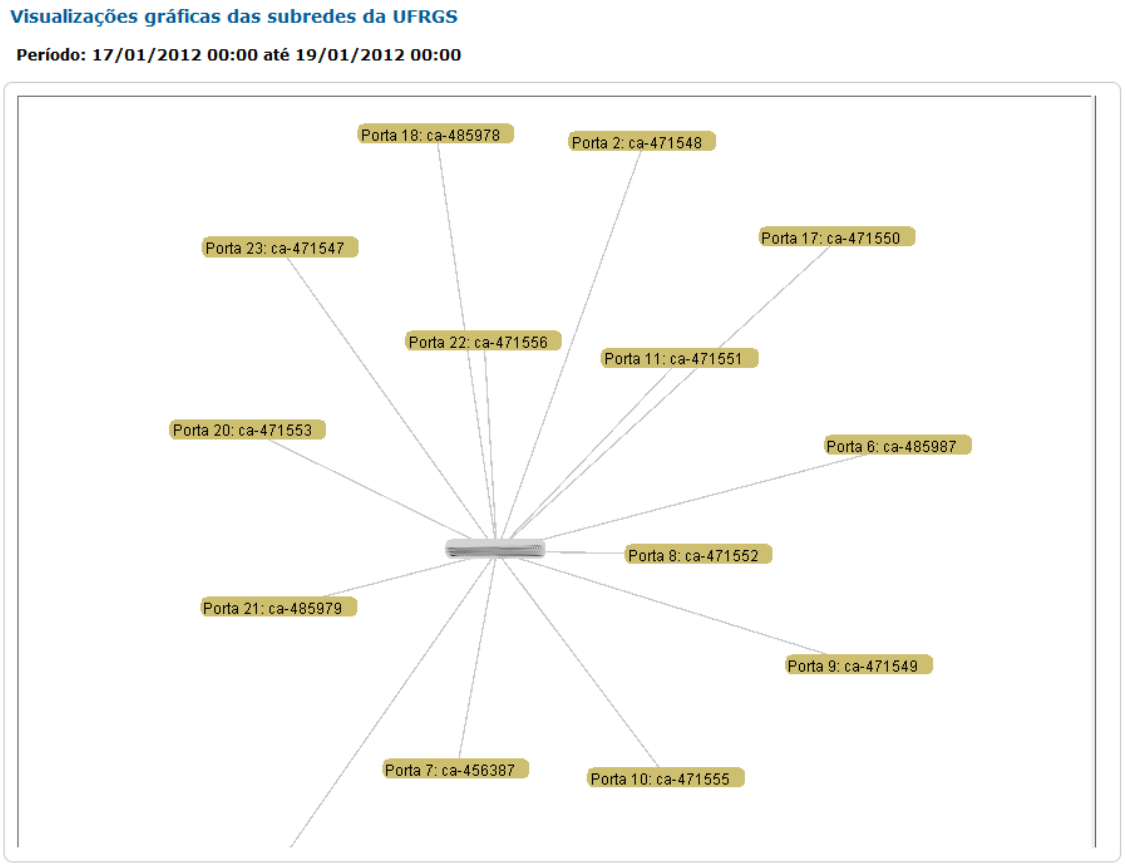

Figura 3.9: Zoom na área destacada no grafo

O funcionamento básico do Prefuse é simples e ele fornece todos os tipos de dados necessários para definir o grafo. O código em java processa todos os parâmetros passados pela interface em PHP, gerando os registros dos nodos e das ligações dos nodos numa estrutura do tipo *graph*.

O próprio Prefuse já é otimizado para receber este tipo de estrutura, gerando a visualização do grafo visto nas Figuras 3.8 e 3.9. A coloração dos nodos é baseada em um atributo definido pelo processamento em PHP. Cada tipo de nodo recebe uma cor única, ou seja, nodos que representam a entidade bloco possuem uma cor diferente dos que representam a entidade switch, e assim por diante. A legenda de cores fica visível na interface abaixo do applet.

3.4.3 Grafos Adicionais

Como citado, o SiViCS possui outros dois grafos adicionais especificados: *Incidentes na Rede Corporativa da Universidade* e *Subredes, blocos e registros.*

A Figura 3.10 exemplifica um grafo de incidentes da rede corporativa da UFRGS, enquanto que as Figuras 3.11 e 3.12 ilustram o grafo de subredes e blocos.

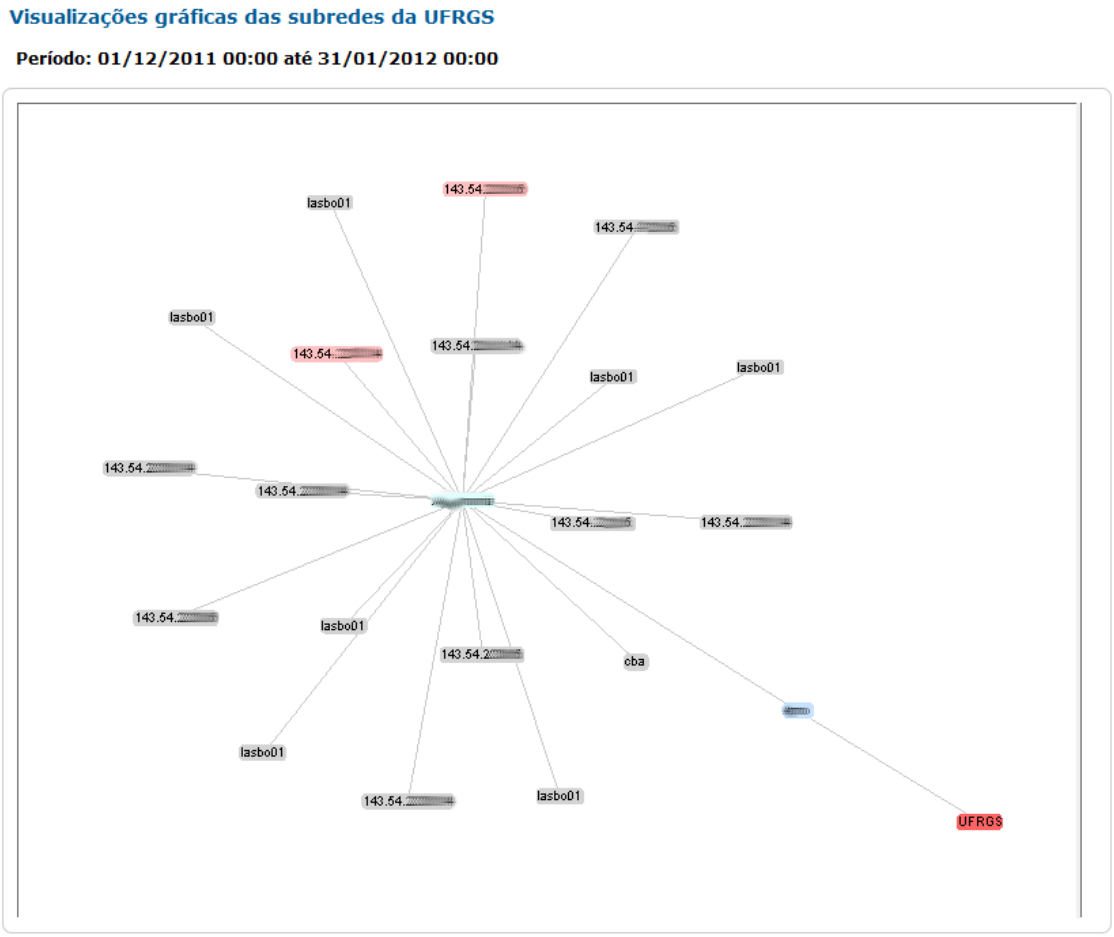

Figura 3.10: Incidentes na subrede

Na Figura 3.10 estão os incidentes de segurança numa das subredes da UFRGS em janeiro de 2012. Em cinza estão os registros já resolvidos, e em vermelho claro os incidentes ainda abertos.

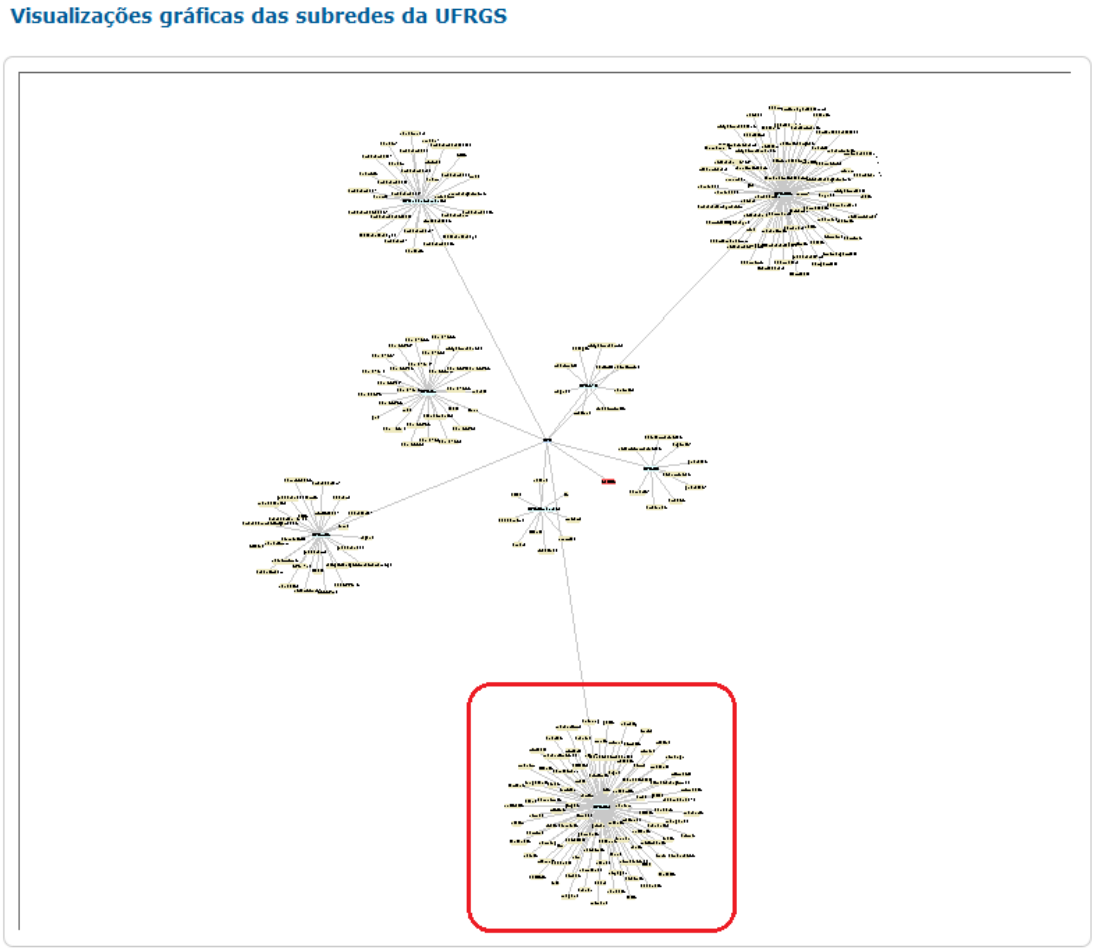

Figura 3.11: Registros da subrede do CPD

Na Figura 3.11 está uma execução do sistema onde podemos ver os registros da subrede do CPD. Cada *grupo* de nodos formado pela visualização são os registros ao redor de um bloco específico. Podemos mudar o zoom da ferramenta e visualizar melhor a área que foi destacada.

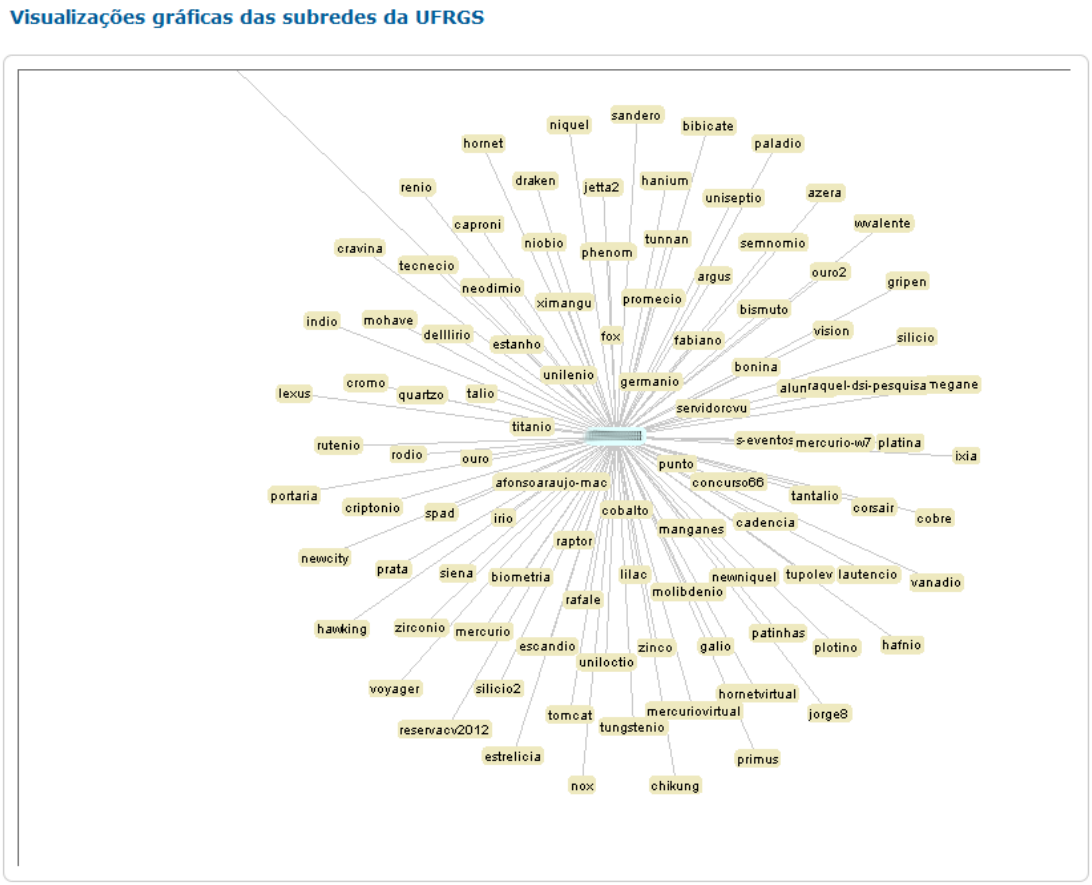

Figura 3.12: Bloco do CPD-DSI dentro da subrede CPD

A Figura 3.12 mostra a área destacada da Figura 3.11, onde podemos ver um bloco específico da subrede do CPD, no caso o bloco CPD-DSI.

3.5 Resultados

Após o desenvolvimento de um script coletor de dados da MIB dos switches, do estabelecimento de uma base temporal para o armazenamento da topologia dos switches e da implementação de uma interface web para visualização gráfica de redes, podemos coletar alguns números referentes aos resultados do trabalho, especialmente nas duas primeiras camadas desenvolvidas: o coletor e a base.

3.5.1 Resultados da Base de Dados Temporal

Os primeiros resultados que podemos tirar do trabalho são os vistos na Tabela 3.4, onde podemos ver alguns números relativos ao estado final da base temporal do trabalho:

Tabela 3.4: Resultados da base temporal

Primeira Coleta	10/10/2011 às 14h50min
Última Coleta	23/01/2012 às 12h30min
Coletas Efetuadas	4834 coletas

Dias de Coleta	104 dias
Switches Coletados	28 switches
Portas Coletadas	412 portas
Registros Gerados	558835 registros
Tamanho das Tabelas	30 MB
Tamanho dos Índices das Tabelas	70 MB

Foram coletados 28 switches da rede da UFRGS, todos situados próximo ao CPD da UFRGS, localizado no campus saúde da mesma. Essa quantidade representava aproximadamente 10% dos switches da UFRGS no início do trabalho (cerca de 260 switches). No encerramento deste trabalho, a UFRGS já contava com aproximadamente 360 switches, o que reduziu a cobertura da coleta para aproximadamente 7%.

3.5.2 Resultados dos Tempos Médios de Coleta

O coletor foi desenvolvido para, além de executar a coleta propriamente dita, manter a informação de início e fim de cada execução. Dessa forma foi possível fazer uma análise do tempo médio de coleta em cada switch coletado da UFRGS, gerando os gráficos a seguir. Na Figura 3.13 estão os resultados dos switches da *subrede de testes I*, na Figura 3.14 estão os resultados dos switches da subrede *de testes II*, e por fim, na Figura 3.15, os resultados dos switches da subrede *de testes III*.

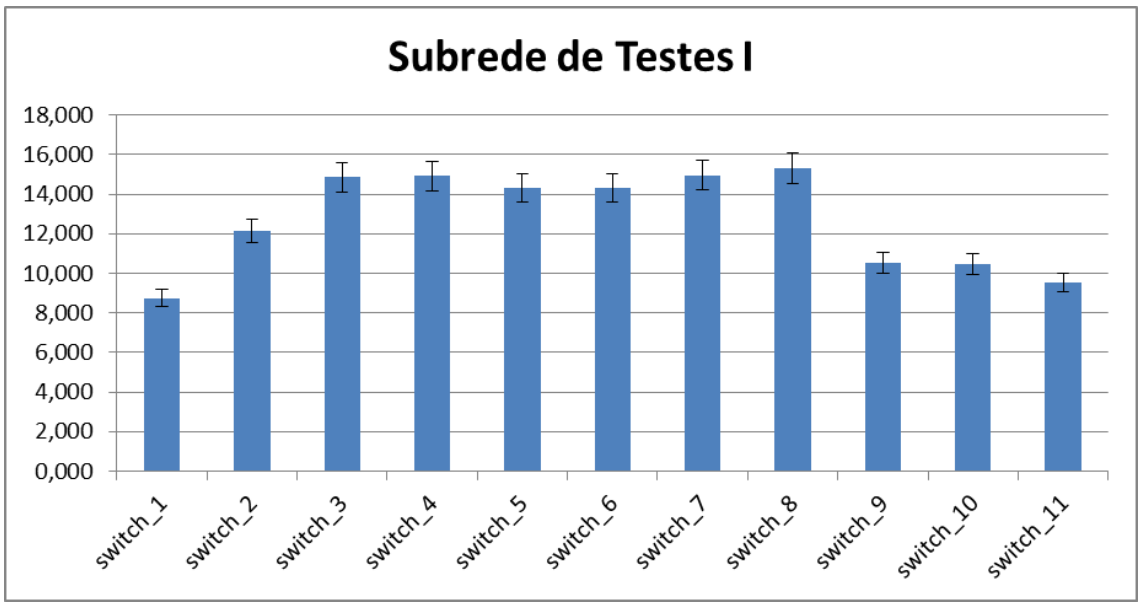

Figura 3.13: Tempo médio de coleta e intervalo de confiança da subrede de testes I

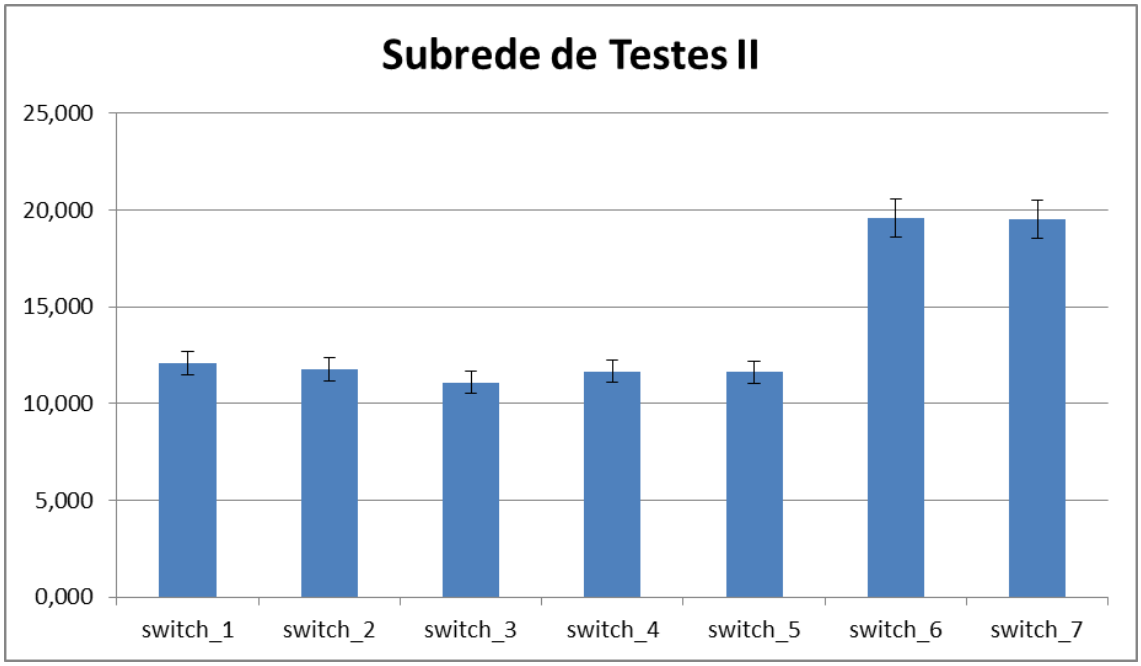

Figura 3.14: Tempo média de coleta e intervalo de confiança da subrede de testes II

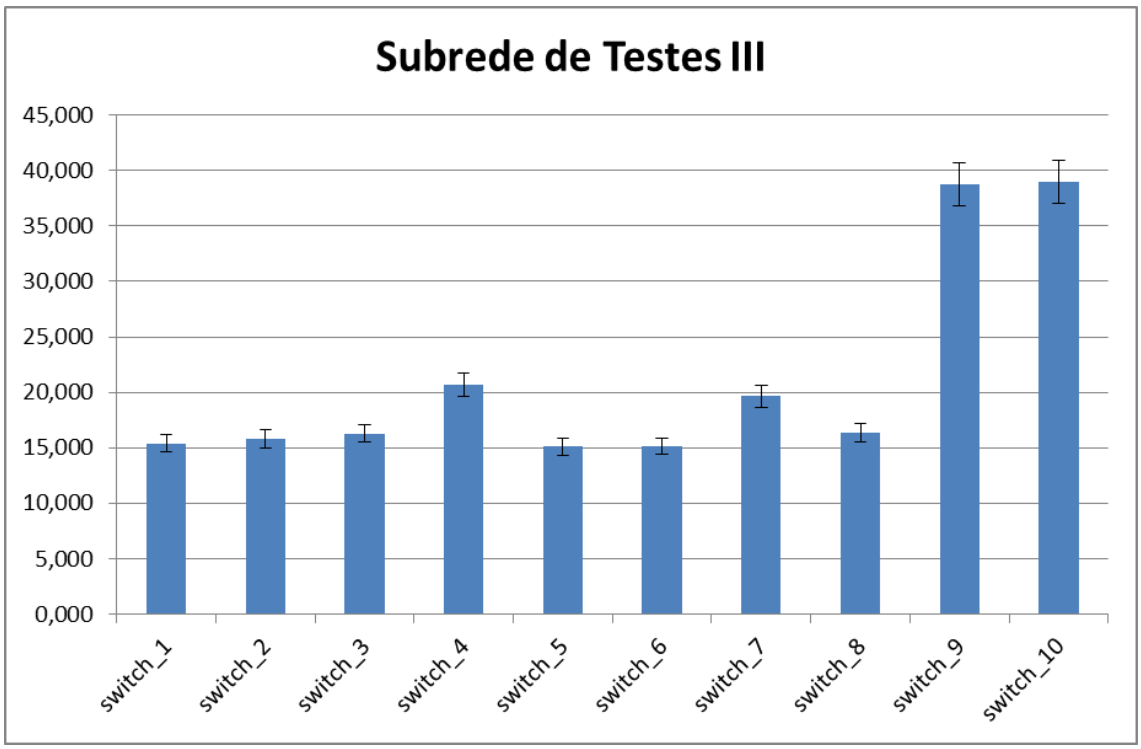

Figura 3.15: Tempo médio de coleta e intervalo de confiança da Subrede de testes III

Os tempos foram calculados em cima de uma amostra de 350 coletas efetuadas, com nível de confiança especificado em 95%. Os gráficos foram plotados sem valores, e nas tabelas a seguir, estão os valores das médias e dos intervalos de confiança.

Tabela 3.5: Médias e intervalos de confiança na subrede de testes I

Subrede de Testes I		
Switch	*Média (seg.)*	*Intervalo (seg.)*
switch_1	8,740	0,840
switch_2	12,134	0,906
switch_3	14,851	1,225
switch_4	14,925	1,238
switch_5	14,304	1,211
switch_6	14,284	1,020
switch_7	14,949	0,937
switch_8	15,296	1,033
switch_9	10,501	0,872
switch_10	10,466	0,878
switch_11	9,540	1,129

Tabela 3.6: Médias e intervalos de confiança na subrede de testes II

Subrede de Testes II		
Switch	*Média (seg.)*	*Intervalo (seg.)*
switch_1	12,093	0,889
switch_2	11,749	0,880
switch_3	11,096	0,860
switch_4	11,648	0,874
switch_5	11,618	0,877
switch_6	19,558	2,783
switch_7	19,525	2,789

Tabela 3.7: Médias e intervalos de confiança na subrede de testes III

Subrede de Testes III		
Switch	*Média (seg.)*	*Intervalo (seg.)*
switch_1	15,367	1,234
switch_2	15,797	1,484
switch_3	16,296	1,774
switch_4	20,669	2,704
switch_5	15,090	1,375
switch_6	15,128	1,371
switch_7	19,660	2,543
switch_8	16,340	1,794
switch_9	38,782	4,606
switch_10	38,928	4,623

Cada tabela apresenta o resultado de uma das subredes coletadas: na Tabela 3.5 temos os tempos da subrede de testes I, na Tabela 3.6 temos os tempos da subrede de testes II, e por fim, na Tabela 3.7 temos os tempos da subrede de testes III.

3.5.3 Análise dos Resultados

Apesar de não termos conseguido cobrir uma grande área da UFRGS, restringido a ferramenta a 3 subredes e 28 switches (menos de 10% da UFRGS), foi criada uma quantidade significativa de dados, e podemos projetar que para uma coleta que cobrisse todos os switches da UFRGS teríamos aproximadamente 6 milhões de registros no mesmo período de coleta.

Quanto a coleta, ela se mostrou extremamente eficiente, principalmente por ela ser direcionada e buscar apenas os endereços físicos da mesma subrede de cada switch coletado. Nesse ponto, a estrutura da UFRGS e do sistema de gerência de redes se mostrou extremamente eficiente, já que permitiu que o coletor fosse executado em poucos minutos no seu pior caso, enquanto que os algoritmos estudados em outros trabalhos, por não ter um conhecimento prévio dos componentes da rede, demorariam horas para redes com o tamanho da rede da UFRGS. O fato de termos o conhecimento prévio dos componentes da rede foi crucial para o resultado final do trabalho.

4 DECISÕES DE PROJETO

Durante o desenvolvimento do trabalho, vários experimentos geraram mudanças que alteraram o resultado final da ferramenta. Este capítulo tem como objetivo descrever as principais mudanças, bem como as decisões e erros que geraram novos caminhos até o estabelecimento da versão da ferramenta apresentada neste trabalho.

4.1 Tipo de dados para chaves na base temporal

Houve um estudo mais refinado para a definição do tipo de dado que representaria a temporalidade de um registro na base temporal. Em um primeiro momento, o estudo foi sobre a utilização do tipo *timestamp* (*8 bytes*) ou do tipo *integer* (*4 bytes*) para representar o dado temporal. Cada possibilidade tinha as suas vantagens: enquanto que a utilização do timestamp fornece uma referência direta ao dado temporal, o uso de um inteiro apontando para uma tabela auxiliar reduziria o tamanho do banco e dos índices criados no banco para agilizar as consultas.

A decisão neste ponto foi a utilização de dois inteiros para representar o período de validade de um registro, bem como a criação de uma estrutura auxiliar que determinaria o valor temporal de cada inteiro. Essa estrutura é a tabela *CollectTime*, e ela recebe um novo registro de coleta a cada execução do coletor. O uso dessa solução trouxe ainda duas novas vantagens:

- A possibilidade de em outras etapas de desenvolvimento criar diversos tipos de coleta com períodos diferenciados e com a possibilidade de filtrar apenas os tipos de coleta desejados;
- Uma consulta totalmente baseada em inteiros, já que os intervalos informados pelo usuário na interface, podem ser convertidos para inteiros.

Após a decisão do uso de inteiros, o estudo seguiu outro caminho, analisando o tipo de inteiro que seria utilizado para a representação do intervalo: integer ou small int.

Um inteiro é representado com 4 bytes dentro do SGBD, permitindo valores entre *0* e *2^31-1 (2.147.483.647)*, enquanto que o tipo small int é representado com 2 bytes, permitindo valores entre *0* e *2^15-1 (32,767)*. Infelizmente, o Postgre não possui implementação de *unsigned*, o que nos restringe ao universo positivo dos números.

Para determinar o tipo de dado que seria utilizado foi feito um cálculo simples buscando saber a quantidade de coletas que poderiam ser feitas com o menor dos dois tipos, ou seja, com *smallint*. Os resultados estão ilustrados na Tabela 4.1 a seguir.

Tabela 4.1: Número de dias até término do sequencial

Intervalo de Coleta	Coletas Diárias	Dias para ID smallint
5 min	288	113 dias
10 min	144	227 dias
15 min	96	342 dias
30 min	48	682 dias

Com os resultados encontrados, foi fácil constatar que teríamos menos de 2 anos de coleta antes de estourar o tamanho máximo do dado (ou aproximadamente 4 anos se utilizássemos números negativos). Como uma das ideias da ferramenta é ter coletas do tempo que for necessário, e talvez **N** tipos de coleta, a decisão foi utilizar um tipo *integer* para representar a validade temporal do dado armazenado, o que nos garante cerca de 120 mil anos de coleta, no caso de um tipo de coleta de 30 em 30 minutos.

4.2 Forma de execução da coleta

Nos primeiros testes de execução da coleta, a ideia era executar a coleta utilizando o comando *snmprealwalk* do *PHP*. Esse comando utiliza as chamadas de *getNext* da suite *Net-SNMP* para retornar o valor de todas as variáveis de um objeto de uma *MIB*. Dessa forma, o coletor faria apenas uma chamada simples em cima do identificador do objeto e da comunidade de cada um dos switches cadastrados, retornando todos os endereços físicos que tiveram pacotes encaminhados por aquele switch.

Porém, durante os primeiros testes, essa solução apresentou diversos problemas. Para começar, na maioria dos switches essa chamada demorou entre 2~10 minutos para terminar. Mesmo em um ambiente de coleta com processos distribuídos, pensando no resultado final com um coletor de 250~300 switches, esses tempos inviabilizariam um intervalo de coleta de 30 minutos.

A solução encontrada foi limitar o escopo da coleta. Ao invés de usar a chamada *PHP* citada anteriormente, o coletor passa a buscar apenas os dados que são interessantes. Para isso, usamos os dados do sistema de gerência de redes da UFRGS como limitador da busca.

Assim, avançamos mais um nível na *MIB*, buscando diretamente as variáveis e não toda a tabela. Para saber o conjunto de dados que devem ser buscados, recuperamos todos os endereços físicos de registros da subrede do switch em questão, bem como todos os endereços físicos de todos os switches do sistema. Essa consulta se mostrou extremamente rápida, demorando menos de 1 minuto em qualquer caso, e, além disso, retorna um conjunto bem aproximado da realidade que nos interessa, ou seja, os registros que são gerenciados pelo sistema de gerência da UFRGS.

4.3 Divisão da interface gráfica

A primeira versão da interface gráfica para visualização do grafo apresentava grafo e filtros na mesma tela. Isso era interessante para se editar as opções de filtros de dados de um grafo e renderizá-lo novamente. A tela seguia o esquema da Figura 4.1 a seguir.

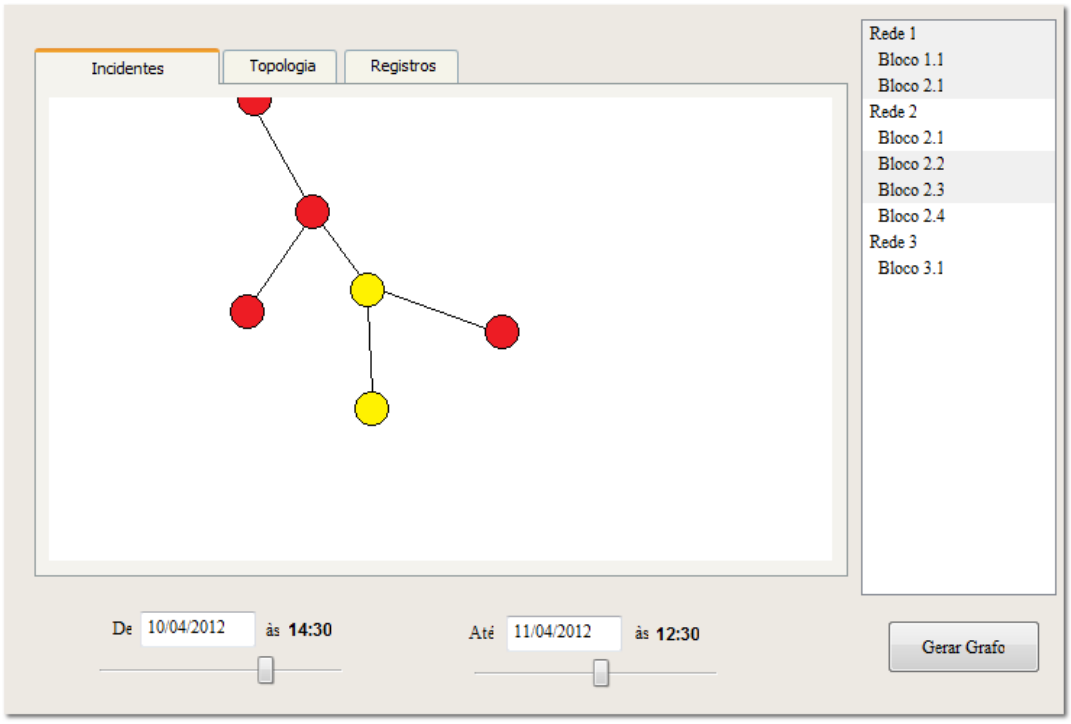

Figura 4.1: Organização Inicial da tela do sistema

Porém, durante o desenvolvimento dessa interface, uma questão foi levantada: não seria interessante ter a possibilidade de gerar vários grafos a partir da mesma execução da ferramenta?

Para cobrir essa questão, a tela de especificação dos filtros do grafo foi separada da tela de visualização do grafo. Assim, ao gerar um novo grafo, uma nova tela é aberta com a visualização do grafo, conforme detalhado no Capítulo 4 que descreve a ferramenta.

4.4 Adição de grafos diferenciados

Apesar de o trabalho ter seu centro na coleta de dados de switches e na geração de um grafo da topologia dos switches da rede da UFRGS, o sistema de gerência da UFRGS possui muitos dados passíveis de ser visualizados de forma gráfica.

Por esse motivo, e para garantir a possibilidade do uso do Prefuse muito antes do final da etapa de coleta de dados dos switches, foram definidos outros dois grafos passíveis de serem gerados pela ferramenta: um grafo da estrutura de blocos do sistema e um grafo com os incidentes de segurança.

4.5 Alteração da aplicação de desktop para web

A última mudança na ferramenta foi uma das mais impactantes: a mudança do ambiente da visualização de grafos. No início do trabalho, a ideia era trabalhar com uma aplicação em desktop, utilizando Java Swing para a implementação da interface. Porém, no decorrer do desenvolvimento, percebemos que uma ferramenta desktop ia contra a ideia do sistema web de gerência de redes da UFRGS.

Foi feito um estudo para analisar as funcionalidades que seriam perdidas e/ou alteradas ao fazer essa migração. As mudanças seriam:

1. *Performance*: a geração de grafos em applets para a web perde em performance quando comparada a aplicações java swing, porém não existem estudos que apontem uma perda significativa.
2. *Interação*: a comunicação entre applet e PHP é possível, mas subiria muito a complexidade do desenvolvimento da ferramenta, e por esse motivo, essa etapa será desenvolvida posteriormente. Dessa forma, o applet é responsável apenas por renderizar o grafo e o PHP lista as informações sobre todos os nodos, não havendo ainda comunicação entre essas duas partes.
3. *Usabilidade*: por todo o sistema de gerência de redes da UFRGS ser implementado em PHP, desenvolver esta nova ferramenta também em PHP tornará ela muito mais útil e usável dentro do sistema já existente, além de facilitar posteriormente a manutenção.

Baseado principalmente neste último ponto, a escolha foi portar a ferramenta para PHP com o uso de applets.

5 CONCLUSÕES

Com este trabalho, definimos e apresentamos uma ferramenta de coleta e visualização de switches: o SiViCS, Sistema para Visualização e Coleta de Switches. A ferramenta foi definida em três camadas: um coletor de dados de switches escrito em PHP, uma base de dados temporal implementada em Postgre e uma interface web para visualização de grafos utilizando PHP e Java.

Como caso de uso foi utilizada a rede da UFRGS e como base para a ferramenta o sistema de gerência de redes desenvolvida e utilizada pela UFRGS. Apesar de o enfoque ser a rede da UFRGS, a ideia de coleta de topologia de switches apresentada pode ser expandida para outras redes, e as ferramentas que foram propostas podem ser adaptadas às estruturas de outros sistemas de gerência de redes, inclusive para fazer a coleta de dispositivos diferentes de switches.

Nem todas as metas do início do trabalho puderam ser atingidas. O coletor e a base temporal foram implementados de acordo com a definição inicial, mas a ideia inicial da visualização gráfica era apresentar a estrutura topológica dos switches entre si, e não somente as máquinas ligadas diretamente as suas portas. A necessidade de esta interface ser apresentada em um mesmo ambiente que o sistema de gerência da UFRGS, ou seja, na web, também fez com que muitas de suas funcionalidades fossem retiradas para permitir que a interface fosse implementável no ambiente de desenvolvimento do CPD da UFRGS. Uma funcionalidade que estava prevista era o deslocamento do grafo no tempo, sendo assim possível visualizar suas mudanças ao longo do tempo de forma mais fácil e intuitiva. Por esses motivos, no estado atual de desenvolvimento, a interface de visualização não é capaz de nos responder as perguntas que foram apresentadas na introdução deste trabalho, que eram *‘Por onde esteve esta máquina antes de me causar problemas aqui?’* ou *‘Quais dos meus switches estão com overhead de tráfego?’*.

Apesar da interface de visualização não ter chegado ao estado definido no início do projeto, a ferramenta como um todo representa uma adição importante ao sistema de gerência de redes da UFRGS. Até o fim deste trabalho, a ferramenta não conta com nenhuma funcionalidade que permita visualização gráfica dos dados em forma de grafo. Além disso, o SiViCS representa um passo no processo de uso de informações coletadas da rede para fornecer mais informações e facilitar a gerência das subredes da UFRGS.

A ferramenta também mostra que uma coleta de dados de dispositivos de rede pode e deve utilizar de qualquer informação de seu sistema de gerência, como foi visto no caso do coletor do SiViCS, que pode apresentar uma performance muito melhor por utilizar da estrutura dos dados existentes no sistema de gerência para direcionar a sua coleta.

Como trabalho futuros podemos citar a implementação de uma nova interface de visualização. A interface desenvolvida neste trabalho será descontinuada, e novas soluções serão buscadas sem a utilização de java ou ainda utilizando java, porém tentando encontrar meios de executar uma comunicação entre applet e banco de dados de forma segura e que não onere a base de dados da UFRGS. Também podemos citar como trabalho futuro a expansão da coleta de dados para outros tipos de dispositivos, e a definição de coletas diferenciadas para redes com maior tráfego de dados.

REFERÊNCIAS

EDELWEISS, Nina. **Bancos de dados temporais : teoria e pratica**. Porto Alegre, UFRGS, 1998, In: Jornada de Atualizacao em Informatica (17. : 1998, ago. 4-7 : Belo Horizonte, Bmg). Anais. Belo Horizonte : Sbc, 1998.

JENSEN, C. S.; et al. **The Consensus Glossary of Temporal Database Concepts** - February 1998 Version. In: Temporal Databases: research and practice. Berlin: Springer-Verlag, 1998. p.367–405. (Lecture Notes in Computer Science, v.1399).

HUBLER, Patrícia Nogueira. **Definição de um Gerenciador para o Modelo de Dados Temporal TF-ORM.** Dissertaçao (Mestrado em Ciência de Computação) - Instituto de Informática, UFRGS, Porto Alegre, 2000.

ELMASRI, Ramez; NAVATHE, Shamkant B. **Fundamentals of Database Systems**. 3rd edition, Addison Wesley Publishing Company, 1999.

HEER, Jeffrey; CARD, Stuart K.; LANDAY, James A. **prefuse: a toolkit for interactive information visualization**. Em *CM Human Factors in Computing Systems (CHI), 421-430, 2005*

BEJERANO, Yigal; BREITBART, Yuri; GAROFALAKIS, Minos; RASTOGI, Rajeev. **Physical Topology Discovery for Large Multi-Subnet Networks**. Em. IEEE Infocom 2003. San Francisco, USA, 2003.

BREITBART, Yuri; GAROFALAKIS, Minos; **Topology Discovery in Heterogeneous IP Networks: The NetInventory System.** Em. IEEE/ACM TRANSACTIONS ON NETWORKING, VOL. 12, NO. 3, JUNE 2004, pg. 401-414.

STALLING, William. **SNMP, SNMPv2, SNMPv3 and RMON 1 and RMON 2 3rd edition.** Addison Wesley Publishing Company, 1999.

CARISSIMI, Alexandre; ROCHOL, Juergen; GRANVILLE, Lisandro. **Redes de Computadores.** 1ª edição, Bookman, 2009.

MACHADO, C.; MARQUEZAN, C.; REY, L.; SOARES, D.; POSTAL, E.;

HOROWITZ, E.; ZIULKOSKI, L. **Sistema Registro de Estações da UFRGS.** No II Workshop de Tecnologia das IFES, Gramado, RS - Brasil, Maio de 2008.

MACHADO, C.; REY, L.; SOARES, D.; CERON, J; BOOS, A. **Implantação do Sistema de Registro de Estações da UFRGS.** No III Workshop de Tecnologia das IFES, Belém, PA - Brasil, Maio de 2009.

MACHADO, C.; REY, L.; SOARES, D.; FIALHO, F.; SILVEIRA, R; RIBEIRO, R. **SABRE - Sistema Aberto de Registro de Estações.** No V Workshop de Tecnologia das IFES, Florianópolis, RS - Brasil, Maio de 2011.

OID Repository - http://www.oid-info.com/ (Acessado em 8 de maio de 2011)

IEETA - Instituto de Engenharia Electrónica e Telemática de Aveiro http://wiki.ieeta.pt/wiki/index.php/Main_Page (Acessado em 5 de maio de 2011)