Sistema de monitoramento de redes baseado nos protocolos SNMP e Spanning Tree

Antonio Matheus Benaion Esteves

Rio de Janeiro, RJ - Brasil
22 de Agosto de 2013

Sistema de monitoramento de redes baseado nos protocolos SNMP e Spanning Tree

Antonio Matheus Benaion Esteves

Dissertação apresentada à Coordenação de Formação Científica do Centro Brasileiro de Pesquisas Físicas para a obtenção do título de Mestre no programa Mestrado Profissional em Física com ênfase em Instrumentação Científica.

Esta versão contém as correções e alterações sugeridas pela Comissão Julgadora durante a defesa realizada por Antonio Matheus Benaion Esteves em 22/08/2013.

Comissão Julgadora:

- Prof. Dr. Nilton Alves Júnior - (CBPF - Orientador)
- Prof. Dr. Márcio Portes de Albuquerque - (CBPF)
- Prof. Dr. Luís Felipe Magalhães de Moraes - (COPPE/UFRJ)

Agradecimentos

Em primeiro lugar, gostaria de iniciar esta seção agradecendo à minha família, que sempre me apoiou e me incentivou a me aprimorar mais e mais ao longo de mais de 20 anos de árduos estudos até finalmente eu conseguir alcançar à apresentação deste trabalho, que marca minha conclusão do curso de Mestrado Profissional em Instrumentação Científica, em especial à minha amada esposa Jéssica dos Santos Benaion Esteves, que foi minha maior incentivadora nesta empreitada.

Além de minha esposa, declaro também toda a minha gratidão e meu mais sincero amor a Antonio de Oliveira Esteves, Elizabeth Benaion, Thiago Benaion Esteves, Carmen Benaion, Eliani Helena Benaion, Gilmar Benaion Lima, Ondilberto Vasques Marins (*in memoriam*), Paulo César Benaion (*in memoriam*) e Luiz Alberto Vasques Marins (*in memoriam*), pois a família é a base de qualquer conquista pessoal na vida de um homem, e vocês constituem tal base em minha vida.

Em segundo lugar, quero aqui manifestar minha gratidão a todos aqueles que compartilharam comigo o ambiente acadêmico ao longo destes mais de 3 anos em que estive no CBPF, sejam meus colegas, com quem compartilhei incontáveis momentos de amizade, estudo, conhecimento e descontração durante o curso, sejam meus professores, os quais tiveram papel fundamental para que eu pudesse adquirir não apenas os conhecimentos necessários para a realização deste curso, mas também contribuiram para que eu pudesse compreender o que o Mestrado é de fato, e sua importância, tanto no aspecto profissional, quanto no aspecto da pesquisa científica. Em especial, minha gratidão aos estimados professores Nilton Alves Jr., Marcio Portes de Albuquerque e Luís Felipe Magalhães de Moraes, que me concederam

a honra de formar a banca examinadora que avaliará meu trabalho. Não citarei nominalmente todos os colegas de Mestrado, nem os professores, pois não quero correr o risco de cometer a injustiça de esquecer o nome de algum deles, quando todos foram de alguma forma relevantes em minha passagem pelo CBPF.

E, por fim, expressarei aqui o desejo de manifestar meu apreço e minha profunda gratidão por todos os amigos que fiz ao longo destes meus 31 anos de caminhada neste mundo, amigos estes que, com sua amizade, consideração e palavras de apoio e camaradagem, têm feito significativa diferença em minha vida para que eu me torne não apenas um profissional melhor, mas também um ser humano melhor e mais atento às coisas e às pessoas à minha volta.

Resumo

Esta dissertação descreve a implementação de um sistema de monitoramento de equipamentos de rede para o projeto *Redes Comunitárias de Educação e Pesquisa* - Redecomep, implementado pela *Rede Nacional de Pesquisas* - RNP, financiado pela agência *Financiadora de Estudos e Projetos* - FINEP, e coordenado, no Rio de Janeiro, pela *Fundação de Amparo à Pesquisa do Estado do Rio de Janeiro* - FAPERJ. Este sistema visa monitorar os *backbones* da RedeRio e da Redecomep, mostrando como estes se comunicam.

Inicialmente, são apresentadas as motivações para a atividade gerência de redes e para o desenvolvimento do sistema descrito nesta dissertação, incluindo alguns exemplos de sistemas de monitoramento já implementados. Em seguida, é feita uma introdução sobre alguns conceitos básicos de redes, bem como sobre os protocolos envolvidos neste projeto: o protocolo de gerência de equipamentos de rede *Simple Network Management Protocol* - SNMP, a base de dados onde o protocolo SNMP opera - *Management Information Base* - MIB, e o protocolo de roteamento *Spanning Tree*, que será utilizado na segunda parte do projeto.

Após isto, seguem as atividades realizadas em cada parte do projeto, junto com as ferramentas utilizadas, e a implementação do sistema de monitoramento. Em seguida são mostrados os resultados obtidos, e as conclusões acerca do que foi realizado neste projeto.

Palavras-chave: Monitoramento de redes, Redecomep, *Spanning Tree*, SNMP, MIB, LAMP.

Abstract

This dissertation describes a monitoring system's implementation for project *Redes Comunitárias de Educação e Pesquisa - Communitary Networks for Education and Research* - Redecomep, developed by *Rede Nacional de Pesquisas - National Network for Researches* - RNP, funded by *Financiadora de Estudos e Projetos - Studies and Projects Funder* - FINEP agency, and coordinated in Rio de Janeiro by *Fundação de Amparo à Pesquisa do Estado do Rio de Janeiro - Research Support Foundation in Rio de Janeiro State* - FAPERJ. This system intends to monitoring RedeRio's and Redecomep's *backbones*, showing the communication between them.

First of all, the reasons to perform the activity of network management and this system's development are exposed, including some examples of existing monitoring systems. After that, this text make an introduction about some basic networks' concepts and the protocols related to this project: the network devices management protocol *Simple Network Management Protocol* - SNMP, the database where the SNMP protocol works - *Management Information Base* - MIB, and routing protocol *Spanning Tree*, which will be used in this project's second part.

After this, follow the activities made in each project's part, together with the used tools, and the monitoring system's implementation. Following this, the results and the final considerations are exopsed.

Keywords: Networks monitoring, Redecomep, *Spanning Tree*, SNMP, MIB, LAMP.

Conteúdo

Lista de Acrônimos

API	Application Programming Interface
ARPANET	Advanced Research Projects Agency Network
AS	Autonomous System
ASN.1	Abstract Syntax Notation One
BPDU	Bridge Protocol Data Units
CBPF	Centro Brasileiro de Pesquisas Físicas
CCITT	Comité Consultatif International Téléphonique et Télégraphique
CERN	Organisation Européene pour la Recherche Nucléaire
CPU	Central Processing Unit
DEC	Digital Equipment Corporation
EGP	Exterior Gateway Protocol
FAPERJ	Fundação de Amparo à Pesquisa do Estado do Rio de Janeiro
FDDI	Fiber Distributed Data Interface
FIFO	First In First Out
FINEP	Financiadora de Estudos e Projetos
GPL	General Public License
GRRW	Gerência da RedeRio via Web
HTML	Hipertext Markup Language
HTTP	Hipertext Tranfer Protocol
IETF	Internet Engineering Task Force
ICMP	Internet Control Message Protocol
IOS	Internetwork Operating System
IP	Internet Protocol
ISA	Industry Standard Architecture
ISO	International Organization for Standardization

LAMP	Linux, Apache, MySQL e Linux
LAN	Local Area Network
LIC	Laboratório de Instrumentação Científica
MAN	Metropolitan Area Network
MAST	Museu de Astronomia e Ciências Afins
MCT	Ministério da Ciência e Tecnologia
MRTG	Multi Router Traffic Grapher
MST	Minimum Spanning Tree
MSTP	Multiple Spanning Tree Protocol
NIC	Network Interface Card
OID	Object Identifier
ON	Observatório Nacional
OSI	Open Systems Interconnection
PAN	Personal Area Network
PCI	Peripheral Component Interconnect
PHP	Hipertext Preprocessor
PING	Packet Internet Grouper
PNG	Portable Network Graphics
PPP	Point to Point Protocol
PVST	Per-Vlan Spanning Tree
PVST+	Per-Vlan Spanning Tree Plus
RAM	Random Access Memory
Redecomep	Redes Comunitárias de Educação e Pesquisa
RFC	Request for Comments
RNP	Rede Nacional de Pesquisas
RRDTool	Round-Robin Database Tool
RSTP	Rapid Spanning Tree Protocol
RTT	Round Trip Time
SGBD	Sistema Gerenciador de Bancos de Dados
SMI	Structure of Management Information
SMTP	Simple Mail Transfer Protocol
SNMP	Simple Network Management Protocol
SQL	Structured Query Language

STP	Spanning Tree Protocol
TCA	Topology Change Notification Acknowledgment
TCN	Topology Change Nofitication
TCP	Transmission Control Protocol
TTL	Time To Live
UDP	User Datagram Protocol
USB	Universal Serial Bus
UIT	Union Internationale des Télécommunications
WAN	Wide Area Network

Lista de Figuras

Lista de Tabelas

Capítulo 1

Introdução

1.1 Motivações para gerência de redes

Para entender o que levou à realização do sistema de monitoramento descrito nesta dissertação, primeiro é preciso compreender o que é a atividade de gerência de redes, o que a motiva e as questões levantadas pelo advento da mesma.

No mundo atual, conhecimento é um aspecto cada vez mais relevante para detectar necessidades e buscar soluções que as atendam. Também é preciso desenvolver formas de disseminar e gerenciar as informações geradas ao longo do tempo, de modo a evitar o retrabalho de se buscar respostas para questões que outros já conseguiram resolver de forma satisfatória. Dentro disto, surge o conceito de redes, onde pessoas e instituições podem compartilhar conhecimento, a fim de solucionar problemas diversos [1].

Ao mesmo tempo em que surge a necessidade de criar redes, nos moldes descritos no parágrafo anterior, aparece também a questão de como estruturá-las, para que grandes distâncias físicas entre os componentes das redes não sejam empecilho às mesmas. Além disto, é preciso garantir que o acesso à informação e a comunicação entre usuários de rede sejam ágeis, viabilizando a rápida solução de problemas.

Para solucionar a primeira questão, iniciou-se a implantação de redes corporativas, utilizando tecnologias como cabos de cobre e fibra óptica, para estabelecer as ligações físicas entre os usuários (com uso de seus computadores para acessar tais redes); e protocolos específicos para gerenciar o funcionamento lógico destes sistemas. Inicialmente, estas redes partem de um contexto local, ligando pessoas em pequenas distâncias, aumentando gradativamente este alcance, de modo a conectar pessoas em qualquer lugar do mundo [2].

Já para a segunda questão, iniciou-se o aumento da largura de banda das linhas de transmissão de dados, agilizando o envio/recebimento de dados, independentemente das distâncias físicas envolvidas [3] [2].

O advento da ampliação da largura de banda trouxe significativo aumento no total de usuários de rede, o que vem mudando a forma como as pessoas se comunicam e compartilham informações. Isto tem trazido também discussões

sobre o que pode ser ou não compartilhado, onde duas visões se destacam, em termos de normas regulatórias de acesso à informação: a visão baseada em competitividade, onde há a preocupação em assegurar os direitos autorais de empresas sobre produtos e serviços oferecidos, combatendo a pirataria dos mesmos; e a perspectiva de mercadorias de interesse público, onde é ponderado que dados produtos/serviços devem ser disponibilizados de forma gratuita, ou a preços baixos, em função de sua importância para a sociedade [3].

Outro ponto relevante a ser debatido, com o aumento do tráfego de dados em rede ocasionado pelo aumento da capacidade de transmissão de dados, é como fazer com que os dados transmitidos cheguem a seus destinos rapidamente, sem circular indefinidamente pela rede, o que pode ocasionar congestionamento nesta (fenômeno conhecido como *loop* de rede). Uma potencial solução é o protocolo Spanning Tree, cujo algoritmo de funcionamento faz com que, a partir de um determinado ponto da rede, chamado de raiz, sejam determinados os caminhos para cada um dos demais pontos, envolvendo o menor custo possível; e estabelecendo um só caminho entre dois elementos quaisquer de uma rede [4] [2].

1.2 Motivações para este trabalho

O projeto *Redes Comunitárias de Educação e Pesquisa* - Redecomep é coordenado pela *Rede Nacional de Pesquisas* - RNP, com recursos da agência *Financiadora de Estudos e Projetos* - FINEP. Tem por objetivo principal promover a implantação de redes metropolitanas comunitárias em 26 cidades que abrigam pontos de presença (PoPs) do backbone da RNP. Antes do surgimento do projeto Redecomep, foi implementado na cidade de Belém o projeto Rede MetroBel, que veio a ser o primeiro projeto de rede metropolitana dedicada a fins acadêmicos desenvolvido no Brasil [5].

O projeto Redecomep contempla a implantação de infra-estrutura de fibras ópticas, equipamentos para a rede lógica e gestão de cada rede metropolitana. Estas atividades serão realizadas em parceria com as instituições de pesquisa e ensino superior das cidades onde estas redes serão implementadas.

Após a implantação, a gestão da operação das redes metropolitanas, bem como seu custeio e sua sustentabilidade ficarão sob a responsabilidade das instituições usuárias estimulando a formação de consórcios, na maioria dos casos para garantir a auto-sustentação deste projeto. Isto será feito através de programas de treinamento e capacitação na operação das redes ópticas, para o pessoal técnico das instituições de pesquisa e educação e dos PoPs da RNP.

Na cidade do Rio de Janeiro, o projeto Redecomep é coordenado pela agência *Fundação de Amparo à Pesquisa do Estado do Rio de Janeiro* - FAPERJ, em conjunto com a RNP. A coordenação técnica do projeto será de responsabilidade da atual equipe da coordenação de engenharia operacional do projeto RedeRio.

Para este projeto, foi estimado um investimento de R$ 9.000.000,00, para implementar uma rede com 303 km de extensão. Para isto, foram estabelecidas parcerias com diversas instituições, tais como: a Prefeitura do Rio de Janeiro, a companhia elétrica Light e as concessionárias MetrôRio, SuperVia e LAMSA [5].

Para que o projeto Redecomep possa ser realizado com êxito, juntamente com as diretrizes para a implementação desta dissertação, cabe a observação do Plano Diretor do CBPF, documento que contempla uma série de metas institucionais do referido órgão. Conforme podemos verificar neste documento, algumas dessas metas são [6]:

- Participar ativamente dos grandes projetos nacionais e internacionais nas áreas de computação e redes de alto desempenho em transmissão de dados;

- Manter a responsabilidade da administração da rede de computadores do Rio de Janeiro, participar da implantação do *backbone* Redecomep do Rio de Janeiro, e dar suporte às redes acadêmicas nacional, estadual e municipal;

- Prover a rede do CBPF com tecnologias de 10 Gbps para acesso externo, interno e conexões de alta velocidade específicas para projetos em grade, *cluster* e vídeo de alta performance, até 2015.

A figura 1.1 apresenta o projeto de rede implementado para o *backbone* Redecomep no Rio de Janeiro.

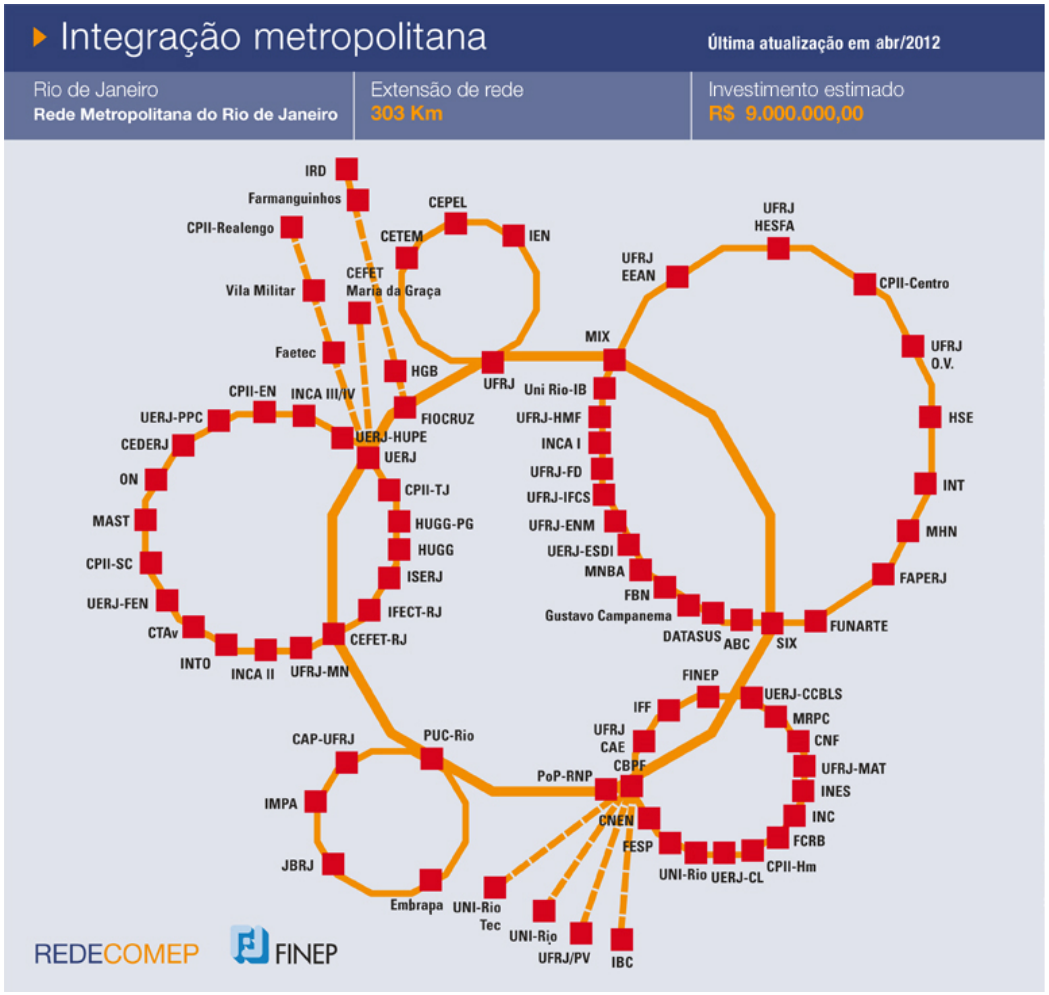


Figura 1.1: Mapa da Redecomep - obtido em www.redecomep.rnp.br

Para que metas como as listadas acima possam ser alcançadas com sucesso, é preciso desenvolver um ambiente integrado de rede, onde se possa garantir a plena comunicação entre diversas localidades e pontos de rede. Conforme observaremos ao longo deste trabalho, o principal propósito do mesmo consiste exatamente em conseguir monitorar a comunicação entre diversos pontos de rede, de modo a garantir o funcionamento da mesma.

1.3 Exemplos de sistemas de monitoramento de redes já existentes

Em função da relevância da atividade de monitoramento e gerência de redes, é possível encontrar diversas soluções de monitoramento prontas para utilização, bem como soluções documentadas na literatura de redes. A seguir, seguem alguns exemplos de soluções utilizadas:

- O aplicativo *Multi Router Traffic Grapher* - MRTG é uma ferramenta de monitoração de entrada e saída de dados em equipamentos de rede, que gera páginas HTML com gráficos de dados coletados, gráficos estes gravados em arquivos de imagens no formato PNG.

  O aplicativo MRTG costuma ser utilizado para monitorar tráfego de dados em equipamentos de rede, mas pode monitorar qualquer outra estatística, caso o equipamento de rede monitorado forneça os dados da mesma via protocolo SNMP ou por meio de um script gerado especificamente para este fim [7].

  O programa de coleta de dados do aplicativo MRTG é um script escrito em linguagem de programação Perl, enquanto que a criação dos gráficos fica a cargo de outro programa escrito em linguagem de programação C. É possível também, para efeito de melhoria de desempenho, integrar o aplicativo RRDTool ao aplicativo MRTG, utilizando-os em conjunto, integração esta utilizada no sistema desenvolvido para este trabalho de dissertação.

  Os dados a serem coletados, por sua vez, são obtidos dos equipamentos de rede via protocolo SNMP, através da rotina de coleta do aplicativo MRTG.

  A figura 1.2 a seguir apresenta um exemplo de geração de gráficos de monitoramento em uma dada interface de rede, através do aplicativo MRTG.

- O aplicativo Nagios, por sua vez, é fortemente baseado no conceito de *plugins*, para a monitoração de diversos serviços e protocolos de rede.

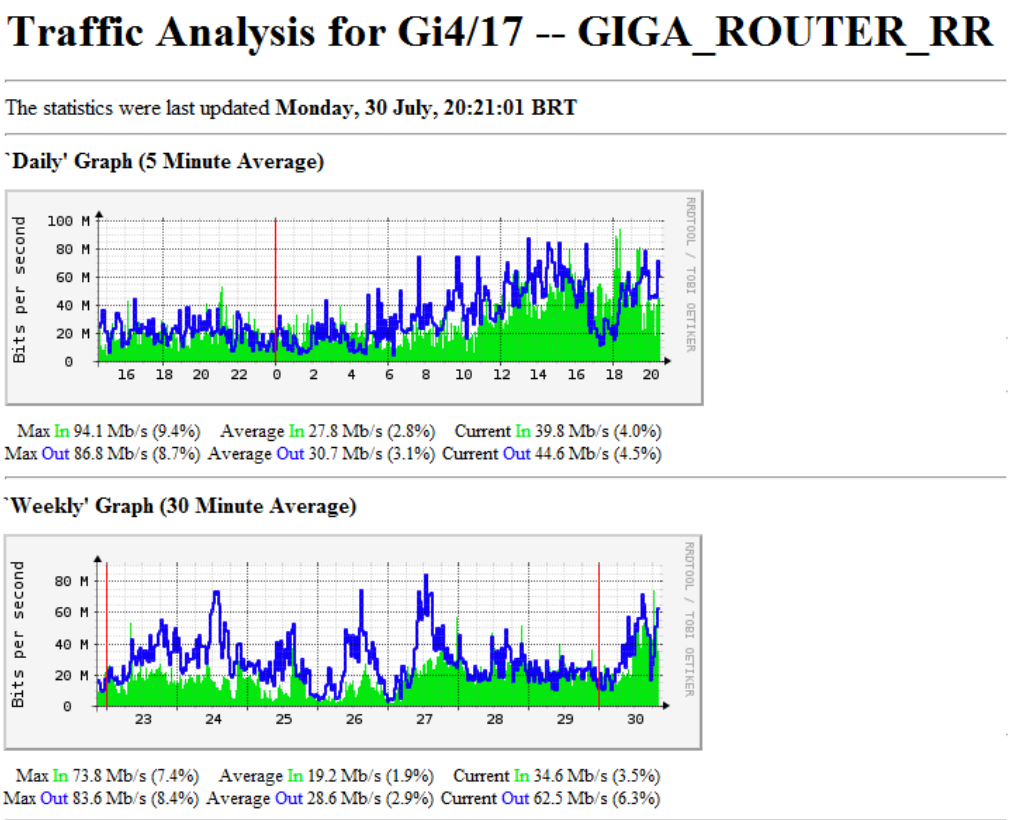


Figura 1.2: Geração de gráficos com o aplicativo MRTG

Graças a isto, diversas linguagens de programação podem ser utilizadas para monitorações baseadas em protocolos de rede como SNMP, SMTP, HTTP, etc [8].

Através do aplicativo Nagios, é possível se fazer também monitoramento de equipamentos de redes com o protocolo Spanning Tree, contudo, isto só é possível quando os equipamentos estão conectados diretamente ao *servidor* que suporta a aplicação, o que não é o caso dos equipamentos de rede monitorados pelo sistema desenvolvido para este projeto, inviabilizando a utilização deste aplicativo para a solução descrita nesta dissertação [9].

A figura 1.3 apresenta um exemplo de monitoramento básico de equipamentos de rede, através do aplicativo Nagios.

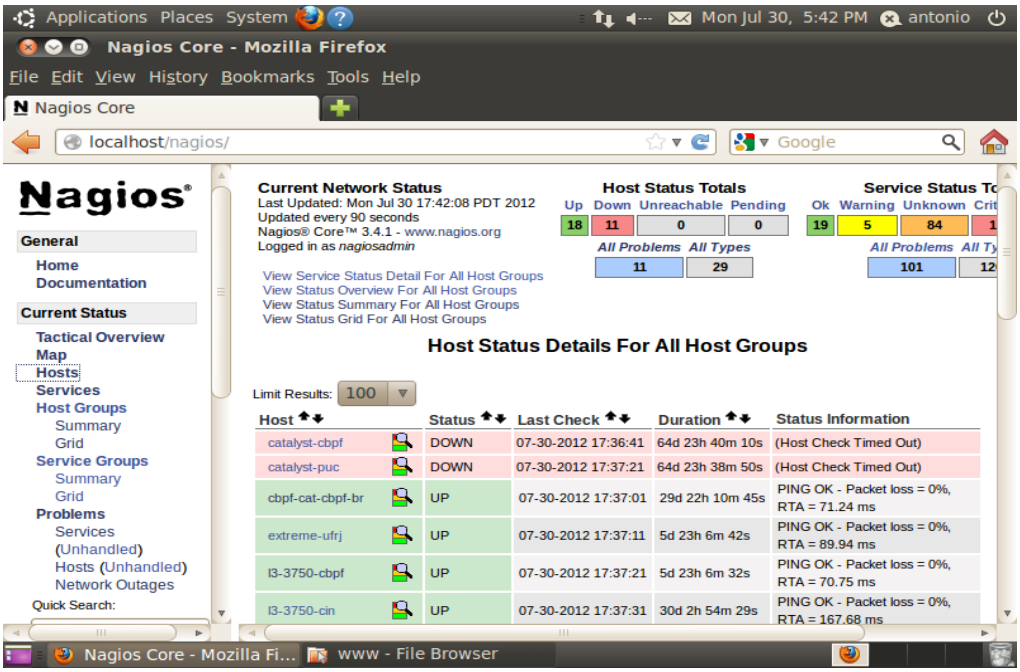


Figura 1.3: Interface de monitoramento do aplicativo Nagios

- Por fim, o aplicativo Cacti se vale do algoritmo de escalonamento Round-Robin na coleta de dados. Este algoritmo é empregado em sistemas operacionais, para execução de múltiplos processos simultaneamente, onde cada processo utiliza a CPU do computador durante certo intervalo de tempo, suficientemente reduzido para que todos os processos correntes possam ser executados sem ter que esperar pelo término de um processo em particular, garantindo que todos os equipamentos de rede monitorados possam ter suas estatísticas de interesse coletadas sem maiores prejuízos entre equipamentos distintos.

  O aplicativo Cacti apresenta uma página WEB indo além da exibição de gráficos, possibilitando ainda, através deste *front-end* desenvolvido em linguagem PHP, a instalação de plugins adicionais, como, por exemplo, o Weathermap, com o qual é possível se obter o atual estado geral de vários equipamentos de uma rede monitorada pelo aplicativo Cacti [10].

  A figura 1.4 apresenta um exemplo de monitoramento básico de equipamentos de rede, através do aplicativo Cacti.

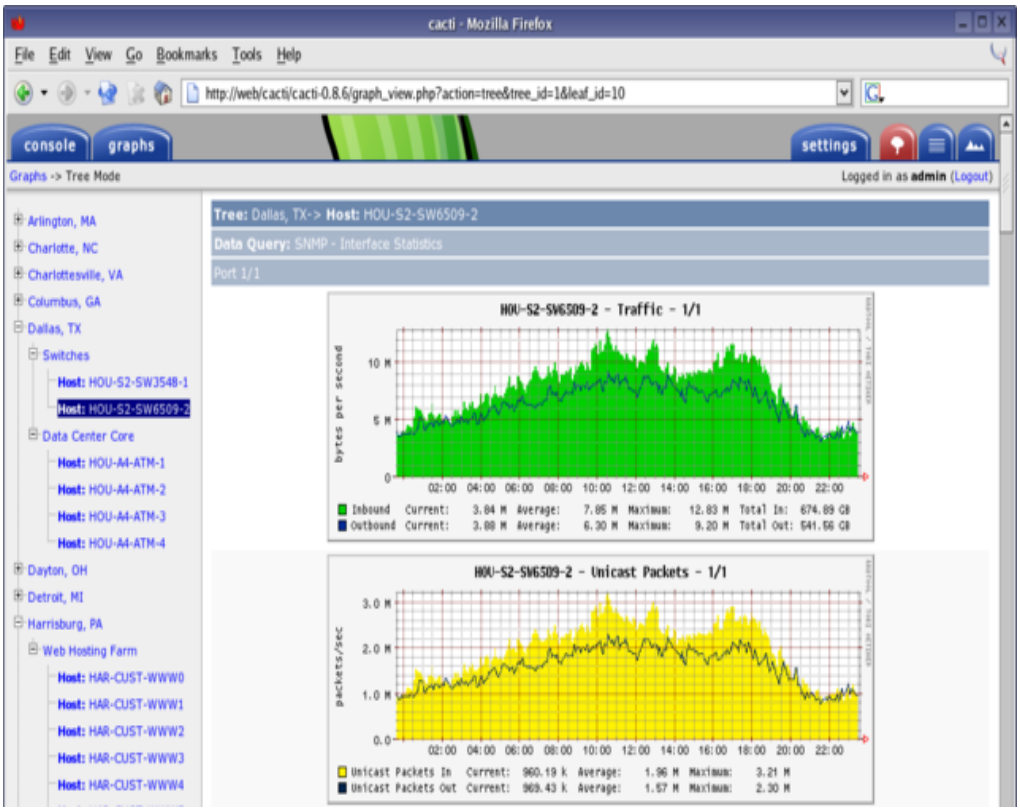


Figura 1.4: Interface de monitoramento do aplicativo Cacti

Como exemplos de sistemas WEB de monitoramento de redes, podemos destacar:

- O sistema GRRW, desenvolvido pelo CBPF para monitorar o tráfego de dados nos equipamentos de rede utilizados no *backbone* da RedeRio, assim como controlar dados cadastrais/técnicos das instituições que os utilizam, dentre outros aspectos de gerência de redes. Estes aspectos, os quais serviram de base para a primeira fase do sistema de monitoramento desenvolvido para esta dissertação, serão explicados com riqueza de detalhes mais adiante, na seção 3.4.1.

  A figura 1.5 apresenta a tela inicial do sistema GRRW.

  A figura 1.6 apresenta a tela de detalhe de de instituição cadastrada no sistema GRRW;

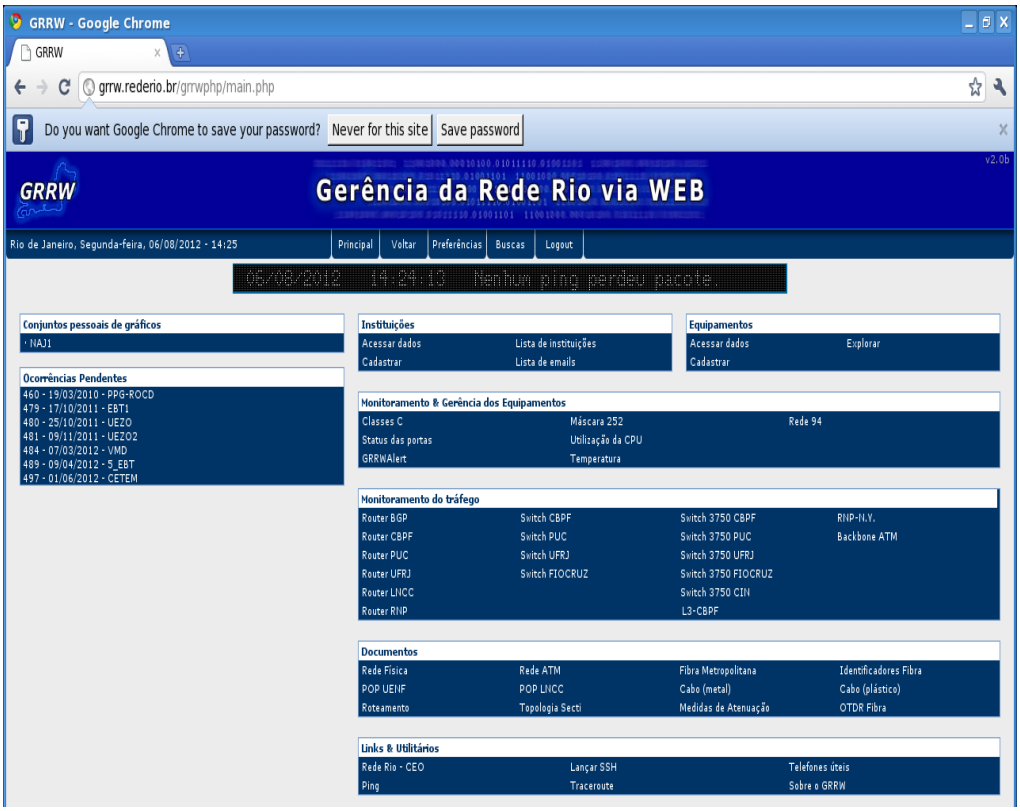


Figura 1.5: Tela inicial do sistema GRRW

- O sistema CERN Gridmap foi desenvolvido pela *Organisation Européene pour la Recherche Nucléaire* - CERN, a fim de monitorar a disponibilidade de acesso dos *sites* e equipamentos de redes de diversas instituições de ensino e pesquisa ao redor do mundo. Esta monitoração aponta o status operacional destes *sites*/equipamentos, a quantidade de elementos conectados a cada um deles, etc. Os status podem ser classificados como [11]:
  - **OK**: Indicado pela cor verde, atesta que não há problemas no momento;
  - **Warning**: Indicado pela cor verde claro, mostra que o sistema em questão segue funcionando, mas inspira cuidados por parte de seus mantenedores;
  - **Critical**: Indicado pela cor vermelha, mostra operação sob condi-

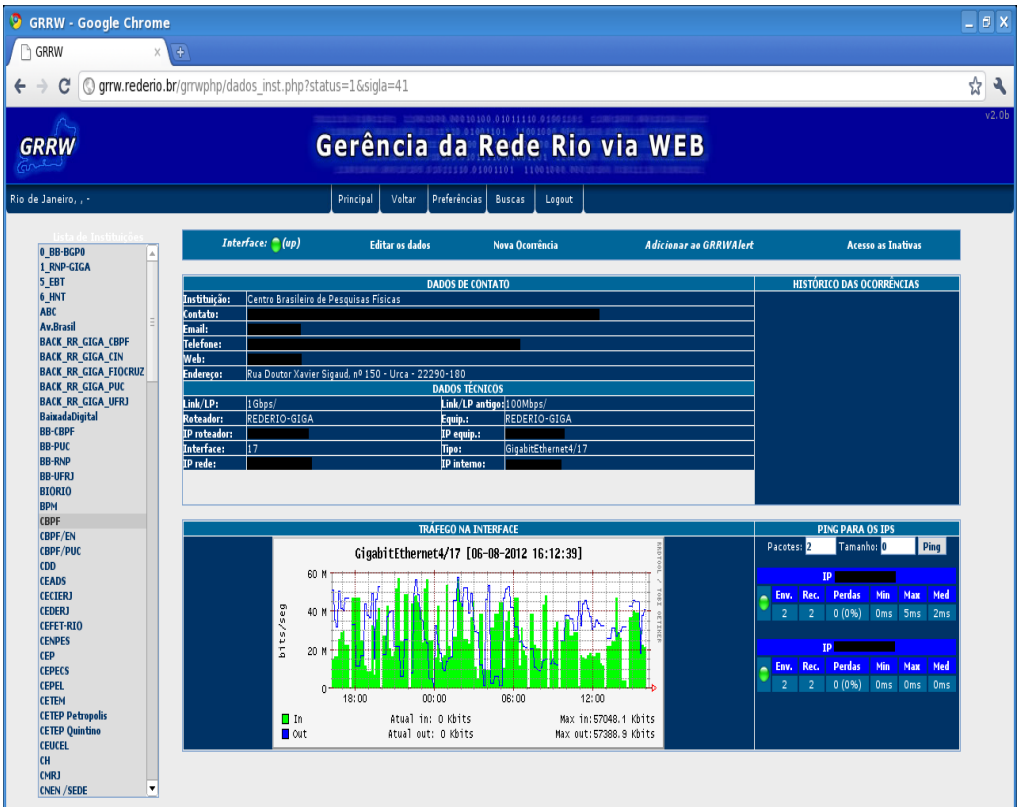


Figura 1.6: Detalhe de instituição cadastrada no sistema GRRW

ções críticas, demandando cuidado urgente por parte de seus mantenedores;

– **Downtime**: Indicado pela cor vermelha, indica indisponibilidade no momento, possivelmente em manutenção;

– **Unknown**: Indicado pela cor cinza claro, atesta que o status atual do sistema é diferente dos padrões esperados para o mesmo;

– **Missing**: Indicado pela cor cinza, mostra que o status atual do *site*/equipamento está ausente da monitoração, sem estar removido da mesma;

– **N/A**: Indicado pela cor cinza escuro, mostra que o status indeterminado no momento;

– **Removed**: Indicado pela cor roxa, mostra que o sistema em questão foi removido da monitoração.

O aplicativo CERN Gridmap tenta determinar a disponibilidade atual de um determinado *site*/equipamento, com base na porcentagem de vezes em que se consegue estabelecer conexões com o mesmo, para envio e recebimento de mensagens (*Success Rate* - Taxa de acerto). Os status são determinados da seguinte forma [12]:

- Acima de 50% de acerto: **OK**;
- Entre 20 a 50% de acerto: **Warning**;
- Abaixo de 20% de acerto: **Critical**.

A figura 1.7 apresenta um exemplo de monitoramento através do aplicativo CERN Gridmap;

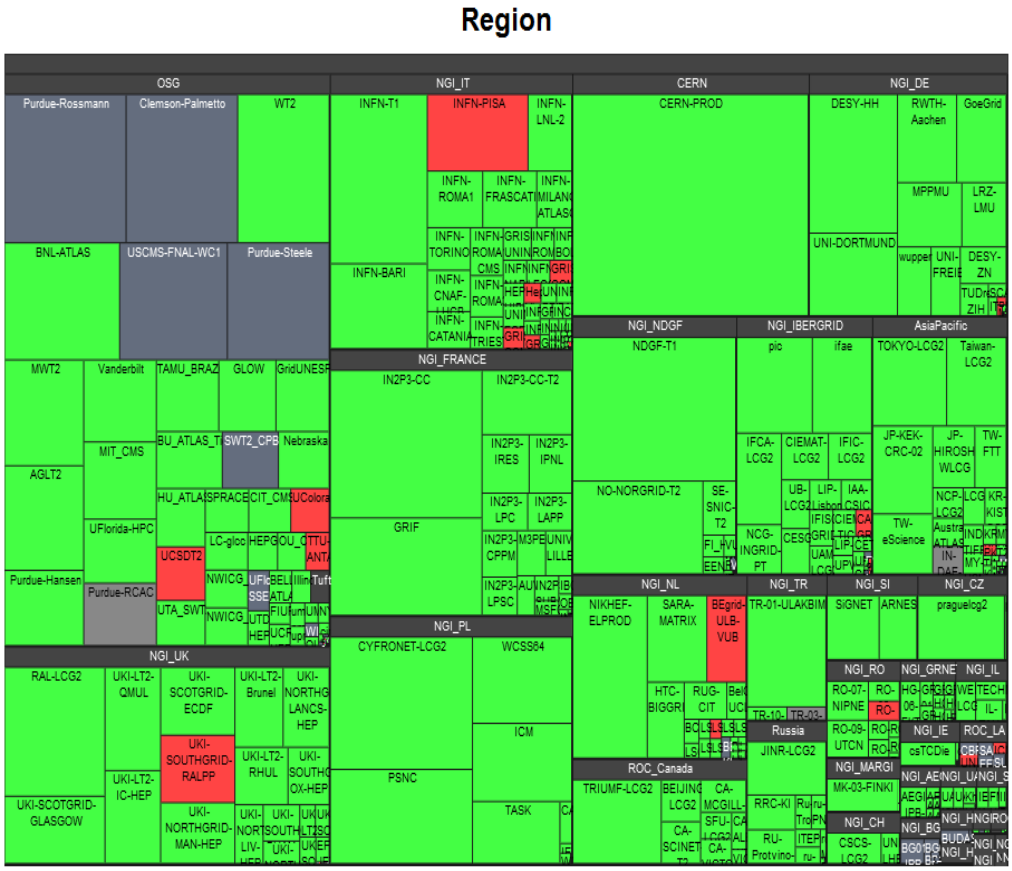


Figura 1.7: Interface CERN Gridmap - obtido em grid-monitoring.cern.ch

- O sistema BrazilFW consiste de uma distribuição Linux composta por um *firewall* e um roteador. Suas funções principais são compartilhar conexões de Internet e prover proteção de *firewall* para os computadores conectados à rede. O sistema BrazilFW não necessita de computadores de alta performance para funcionar. Um computador com pelo menos 16 MB (Mega Bytes) de memória RAM e duas placas de rede já é o suficiente para o pleno funcionamento do sistema.

  O sistema BrazilFW foi baseado no antigo sistema de *firewall* e roteamento do sistema operacional baseado na distribuição Coyote Linux. Com o sistema BrazilFW, é possível instalar serviços de rede como, por exemplo, compartilhamento de internet, *firewalls*, ou pontos de acesso sem fio. Embora seu foco principal continuem sendo as aplicações da distribuição Coyote Linux, ela agrega funções extras, procurando ainda manter a simplicidade na administração e nos requisitos de hardware para operação do sistema de monitoramento [13].

  A figura 1.8 ilustra a interface inicial de monitoramento do sistema BrazilFW.

Figura 1.8: Interface de monitoramento do aplicativo BrazilFW

Capítulo 2

Conceitos de redes, protocolos e aplicativos

2.1 Conceitos básicos de redes

2.1.1 Início

Para melhor entendimento do sistema descrito nesta dissertação, é necessário conhecer alguns conceitos sobre redes, bem como ter a compreensão acerca de alguns protocolos utilizados para o gerenciamento destas. Este capítulo abordará estes aspectos.

2.1.2 Protocolos

Um protocolo é definido como a descrição formal de como são realizadas as comunicações entre equipamentos de rede, explicitando os procedimentos a serem executados, a fim de obter a já mencionada comunicação, e assim conseguir fazer transferência de dados entre estes dispositivos.

Para melhor compreensão destes conceitos, podemos pensar na linguagem escrita, que possui regras gramaticais e vocabulário próprio, os quais possibilitam o entendimento de uma dada mensagem escrita nesta linguagem, por qualquer pessoa que a leia, contanto que esta conheça e entenda o funcionamento das regras utilizadas na linguagem em questão [14][15].

Uma vez definidas as regras do protocolo, este então é transformado em uma rotina a ser utilizada sempre que um dado tipo de comunicação for necessário entre dois ou mais equipamentos de rede, e este protocolo servir para esta comunicação. Os protocolos operam em todas as camadas do modelo de redes, desde as camadas lógicas, até a camada física da rede [15].

Alguns exemplos de protocolos utilizados são o *Hipertext Transfer Protocol* - HTTP, utilizado para o carregamento de páginas de Internet; o *Transmission Control Protocol* - TCP, utilizado para fazer transferência segura de pacotes, estabelecendo conexão prévia entre dois sistemas finais antes de iniciar o envio de pacotes de um a outro; o *User Datagram Protocol* - UDP, que realiza a mesma função do protocolo TCP, no entanto sem conexão prévia, e, portanto, fazendo transferência não-segura de pacotes; o *Internet Protocol* - IP, utilizado para fazer encaminhamento de dados de um ponto a outro na rede, sejam sistemas finais ou comutadores de pacotes; e o protocolo

Spanning Tree Protocol - STP, objeto de estudo desta dissertação, que calcula os caminhos ótimos entre diversos pontos de rede, a partir de um nó específico [15]. Para cada protocolo de rede criado, ou modificado, a organização IETF tem criado documentos de especificação denominados *Request for Comments* - RFC's, onde são abordados os aspectos técnicos e operacionais de cada protocolo atualmente operacional.

Esses e outros protocolos operam em camadas diversas de rede, camadas estas encarregadas de dar suporte para a realização bem-sucedida de todas as operações em uma rede. Estas camadas operam desde as atividades no ambiente físico, até as operações lógicas responsáveis pelo funcionamento dos aplicativos de Internet nos sistemas finais. Na seção 2.1.4, o conceito de camadas de rede será debatido com maior profundidade.

2.1.3 Estrutura de redes

Em uma definição simples, redes podem ser definidas como conjuntos de computadores conectados entre si, compartilhando diversos tipos de dados. Por outro lado, em uma definição mais aprofundada, estes computadores seriam apenas a parte "visível" das redes, aos olhos de um usuário convencional, sendo denominados como hospedeiros ou sistemas finais.

Estes sistemas finais são conectados uns aos outros através de equipamentos encarregados de repassar informações através das redes, com o auxílio de regras de roteamento de dados. Com estas regras, os equipamentos criam a rota, ou caminho de rede, de um dado pacote qualquer, decidindo como este "caminhará" pela rede, a partir do momento em que ele deixa o sistema final "fonte", até chegar ao "destino". A técnica de criação desta rota é conhecida por comutação de pacotes [15].

São exemplos de equipamentos: os roteadores e os *switches* (estes também conhecidos como comutadores), onde a principal diferença entre eles é o fato de que, enquanto os roteadores operam na camada de rede, os *switches* operam na camada de enlace. Mais adiante, serão explicados os conceitos relacionados às camadas onde os diversos elementos de rede operam.

Os comutadores de pacotes, por sua vez, conectam-se entre si e aos

sistemas finais através de enlaces de comunicação, ou *links* de comunicação. Estes enlaces se dividem em meios de transmissão com fio, onde existe a conexão física propriamente dita; e os meios sem fio (*wireless*), onde os dados são transmitidos em forma de diversos tipos de onda, sem uma conexão física determinando diretamente a forma de propagação da transmissão de dados.

Como exemplos de cada tipo de *links* de comunicação, temos [15]:

- Transmissão com fio: cabos coaxiais, cabos de par trançado, fibra óptica;
- Transmissão sem fio: ondas de rádio, via satélite, ondas infravermelhas.

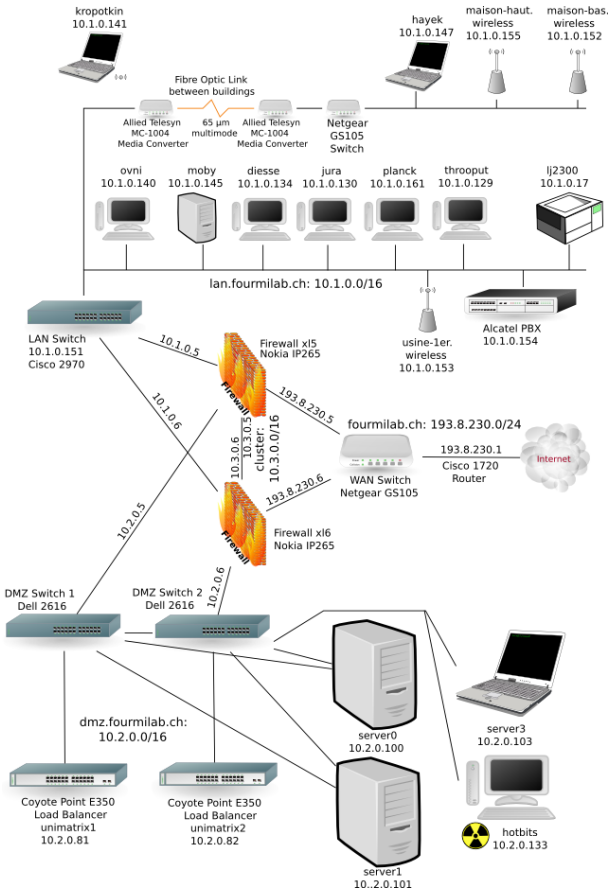


Figura 2.1: Exemplo de arquitetura de rede - obtido em www.fourmilab.ch

A figura 2.1 mostra um exemplo de arquitetura de uma rede, onde sistemas finais se conectam a equipamentos de rede (roteadores, *switches*, etc.),

através de *links* de comunicação, o que permite a estes sistemas finais se comunicarem com outros sistemas, eventualmente localizados em outras redes.

Com relação ao tamanho, em função de quantidade de equipamentos conectados, redes podem ser classificadas como [14]:

- ***Personal Area Network*** – **PAN**: caracteriza-se por um pequeno raio de atuação e número limitado de dispositivos conectados. Como exemplo, podemos citar uma rede utilizando um cabo USB, para conectar dispositivos como câmeras digitais ao computador;

- ***Local Area Network*** – **LAN**: utilizada para compartilhamento de dados e recursos de rede, tais como conexões à Internet, em espaços físicos delimitados, como escritórios, conjuntos de salas, ou ainda edifícios;

- ***Metropolitan Area Network*** – **MAN**: se encontra entre as redes LAN e WAN, diferenciando-se umas das outras pelo alcance geográfico que cada rede possui. São exemplos de redes MAN as universidades, onde é preciso criar estruturas (com ou sem cabeamento) para conectar diversos prédios a fim de efetuar o compartilhamento de dados e recursos de rede;

- ***Wide Area Network*** – **WAN**: a Internet é o principal exemplo de uma rede WAN, onde o alcance dos equipamentos conectados à rede pode se estender a todo o planeta. Surgiu da necessidade de se conectar equipamentos de rede em cidades, estados e países geograficamente muito distantes uns dos outros, o que não poderia ser contemplado com redes LAN, ou mesmo MAN, que apresentam restrições, no que concerne ao compartilhamento de dados a longas distâncias. A rede WAN apresenta como diferencial a existência, dentro de sua estrutura, de protocolos de comunicação que possibilitam o compartilhamento de dados em longas distâncias, tais como os protocolos Frame-Relay e Ponto-a-Ponto (PPP).

2.1.4 Camadas de rede

Atualmente existem dois principais modelos de arquitetura de camadas, servindo como referência para arquiteturas de rede. São eles: o modelo TCP/IP (mais utilizado para arquiteturas de redes), nome oriundo da fusão dos protocolos TCP e IP, com 5 camadas; e o modelo Interconexão de Sistemas Abertos (*Open Systems Interconnection*) - OSI, idealizado pela Organização Internacional para Padronização (*International Organization for Standardization*) - ISO, dividindo a arquitetura de rede em 7 camadas de operação [15][16].

No modelo TCP/IP, a arquitetura de rede se encontra divida nas seguintes camadas [15]:

- Aplicação: responsável pelos aplicativos de rede visualizados pelos sistemas finais, e os protocolos relacionados a estes. Opera somente nos sistemas finais da rede;
- Transporte: transporta as mensagens da camada de aplicação de um sistema final para outro. Também opera somente nos sistemas finais da rede, configurando os dados para serem passados à camada de rede;
- Rede: realiza a movimentação lógica de dados pela rede. Opera nos sistemas finais, e também em equipamentos intermediários da rede, calculando os caminhos a serem percorridos por um dado e repassando-os à camada de enlace;
- Enlace: realiza a movimentação física de dados pela rede, através de quadros da camada de enlace, chegando ao nível físico de operação na rede. Opera em todos os pontos da rede;
- Física: realiza a movimentação física de dados *bit*-a-bit pela rede. Opera em todos os pontos da rede.

A figura 2.2 ilustra o funcionamento do modelo TCP/IP. De cima para baixo temos: os aplicativos utilizados diretamente pelo usuário final; alguns exemplos de protocolos da camada de aplicação; outros protocolos, desta

vez da camada de transporte; o protocolo IP, único a operar na camada de rede; protocolos da camada de enlace; protocolos da camada física; e alguns exemplos de meios físicos empregados para transferência de dados.

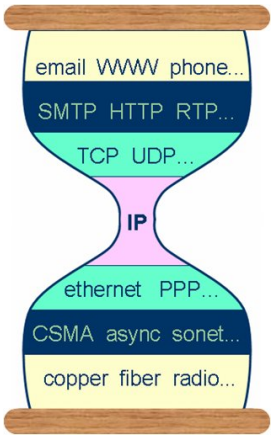


Figura 2.2: Modelo de arquitetura TCP/IP - obtido em gumuskaya.com

No modelo OSI, a arquitetura de rede se encontra divida nas seguintes camadas [16]:

- Aplicação: responsável pelos aplicativos de rede visualizados pelos sistemas finais, e os protocolos relacionados a estes. Opera somente nos sistemas finais da rede;

- Apresentação: responsável por traduzir os dados recebidos por um sistema final, vindos da camada de transporte para a camada de aplicação, e vice-versa, tarefa realizada pela camada de aplicação no modelo TCP/IP. Opera somente nos sistemas finais da rede;

- Sessão: responsável pelo estabelecimento de conexões entre sistemas finais, para transferência de dados, tarefa realizada pela camada de aplicação no modelo TCP/IP. Opera somente nos sistemas finais da rede;

- Transporte: transporta as mensagens da camada de aplicação de um sistema final para outro. Também opera somente nos sistemas finais

da rede, configurando os dados para serem passados à camada de rede, da mesma forma que a camada de transporte, no modelo TCP/IP;

- Rede: realiza a movimentação lógica de dados pela rede. Opera nos sistemas finais, e também em equipamentos intermediários da rede, calculando os caminhos a serem percorridos por um dado e repassando-os à camada de ligação de dados, da mesma forma que a camada de rede, no modelo TCP/IP;

- Ligação de dados: realiza a movimentação física de dados pela rede, através de quadros da camada de enlace, chegando ao nível físico de operação na rede. Opera em todos os pontos da rede, da mesma forma que a camada de enlace, no modelo TCP/IP;

- Física: realiza a movimentação física de dados *bit*-a-bit pela rede. Opera em todos os pontos da rede, da mesma forma que a camada física, no modelo TCP/IP.

A figura 2.3 apresenta um comparativo entre os modelos TCP/IP e OSI.

Conforme pode ser observado, a camada de aplicação do modelo TCP/IP, quando comparada com o modelo OSI, é fragmentada em três camadas: aplicação, apresentação e sessão. Outro ponto é o fato de que, em alguns casos na literatura de redes, as camadas física e de enlace do modelo TCP/IP aparecem como uma única camada, efetuando as atribuições de ambas.

A transição de dados entre camadas de rede é realizada através das práticas de encapsulamento e desencapsulamento de dados. Estes procedimentos ocorrem sempre que é necessário se fazer um pacote, datagrama, etc. passar a operar em uma camada de rede mais baixa, ou seja, mais próxima do meio físico e mais afastada do meio lógico (encapsulamento), ou quando é preciso fazer com que a operação passe a ocorrer em uma camada de rede mais alta, portanto mais próxima do meio lógico e mais distante do meio físico (desencapsulamento), para que o dado enviado opere na camada correta de rede [15].

A figura 2.4 ilustra os conceitos de encapsulamento/desencapsulamento de pacotes nas camadas de rede.

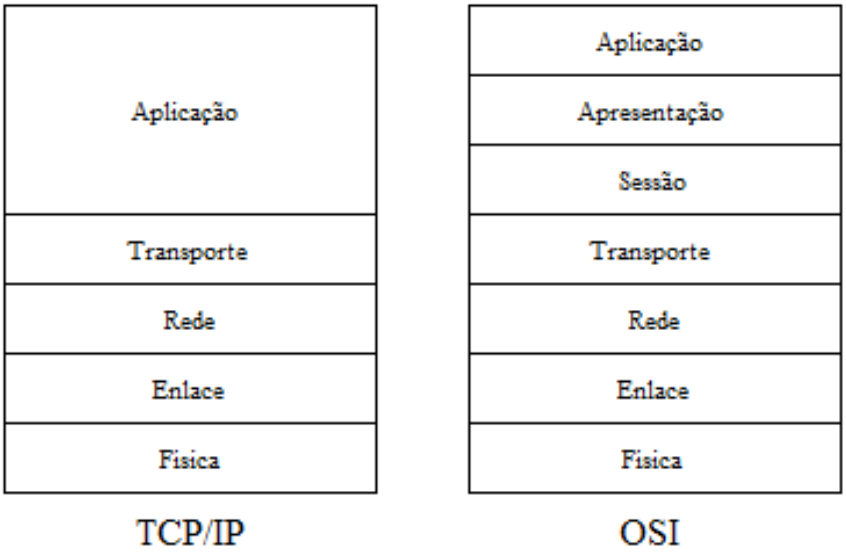


Figura 2.3: Comparação entre os modelos TCP/IP e OSI

Conforme mostrado na figura 2.4, quando uma dada mensagem é enviada de um terminal para outro, inicialmente esta mensagem, que se encontra na camada de aplicação da rede, é encapsulada, na seguinte ordem (modelo TCP/IP):

- Segmento (camada de transporte);
- Datagrama (camada de rede);
- Quadro (camada de enlace).

Em cada operação de encapsulamento, acrescenta-se um cabeçalho ao dado enviado pela camada anterior, para indicar a origem da mensagem em que esta operação foi realizada, para posterior desencapsulamento da mensagem em questão. Cada camada de rede acrescenta um cabeçalho específico, pertinente à mesma. Na figura 2.4, estes cabeçalhos se encontram representados nas nomenclaturas H_t (camada de transporte), H_n (camada de rede) e H_l (camada de enlace).

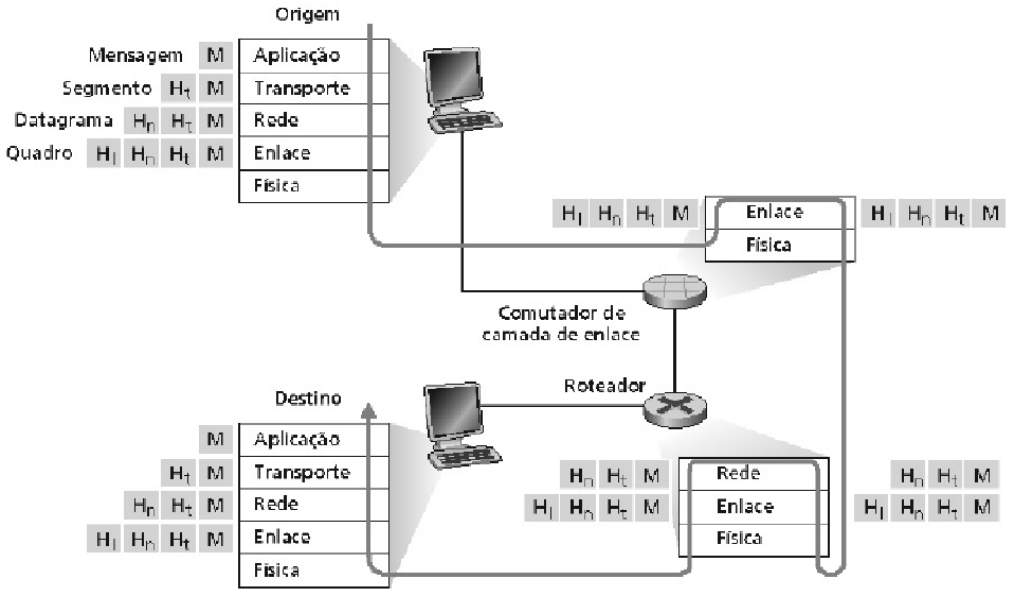


Figura 2.4: Encapsulamento/desencapsulamento - obtido em [15]

Cabe observar que, entre as camadas de enlace e física, não ocorre encapsulamento/desencapsulamento de dados, uma vez que a camada física realiza apenas operações físicas de transferência de dados.

Ao passar por diversos pontos de rede, os dados enviados serão ou não encapsulados/desencapsulados, dependendo da camada de rede em que o equipamento operacional em cada ponto atue. Por exemplo, ao passar por um comutador da camada de enlace, os dados enviados pela camada física não serão encapsulados/desencapsulados, devido ao fato de que tais operações não ocorrem entre as camadas de enlace e física. Por outro lado, ao passar por um roteador (operacional na camada de rede), o dado enviado será desencapsulado do formato de quadro, para que o roteador possa trabalhar com o segmento (formato utilizado na camada de rede) previamente encapsulado. Ao se repassar o segmento para o próximo ponto de rede, este segmento será novamente encapsulado em um quadro da camada de enlace.

Quando o dado alcançar seu terminal de destino, então será desencapsulado dos formatos das camadas anteriores à de aplicação, o que inclui a remoção de seus respectivos cabeçalhos, fornecendo à camada de aplicação do terminal a mensagem inicialmente enviada.

2.1.5 A Internet

A Internet é a maior rede de computadores existente no mundo, e consiste na junção de várias redes menores conectadas umas às outras, possibilitando assim a comunicação entre equipamentos, independentemente de sua localização.

Para que uma conexão à Internet seja efetuada com sucesso, é necessário que três requisitos sejam preenchidos. São eles: conexão física, conexão lógica e aplicações.

A conexão física é implementada por um dispositivo conectado ao computador. Dispositivo este que pode ser [14]:

- Um Modem (Modulador-Demodulador);
- Uma placa de rede com ou sem fio (neste caso, também conhecida como placa *wireless*).

A conexão lógica entre equipamentos de rede utiliza rotinas conhecidas como protocolos, específicas para as camadas lógicas de rede, que se encarregam de estabelecer quais aplicações dos equipamentos ficarão encarregadas de replicar, em nível de *software*, as conexões físicas estabelecidas pelo *hardware* dos equipamentos conectados [14]. Mais adiante, nesta seção, o conceito de protocolos será discutido.

As aplicações consistem em programas de computador que utilizam as conexões lógicas estabelecidas via protocolos, para troca de dados. Dentre as principais aplicações utilizadas, podemos destacar os navegadores de Internet (*browsers*), utilizados para acessar páginas de Internet, e documentos diversos hospedados nestas páginas; e os aplicações do tipo clientes de correio eletrônico, utilizados para envio e recebimento de mensagens de texto, podendo ou não conter arquivos diversos anexados à mensagem [14][15].

Dentre aplicações do tipo navegadores, as mais comuns são: Microsoft Internet Explorer, Mozilla Firefox e Google Chrome. Por outro lado, em se tratando de aplicações do tipo clientes de correio eletrônico, destacam-se o Microsoft Outlook e o Mozilla Thunderbird.

Nesta seção, serão apresentados alguns protocolos relevantes para a compreensão desta dissertação.

2.2 Protocolos

2.2.1 O protocolo IP e endereçamento IP

2.2.1.1 O protocolo IP

Trata-se do protocolo responsável pela identificação das máquinas e das redes onde estas máquinas estão localizadas, responsável também pelo envio de mensagens de um equipamento a outro. É definido como um sistema de transmissão de pacotes sem conexão fixa preestabelecida entre as máquinas/redes origem e destino, portanto um protocolo não confiável, ou seja, não garante que os datagramas serão recebidos pelo destinatário; tampouco realiza medidas corretivas em caso de insucesso no envio. O protocolo IP também não garante que os datagramas serão todos transmitidos através de um único caminho de rede [17][15].

A versão correntemente utilizada do protocolo IP é a IPv4, especificada na RFC791 [18]. Com isso, sempre que for mencionado protocolo IP sem especificação de versão neste trabalho, deve-se entender como sendo a versão IPv4 do mesmo.

Os campos existentes em um datagrama IP são [18]:

- **Versão**: Utiliza 4 *bits*, e indica o formato do cabeçalho do datagrama, de acordo com a versão utilizada do protocolo;
- **Comprimento**: Também utiliza 4 *bits*, indicando o tamanho do cabeçalho do datagrama;
- **Tipo de serviço**: Utiliza 8 *bits*, e indica os parâmetros utilizados para o serviço desejado pelo datagrama (prioridade de envio na rede, tempo de espera para envio e tipo de taxa de transferência de dados);
- **Comprimento total**: Utiliza 16 *bits*, e indica o tamanho total do datagrama;

- **Identificação**: Utiliza 16 *bits*. É determinado pelo usuário para auxiliar na junção de datagramas que contenham fragmentos de uma dada informação, caso esta seja grande demais para ser enviada em um único datagrama;

- **Flags**: Utilizam 3 *bits*, e indicam se um determinado datagrama é ou não um fragmento de uma informação. Caso seja, estes campos indicam também se o fragmento é o último de sua sequência de envio;

- **Offset do fragmento**: Utiliza 13 *bits*, e indica qual parte da informação está contida no datagrama, através do número de sequência (*offset*) do fragmento;

- **Time to Live (TTL)**: Utiliza 8 *bits*, e indica a quantidade máxima de saltos que um datagrama pode executar ao longo da rede, até alcançar seu destino. A cada salto do datagrama de um ponto para outro, o valor do campo é decrementado em uma unidade. Caso o valor de TTL do datagrama seja zero, o mesmo deverá ser imediatamente descartado, de modo a impedí-lo de circular indefinidamente pela rede, e eventualmente causar congestionamentos à mesma;

- **Protocolo**: Utiliza 8 *bits*. Indica qual o protocolo a ser utilizado pelo datagrama na camada de nível superior à que roda o protocolo IP (Camada de Transporte) [18][19];

- **Checksum**: Utiliza 16 *bits*. Verifica se a integridade do datagrama não foi comprometida durante o processo de transmissão;

- **Endereço de origem**: Utiliza 32 *bits* para indicar o endereço do qual o datagrama está sendo enviado;

- **Endereço de destino**: Utiliza 32 *bits* para indicar o endereço para o qual o datagrama está sendo enviado;

- **Opções**: Tamanho variável. Utilizado para definir configurações diversas de envio do datagrama, tais como segurança, caminho a ser percorrido na rede, etc;

- **Padding**: Tamanho variável. Assegura que o cabecalho do datagrama possui exatamente 32 *bits*.

Na figura 2.5, é apresentada a estrutura de um datagrama IP.

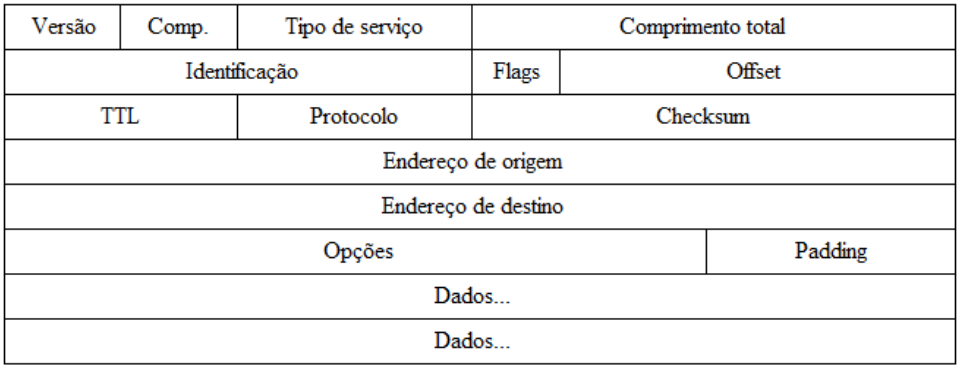


Figura 2.5: Estrutura de um datagrama IP

2.2.1.2 O endereçamento IP

O endereçamento IP utilizado atualmente em redes se encontra na versão quatro, conhecida como IPv4. Para que não ocorra nenhum problema na comunicação entre dois pontos de rede distintos, não pode haver, na rede em questão, dois endereços IP idênticos. Deve-se ter em mente que, neste caso, quando fala-se de rede, isto pode signifcar desde uma rede local, ou até mesmo a Internet. Em qualquer um destes casos, todos os equipamentos conectados à rede devem possuir sempre endereços IP únicos [15][14].

Um equipamento de uma rede poderá eventualmente possuir endereço IP igual a outro equipamento de uma outra rede, desde que estas redes não estejam conectadas entre si, direta ou indiretamente (no caso indireto, não haveria conexão entre as redes, contudo, seria possível, através de redes intermediárias, criar um caminho que as ligue).

Cada *host* (equipamento) possui seu endereçamento IP representado por um conjunto de quatro octetos (*bytes*, ou conjuntos de 8 *bits*), onde cada um pode assumir até 256 valores diferentes.

Para este conjunto de *bytes* é dado o nome de endereço IP do equipamento, responsável por situar o equipamento dentro da rede onde ele se encontra [15][14].

Para melhor aproveitamento dos endereços IP, em razão da grande quantidade de endereços existentes na Internet, foi criada uma divisão destes endereços em classes, divisão esta cujo propósito consiste em viabilizar a divisão da rede em sub-redes, facilitando a distribuição e administração destes endereços [14].

O critério utilizado para identificar a que classe de rede um determinado endereço IP pertence baseia-se no uso dos primeiros *bits* do primeiro *byte*. As redes são então classificadas em classes A, B, C, D e E, e a utilização dos *bits* para esta definição de classes pode ser observada a seguir:

- **Classe A**: Abrange todos os endereços IP cujo primeiro *bit* do primeiro *byte* seja igual a zero (0XXXXXXX), indo de 0.0.0.0 até 127.255.255.255;
- **Classe B**: Abrange todos os endereços IP cujo primeiro *bit* do primeiro *byte* seja igual a um; e o *bit* seguinte igual a zero (10XXXXXX), indo de 128.0.0.0 até 191.255.255.255;
- **Classe C**: Abrange todos os endereços IP cujos três primeiros *bits* do primeiro *byte* sejam iguais a 110 (110XXXXX), indo de 192.0.0.0 até 223.255.255.255;
- **Classe D**: Abrange todos os endereços IP cujos quatro primeiros *bits* do primeiro *byte* sejam iguais a 1110 (1110XXXX), indo de 224.0.0.0 até 239.255.255.255;
- **Classe E**: Abrange todos os endereços IP cujos quatro primeiros *bits* do primeiro *byte* sejam iguais a 1111 (1111XXXX), indo de 240.0.0.0 até 255.255.255.255.

Para facilitar a visualização do endereço IP, os valores armazenados em cada *byte* são apresentados em sua representação decimal, possibilitando assim os endereços IP variarem de 0.0.0.0 (00000000.00000000.00000000.00000000, em representação binária), até 255.255.255.255 (11111111.11111111.11111

111.11111111, em representação binária). Além disto, a representação de um endereço IP também mostra a máscara de rede utilizada pelo endereço, máscara esta que representa a rede onde este endereço está sendo utilizado.

Por exemplo, se um equipamento de rede possuir o endereço IP 192.168.16 .1/24, isto significa que 192.168.16.1 corresponde ao endereço individual do equipamento; enquanto que o número 24 indica a quantidade de *bits* utilizada no endereçamento para representar a rede onde o equipamento se encontra, contando a partir dos *bits* mais à esquerda no endereço (neste caso, os 24 primeiros *bits*). Com isto, chega-se à conclusão de que a rede possui endereço (máscara) 192.168.16.0/24; e que todos os equipamentos vinculados a esta possuirão endereços IP no formato 192.168.16.X, onde X pode variar entre 0 e 255, totalizando assim 256 equipamentos de rede como a capacidade máxima de equipamentos conectáveis a esta rede.

Caso o endereço IP do equipamento fosse 192.168.16.1/20, isto indicaria que apenas os 20 *bits* mais à esquerda no endereço são utilizados para definir a máscara de rede. Ou seja, a máscara de rede seria 192.168.16.0/20 (11000000.10101000.00010000.00000000); e que os equipamentos vinculados a esta possuirão endereços IP no formato 11000000.10101000.0001XXXX.XXX XXXXX, onde X pode variar entre 0 e 1, totalizando assim 4096 (2^{12}) equipamentos de rede como a capacidade máxima de equipamentos possíveis de serem conectados a esta rede [15].

Em uma dada sub-rede, normalmente o primeiro e o último endereços desta sub-rede não são utilizados para equipamentos de rede específicos. O primeiro, que normalmente é do tipo X.X.X.0, é chamado de endereço de rede, e possui a função de especificar a rede para envio/recebimento de dados de outras redes; já o segundo, normalmente do tipo X.X.X.255, é chamado endereço de *broadcast*, utilizado para repassar dados da rede para todos os equipamentos vinculados a ela.

2.2.1.3 O protocolo IPv6

No início dos anos 90, a organização conhecida como *Internet Engineering Task Force* - IETF, ao avaliar que os endereços IP criados no padrão IPv4

poderiam se esgotar em pouco tempo, iniciou um esforço no sentido de evitar o esgotamento dos endereços IP disponíveis para as redes de computadores. Deste esforço nasceu uma nova versão do protocolo IP, o IPv6 (chamado inicialmente como *Next Generation IP* - IPng [20]), cuja principal diferença, em relação ao protocolo IPv4, reside no total de *bits* disponíveis para endereçamento de equipamentos de rede. Enquanto o protocolo IPv4 disponibiliza um total de 32 *bits* para endereçamento, o protocolo IPv6 disponibiliza 128 *bits* para a mesma finalidade [15][21].

A especificação do protocolo IPv6 pode ser encontrada na RFC2460 [21]. Os campos existentes em um datagrama IPv6 são [15][21]:

- **Versão**: Utiliza 4 *bits*, e indica o formato do cabeçalho do datagrama, de acordo com a versão utilizada do protocolo;
- **Classe de tráfego**: Utiliza 8 *bits*, e funciona de forma similar ao campo Tipo de serviço, no protocolo IPv4;
- **Rótulo de fluxo**: Utiliza 20 *bits*, e identifica um dado fluxo de datagramas (uma transmissão de dados que demande maior prioridade, por exemplo);
- **Comprimento de carga útil**: Utiliza 16 *bits*, e indica o tamanho total, em *bytes*, do datagrama, que vem após o cabeçalho;
- **Próximo cabeçalho**: Utiliza 8 *bits*. Funciona da mesma forma que o campo Protocolo, do datagrama IPv4;
- **Limite de saltos**: Utiliza 8 *bits*, e funciona segundo o mesmo princípio de atuação que o campo *Time to Live*, do datagrama IPv4;
- **Endereço da fonte**: Utiliza 128 *bits* para indicar o endereço do qual o datagrama está sendo enviado;
- **Endereço do destino**: Utiliza 128 *bits* para indicar o endereço para o qual o datagrama está sendo enviado;
- **Dados**: Tamanho variável. Campo utilizado para armazenar os dados que devem ser enviados de um ponto a outro na rede.

Na figura 2.6, é apresentada a estrutura de um datagrama IPv6.

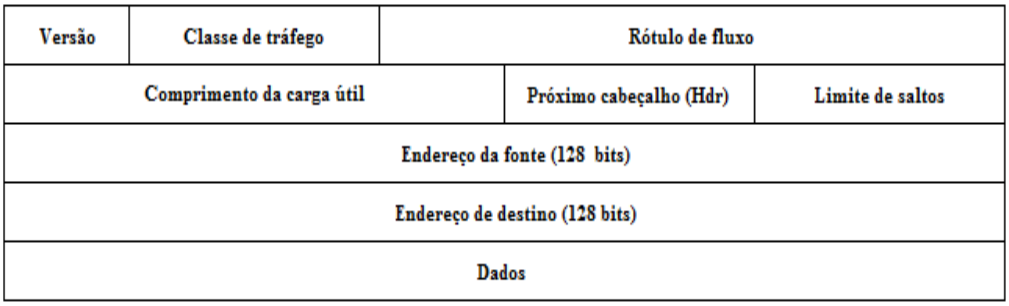


Figura 2.6: Estrutura de um datagrama IPv6

2.2.1.4 O protocolo ICMP

O protocolo *Internet Control Message Protocol* - ICMP faz parte do protocolo IP. Funciona de forma similar ao protocolo UDP, no entanto, não indica as portas de origem e destino do datagrama [17].

O protocolo ICMP é obrigatório em quaisquer implementações da camada de rede, uma vez que, normalmente indica ocorrência de problemas no transporte de algum datagrama, ou ainda é utilizado em operações de controle. Problemas de transporte podem ser ocasionados, por exemplo:

- Pelo fato da máquina destino não estar conectada à rede;
- Quando o campo TTL do datagrama IP fica com o valor zero;
- Quando os roteadores estão muito congestionados, o que provoca o descarte dos datagramas que não podem ficar armazenados em seus respectivos *buffer*s.

Nestes casos, o protocolo ICMP envia mensagens de erro ou controle para as máquinas que fizeram o envio de datagramas, sem no entanto, propor nenhuma solução para o erro encontrado, o que fica a cargo de quem enviou o datagrama [17].

Como o protocolo ICMP utiliza o protocolo IP para transportar as mensagens, não oferece garantia de entrega, nem conexão segura entre as máquinas

de origem e destino. A figura 2.7 apresenta o encapsulamento de uma mensagem ICMP dentro de um datagrama IP.

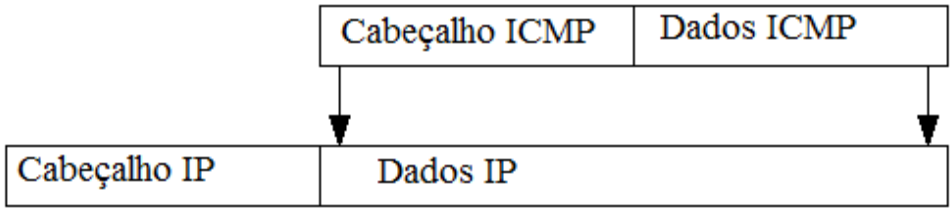


Figura 2.7: Encapsulamento de uma mensagem ICMP

Conforme pode ser visualizado na figura 2.7, ao ocorrer algum problema no envio do datagrama, dentro das situações previstas pelo protocolo ICMP, este envia uma mensagem de erro específica do que ocorreu para o protocolo IP, que então a adiciona ao cabeçalho do datagrama IP e o envia para a máquina que enviou a mensagem inicial [17].

A especificação do protocolo ICMP pode ser encontrada na RFC792 [22].

Os campos existentes em um datagrama ICMP são [17][22][23]:

- **Cabeçalho IP**: Utiliza 32 *bits*, e indica o cabeçalho do datagrama IP relacionado ao erro a ser enviado;
- **Tipo**: Utiliza 8 *bits*, e identifica o tipo de mensagem que o protocolo ICMP está enviando;
- **Código**: Utiliza 8 *bits*, e é utilizado na especificação do erro ocorrido no envio do datagrama original;
- ***Checksum***: Utiliza 16 *bits*. Verifica se a integridade do datagrama não foi comprometida durante o processo de transmissão;
- **Identificação**: Utiliza 16 *bits*. É utilizado para identificar as mensagens enviadas por um dado processo relacionado ao erro;
- **Número de sequência**: Utiliza 16 *bits*, e indica o número da mensagem enviada pelo protocolo ICMP.

Na figura 2.8 é apresentada a estrutura de um típico datagrama ICMP:

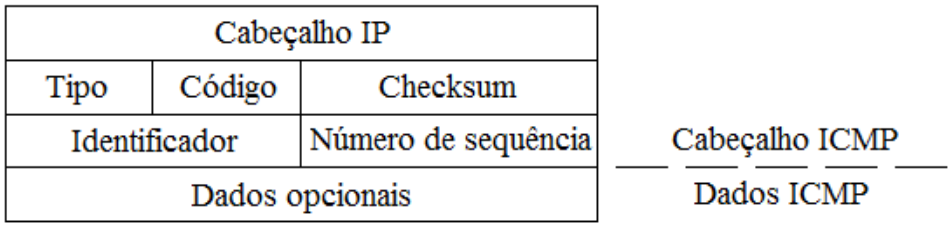


Figura 2.8: Estrutura de um datagrama ICMP

2.2.2 O protocolo TCP

O protocolo TCP se caracteriza pelo estabelecimento de uma conexão prévia entre dois equipamentos de rede, conexão esta operando na camada de transporte, conforme o modelo TCP/IP. Este tipo de conexão faz com que o protocolo TCP seja considerado como um protocolo orientado para conexão [15].

A partir do momento em que o protocolo TCP estabelece uma conexão entre dois computadores, esta conexão fornece um ambiente propício à transferência confiável de dados, ou seja, o protocolo TCP assegura que não haverá perda de dados no processo de transmissão. Isto é possibilitado pelo fato do protocolo TCP operar com os princípios de [15]:

- Detecção de erro, onde, se após um determinado período de tempo, o terminal remetente não receber nenhuma notificação de recebimento bem-sucedido de um dado segmento, entende-se isto como um erro no envio deste segmento;

- Retransmissão, onde, uma vez detectado um erro na transmissão, o terminal responsável pelo envio de dados reenvia o segmento TCP do qual foi apontado o erro de envio.

Outras características relevantes de uma conexão TCP são o serviço *full-duplex*, onde, uma vez estabelecida a conexão, o envio e o recebimento de dados podem ser feitos por ambos os terminais conectados, e de forma simultânea. Isto significa que, um terminal **A**, conectado a outro terminal **B**

via protocolo TCP pode, ao mesmo tempo enviar e receber dados do terminal **B**, independente de quem solicitou a conexão TCP.

A outra característica relevante citada no parágrafo anterior é o fato de que a conexão TCP é sempre do tipo ponto a ponto, ou seja, dois, e somente dois terminais podem ser conectados através de uma dada conexão TCP. Caso seja necessário fazer conexões TCP com outros terminais, estas conexões deverão ser estabelecidas especificamente para cada par de terminais a serem conectados entre si [15].

O princípio de conexão entre terminais no protocolo TCP se dá da seguinte forma [15]:

- Para que dois terminais se conectem, é necessário que cada um deles possua um processo rodando internamente. Um destes processos, chamado *processo cliente*, enviará uma mensagem à camada de transporte do terminal onde ele está operando (este terminal também será chamado *cliente*), mensagem esta solicitando uma conexão com o processo rodando no outro terminal (tanto o processo quanto o terminal que serão contactados serão chamados de *servidores*);

- A mensagem enviada para a camada de transporte no terminal *cliente* é então enviada para a camada de transporte do terminal *servidor*, por intermédio do endereço IP deste, e do número da porta, correspondente ao processo *servidor* com o qual a conexão será estabelecida. Este par de informações, juntamente com o protocolo de rede utilizado na conexão, é chamado de ***socket***;

- Uma vez que o *servidor* receba com sucesso a mensagem inicial, este envia outra mensagem para a camada de transporte do terminal *cliente*, para comunicar o estabelecimento bem-sucedido da conexão;

- O terminal *cliente* responde então com outra mensagem (também chamada de segmento TCP), começando a partir deste ponto o envio da chamada "carga útil", carga esta onde estarão contidos os dados propriamente ditos, seja para solicitar o envio de dados (terminal *cliente*); seja para fazer o envio dos mesmos (terminal *servidor*). Em função da

necessidade de se enviar pelo menos três segmentos TCP para que haja efetivamente transferência de dados, esta forma de conexão é conhecida como **apresentação de três vias**.

A partir do momento em que a conexão TCP está estabelecida, os terminais *cliente* e *servidor* podem enviar livremente segmentos TCP um para o outro, sem necessidade de estabelecer nova conexão TCP, pelo tempo em que esta conexão se mantiver em funcionamento. Este envio é realizado da seguinte forma [15]:

- Suponhamos que o terminal *cliente* necessite enviar dados para o terminal *servidor*. O processo TCP relativo ao *cliente* então envia estes dados para a porta por ele utilizada na conexão TCP com o terminal *servidor*;

- Esta porta, por sua vez, repassa tais dados para o *buffer* de envio, que consiste na fila de onde os dados serão enviados para seu destino. Este *buffer* é criado no momento em que a conexão TCP é estabelecida;

- Dentro do *buffer*, o protocolo TCP divide os dados, conforme a largura de banda disponível na conexão, anexando um cabeçalho TCP a cada pedaço de dados a ser enviado. A cada conjunto cabeçalho + dados criado, dá-se o nome de segmento TCP;

- Cada um destes segmentos TCP são então repassados para a camada de rede, onde serão encapsulados dentro de datagramas IP e enviados, através da rede, para o *buffer* de recepção do terminal *servidor*. Ao repassar os dados para o *buffer*, o protocolo TCP desencapsulará os segmentos TCP de dentro dos datagramas IP e repassará os dados contidos nos segmentos para o *buffer*;

- O *buffer* de recepção repassa os dados recebidos para a porta do terminal *servidor*; e esta envia os dados para o processo TCP do mesmo terminal, que os lê e repassa os conteúdos destes para o *servidor* propriamente dito.

A figura 2.9 ilustra o envio de segmentos TCP através da rede, conforme o texto acima.

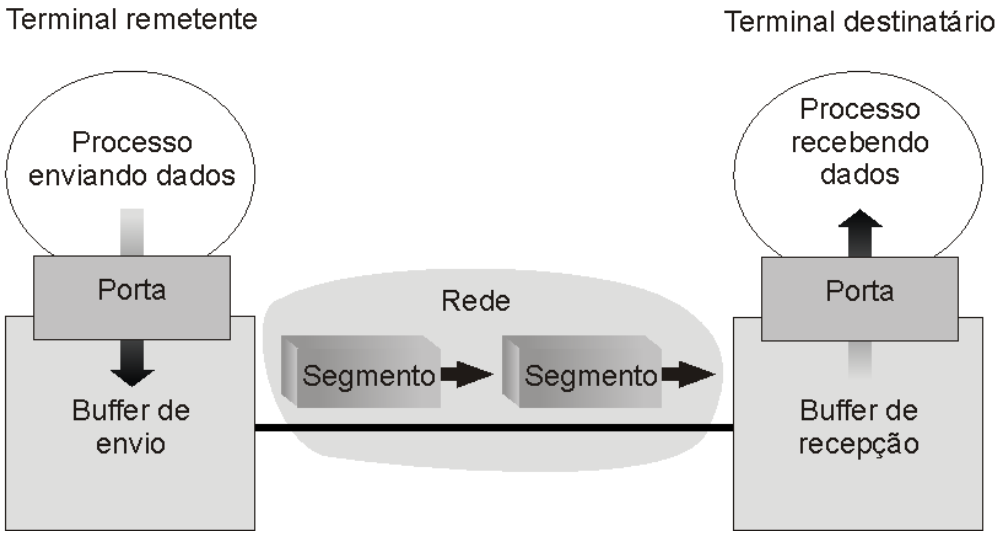


Figura 2.9: Envio de segmentos TCP

Para que os dados enviados via protocolo TCP possam ser recebidos na ordem correta por seus destinatários, são utilizados nos segmentos TCP os campos de número de sequência e número de reconhecimento. A partir do momento em que os dados a serem enviados começam a ser estruturados em segmentos, o protocolo atribui, para o primeiro *byte* do segmento, um número sequencial, levando em conta a quantidade de *bytes* de dados suportados para os segmentos TCP criados pelo protocolo. A este número, se convencionou a denominação de número de sequência. Por exemplo, se cada segmento TCP comportar 1000 *bytes* de dados, os números de sequência para cada segmento TCP serão: 0 (primeiro *byte* do primeiro segmento), 1000 (primeiro *byte* do segundo segmento), e assim por diante.

A figura 2.10 apresenta a estruturação de *bytes* em segmentos TCP, juntamente com a atribuição de números de sequência para cada segmento TCP criado.

O número de reconhecimento, por outro lado, corresponde ao número de sequência que o terminal que está recebendo dados espera receber do outro terminal com quem está mantendo a conexão TCP corrente. Este número é

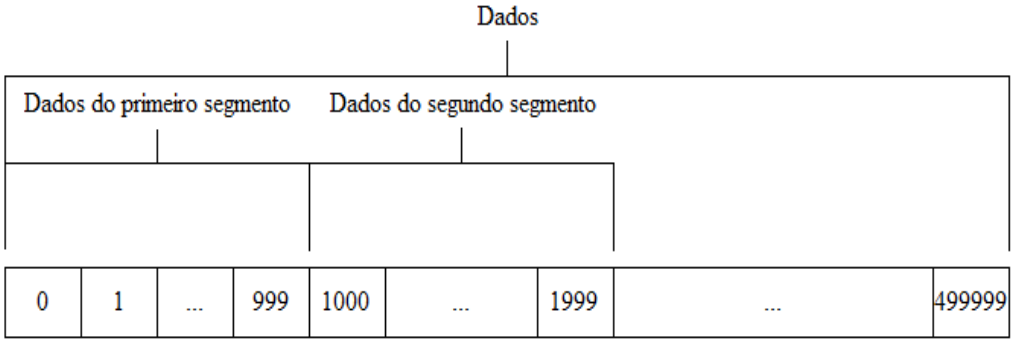


Figura 2.10: Estruturação de *bytes* em segmentos TCP

necessário em razão da natureza *full-duplex* do protocolo TCP, onde ambos os terminais conectados podem, ao mesmo tempo, enviar e receber segmentos.

Por exemplo, se um terminal **A** estiver recebendo segmentos TCP, com 1000 *bytes* de dados cada, de outro terminal **B**, e até o presente momento, **B** tenha enviado com sucesso os primeiros 25 segmentos TCP para **A**, isto significa que, até que **A** consiga receber com sucesso o 26º segmento vindo de **B**, o número de reconhecimento de todos os segmentos TCP orinudos de **A** para **B** será 2600 (primeiro *byte* do 26º segmento).

Em função da regra do protocolo TCP para envio e recebimento ordenados de segmentos, existem duas formas em o mesmo pode proceder, quando do recebimento desordenado de segmentos (o que pode ser causado, por exemplo, por erros de envio) [15]:

- Descarte imediato de quaisquer segmentos recebidos fora de ordem, e consequentemente o recebimento posterior destes, quando de forma ordenada;

- Conservação, por parte do terminal destinatário, dos segmentos recebidos desordenadamente, até que os segmentos anteriores que estejam faltando sejam recebidos, de modo a completar os dados enviados. Esta é a solução utilizada na prática, por causar menor tráfego de dados na rede.

A especificação do protocolo TCP pode ser encontrada na RFC793 [24].

Os campos existentes em um segmento TCP são [15][24]:

- **Porta de origem**: Utiliza 16 *bits*, e indica o número da porta de onde o segmento é enviado;
- **Porta de destino**: Também utiliza 16 *bits*, e indica o número da porta para onde o segmento será enviado;
- **Número de sequência**: Utiliza 32 *bits*, e indica o nḿero do primeiro *byte* de um segmento TCP;
- **Número de reconhecimento**: Utiliza 32 *bits*, e indica o nḿero do próximo *byte* esperado pelo destinatário do segmento TCP;
- **Comprimento do cabeçalho**: Utiliza 4 *bits*, e indica o tamanho do cabeçalho do segmento TCP, que não possui tamanho fixo, por causa do campo de Opções, cujo tamanho pode variar, dependendo da natureza do segmento TCP;
- **Bits não utilizados**: Conjunto de 6 *bits* reservados para uso futuro, a ser definido;
- **Flag**: Utiliza 6 *bits*, e funciona como dispositivo de controle de funcionalidades do protocolo TCP, que serão utilizadas, caso os *bits* estejam com valor "1". São elas:
    1. **URG**: Indica a existência, se houver, de dados marcados como "urgentes" no segmento TCP;
    2. **ACK**: Indica se o valor carregado no campo Número de reconhecimento é válido, caracterizando assim o envio bem sucedido de segmentos anteriores;
    3. **PSH**: Indica a necessidade da imediata transmissão de dados para a camada de aplicação;
    4. **RST**: Indica a reinicialização da conexão;
    5. **SYN**: Indica a sincronização (atualização) do campo Número de sequência do segmento;

6. **FIN**: Indica que o segmento atual é o último segmento da cadeia de dados corrente.

- **Janela de recepção**: Utiliza 16 *bits*, e indica o total de *bytes* que o terminal destinatário aceitará;
- **Valor de verificação da Internet**: Utiliza 16 *bits*, e verifica se a integridade do segmento não foi comprometida durante o processo de transmissão;
- **Ponteiro para dados urgentes**: Utiliza 16 *bits* e indica, caso o *bit* do campo URG esteja sinalizado com valor "1", a localização do último *bit* dos dados considerados "urgentes" no segmento corrente;
- **Opções**: Tamanho variável, utilizado para configurar opções extras do segmento TCP, como, por exemplo, o tamanho máximo que um segmento TCP a ser transmitido poderá ter;
- **Dados**: Tamanho variável, contendo a informação a ser enviada do terminal remetente para o terminal destinatário.

A figura 2.11 apresenta a estrutura de um segmento TCP.

<table>
<tr><td colspan="8">Porta de origem</td><td colspan="8">Porta de destino</td></tr>
<tr><td colspan="16">Número de sequência</td></tr>
<tr><td colspan="16">Número de reconhecimento</td></tr>
<tr><td colspan="4">Comp. Cabeçalho</td><td colspan="2">-</td><td>URG</td><td>ACK</td><td>PSH</td><td>RST</td><td>SYN</td><td>FIN</td><td colspan="8">Janela de recepção</td></tr>
<tr><td colspan="10">Valor de verificação da Internet</td><td colspan="6">Ponteiro para dados urgentes</td></tr>
<tr><td colspan="16">Opções</td></tr>
<tr><td colspan="16">Dados</td></tr>
</table>

Figura 2.11: Estrutura de um segmento TCP

2.2.3 O protocolo UDP

O protocolo UDP fornece uma forma mais simples de acesso ao sistema de comunicação, se comparado ao protocolo TCP. Entretanto, ao contrário

do TCP, que é baseado no estabelecimento de conexão prévia e segura para transmissão de dados, e que garante a correção de segmentos não enviados corretamente, o protocolo UDP provê um serviço sem conexão prévia entre a máquina que envia os dados e a que deve recebê-los, sem confiabilidade e sem correção de erros [17][15].

O protocolo UDP funciona como multiplexador/demultiplexador para o envio e recepção de segmentos, utilizando as portas disponíveis em cada nó de rede para direcioná-los.

Para tratar o recebimento de mensagens, o protocolo UDP recebe os datagramas IP, desencapsula os segmentos e os demultiplexa com base na porta de destino UDP, conforme apresentado na figura 2.12. Assim, o protocolo UDP verifica se a porta de destino coincide com alguma porta existente no nó de rede para onde o segmento será enviado. Em caso negativo, uma notificação ICMP de porta não acessível é transmitida para o emissor do segmento e o mesmo é descartado. Do contrário, se a porta destino estiver ocupada no momento da requisição, o segmento é colocado no *buffer* da mesma. Se o *buffer* estiver cheio, então o segmento é descartado; caso contrário, o protocolo UDP, assim que possível, multiplexa o segmento e o envia para a porta destino [17].

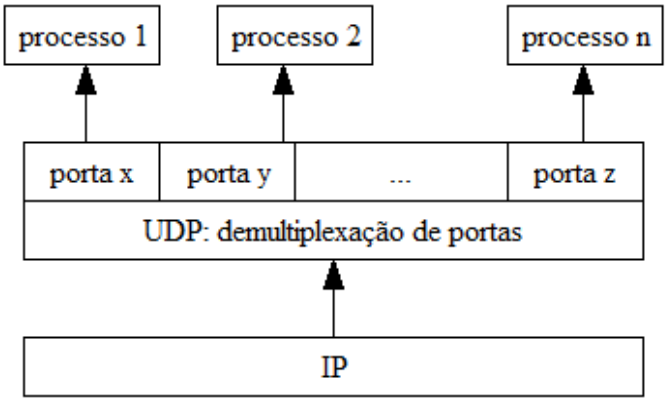


Figura 2.12: Funcionamento do protocolo UDP

Em tese, os processos de multiplexação/demultiplexação realizados pelo

protocolo UDP ocorrem através do uso de portas. Mas, na prática, cada aplicação, antes de enviar dados para o protocolo UDP, precisa negociar com o sistema operacional para obter uma porta. Uma vez satisfeito este requisito, os segmentos UDP enviados pela porta obtida terão o número dela em seu campo porta de origem [17].

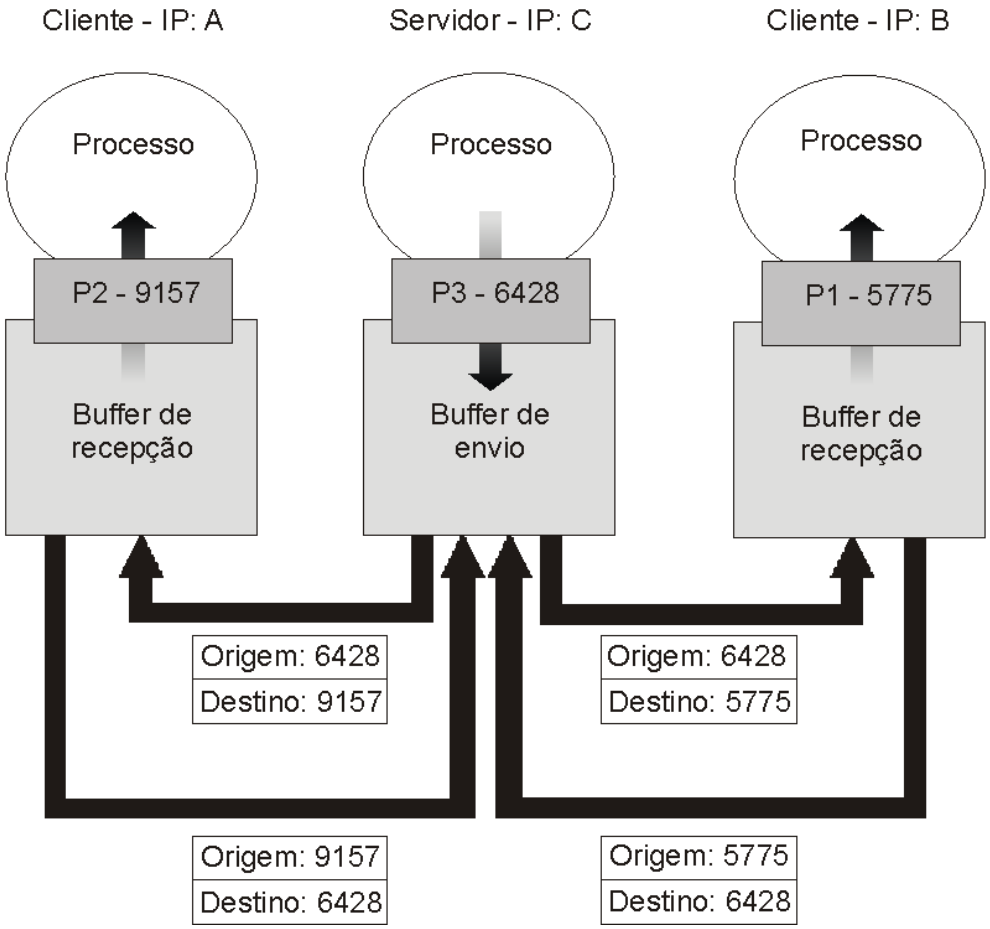


Figura 2.13: Exemplo de envio de segmentos UDP através de portas

No exemplo apresentado pela figura 2.13, podemos observar o funcionamento do segmento UDP, conforme descrito no parágrafo anterior. Uma vez que o *cliente* **A** tenha negociado com o sistema operacional do *servidor* **C** a utilização da porta **P3** de **C**, cujo número é 6428, em conjunto com a porta **P2** de **A**, de número 9157, o protocolo UDP realizará o envio de segmentos entre **A** e **C** através destes números de porta, que serão utilizados nos campos

porta origem e porta destino dos segmentos UDP, dependendo do sentido de envio de cada segmento. O mesmo raciocínio se aplica à relação entre o *cliente* **B** e o *servidor* **C** [17][15][25].

A especificação do protocolo UDP pode ser encontrada na RFC768 [26]. Os campos existentes em um segmento UDP são [26][25]:

- **Porta de origem**: Utiliza 16 *bits*, e indica o número da porta de onde o segmento UDP é enviado;
- **Porta de destino**: Também utiliza 16 *bits*, e indica o número da porta para onde o segmento UDP será enviado;
- **Tamanho**: Utiliza 16 *bits*, e indica o tamanho total do segmento UDP;
- **Checagem (*checksum*)**: Utiliza 16 *bits*. Verifica se a integridade do segmento UDP não foi comprometida durante o processo de transmissão.

Na figura 2.14 é apresentada a estrutura de um segmento UDP:

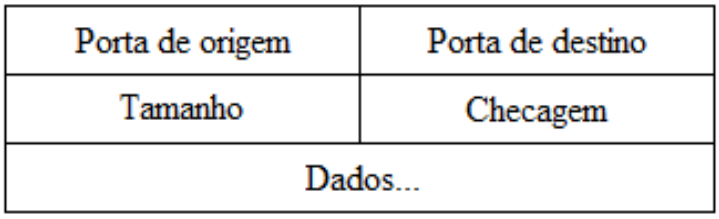


Figura 2.14: Estrutura de um segmento UDP

Como o protocolo UDP opera na camada de transporte, é preciso fazer com que os dados do segmento UDP sejam inseridos dentro de um ou mais datagramas IP, para que possam operar na camada de rede.

Assim, cada segmento UDP é encapsulado dentro de um datagrama IP, incluindo-se nisto cabeçalho e dados, conforme a figura 2.15 [17][15]:

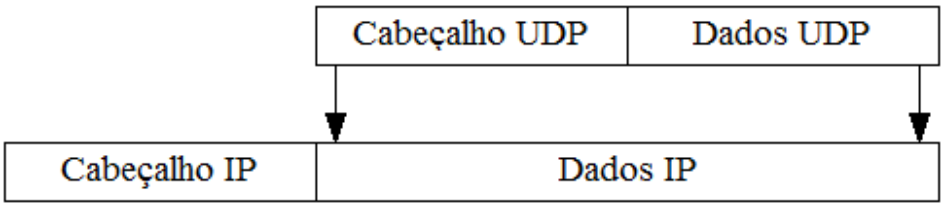


Figura 2.15: Encapsulamento de um segmento UDP em um datagrama IP

2.2.4 O protocolo Telnet

O protocolo Telnet opera na camada de aplicação de rede. Este nome também é utilizado para se referir ao aplicativo desenvolvido com base neste protocolo. O aplicativo Telnet tem como propósito o acesso remoto, a partir do *host* de onde está sendo executado, a outros *hosts*, normalmente utilizando conexões TCP na porta 23 do *host* acessado [27].

O protocolo Telnet foi criado em 1974, e disponibilizado pela empresa *Bolt, Beranek, and Newman* - BB&N. Seu propósito consiste em fornecer uma conexão bidirecional de 8 *bit*s de largura de banda entre dois *hosts*, juntamente com a interface do *host* acessado, para manipulação de dados no mesmo [27].

O protocolo Telnet baseia seu funcionamento em três conceitos. São eles [27]:

- ***Network Virtual Terminal* - NVT**: No momento em que o protocolo Telnet estabelece uma conexão com um dado *host*, gera-se uma abstração conhecida como NVT. Esta abstração é um equipamento virtual que fornece uma interface padrão de conexão via terminal Unix com o *host* acessado;

- **Princípio de opções negociadas**: Este princípio fornece serviços adicionais ao protocolo Telnet, ou desabilita serviços em funcionamento. Para isto, é preciso enviar uma solicitação de ativação/desativação de dado serviço ao *host* remoto, que enviará uma resposta positiva ou negativa, em caso de pedido de ativação; ou positiva, se o pedido for

para desativar dado serviço;

- **Regras de negociação**: Impedem a ocorrência de *loops*, onde não é possível concluir um dado pedido de conexão.

Como o aplicativo Telnet utiliza normalmente conexões TCP, assegura-se assim uma conexão segura para envio/recebimento de dados entre os *hosts* conectados.

O aplicativo Telnet pode ser usado não apenas para acessar outros computadores, mas também para acessar equipamentos de rede, como roteadores *switches*, etc. Uma vez que tal conexão esteja estabelecida, os comandos a serem utilizados no *host* acessado irão variar de acordo com a natureza deste.

O comando a ser utilizado pelo aplicativo Telnet varia de acordo com o sistema operacional de onde o mesmo estiver sendo executado. Contudo, o comando mais comum para este aplicativo obedece a sintaxe a seguir [28]:

telnet endereco_a_ser_acessado numero_da_porta

Onde, além do aplicativo propriamente dito, serão passados como parâmetros o *host* que se deseja acessar, e o número da porta pela qual o acesso deverá ser feito. Caso nenhum número de porta seja especificado, o aplicativo assumirá que o número da porta é 23.

Como exemplo da utilização do Telnet, suponha-se que se deseja fazer um acesso remoto ao *host* wubba.cs.widener.edu, pela porta padrão (23). O aplicativo seria então utilizado da seguinte forma:

telnet wubba.cs.widener.edu

E apresentaria o resultado abaixo:

Trying 147.31.254.999...
Connected to wubba.cs.widener.edu.
Escape character is '^]'.

O caractere de escape representa o que o usuário deverá digitar para fechar a conexão aberta, retornando assim para a linha de comando de sua própria máquina local.

As especificações básicas do protocolo Telnet estão disponíveis na RFC854, enquanto as ferramentas adicionais deste são descritas nos RFC's855 a 861 [27][29][30][31][32][33][34][35].

2.2.5 O protocolo SNMP

No início da década de 80, iniciou-se o desenvolvimento do protocolo de gerência de redes *Simple Network Management Protocol* - SNMP, por parte da organização *Internet Engineering Task Force* - IETF, para disponibilizar uma forma simples de gerenciar equipamentos de uma dada rede de computadores [36]. Este protocolo começou a ser empregado efetivamente na década de 90, com o advento da RFC1157 [37].

Antes do protocolo SNMP, o protocolo utilizado para a gerência de rede era o *Simple Gateway Management Protocol* - SGMP. A principal limitação do SGMP residia no fato dele ter sido desenvolvido apenas para a gerência de roteadores, restringindo sua utilização em gerência de redes. Por sua vez, o protocolo SNMP é capaz de gerenciar sistemas operacionais, periféricos, etc [38]. A especificação do protocolo SGMP se encontra na RFC1028, publicada em 1987 [39].

Atualmente, existem 3 versões do protocolo SNMP disponíveis para utilização. A versão SNMPv1 é a versão corrente, como padrão completo da organização IETF. Sua segurança está pautada no conceito de comunidades (*communities*). *Communities* funcionam de forma similar a senhas, sendo dados do tipo *string* utilizados para:

- Estabelecer conexões seguras entre equipamentos monitorados e as máquinas que os gerenciam [38];

- Acessar estatísticas de um dado equipamento de rede, desde que a *community* em questão tenha permissão para acessar tais estatísticas [37].

Conforme exposto anteriormente, esta versão foi disponibilizada para uso em 1990, quando da publicação da RFC1157 [37].

A versão SNMPv2, também conhecida em algumas literaturas como SNMPv2c, também possui status IETF, contudo, em nível Experimental[1][40], embora alguns fornecedores já a utilizem na prática [38]. Especificações desta versão podem ser encontradas nas RFC's1902 [41], 1903 [42], 1904 [43], 1905 [44], 1906 [45], 1907 [46], 1908 [47] e 1909 [48], todas publicadas em 1996. As principais mudanças que esta versão traz em relação à versão SNMPv1 são:

- Correção de falhas de segurança na autenticação das *communities*, quando algum aplicativo do protocolo SNMP tenta estabelecer conexão, ou acessar uma estatística qualquer de um equipamento monitorado. Desta mudança vem a razão para o nome SNMPv2c, onde o protocolo tem sua segurança mais fortemente baseada no conceito de *communities*, justificando o "c" ao final do nome desta versão [38][48];

- Criação do tipo de dado Counter64, como versão estendida do tipo Counter, criado na versão SNMPv1. Este tópico será abordado mais adiante [41];

- Criação do tipo de mensagem InformRequest, onde dois ou mais gerentes podem se comunicar, possibilitando assim um gerenciamento descentralizado de uma rede [44];

- Criação do tipo de mensagem GetBulkRequest, para otimizar a recuperação de dados de equipamentos monitorados [44].

Já a versão SNMPv3, que ainda não possui status IETF, apresenta, em relação às suas predecessoras, maior ênfase no aspecto da segurança fornecida às atividades do protocolo [40]. Especificações desta versão podem ser encontradas nas RFC's2570 [49], 2571 [50], 2572 [51], 2573 [52], 2574 [53] e 2575 [54], todas publicadas em 1999. Dentre outros aspectos, pode-se destacar [38][49]:

[1] O status IETF Experimental é concedido às RFC's que fazem parte de alguma pesquisa em determinado assunto, mas ainda sem serem largamente utilizadas pelos fabricantes de equipamentos de redes. As RFC's largamente utilizadas por diversos fabricantes recebem status Informacional; e as RFC's consideradas obsoletas recebem status Histórico.

- Suporte para autenticação rigorosa junto a cada equipamento monitorado pelo protocolo;
- Comunicação privativa entre os equipamentos monitorados.

Atualmente, o principal objetivo das pesquisas em gerenciamento de redes consiste na obtenção da máximo desempenho no funcionamento da rede, otimizando assim o uso de seus elementos de gerência e monitoramento. Conforme explicado nos parágrafos anteriores sobre as versões do protocolo SNMP, este também tem sido o principal enfoque de melhorias e correções deste protocolo [36].

2.2.5.1 MIB

Antes de se falar sobre o protocolo SNMP propriamente dito, é preciso entender a base de dados acessada por este protocolo, para obter todas as estatísticas de interesse dos equipamentos de rede monitorados. Esta base de dados em questão é a MIB.

A *Management Information Base* - MIB é o conjunto de objetos de um dado equipamento de rede, organizados de modo a contemplar todas as estatísticas deste equipamento para seu respectivo gerenciamento. Pode ser considerada ainda como um banco de dados, responsável por armazenar todas estas estatísticas [36][38].

Objetos são a abstração dos recursos de um sistema. Todos os recursos a serem gerenciados via SNMP são modelados, e as estruturas de dados resultantes desta modelagem são então denominadas objetos, que contém as estatísticas de um equipamento de rede. Os objetos podem ser configurados para leitura/escrita, ou somente leitura, dependendo do nível de permissão concedido ao gerente. Cada leitura representa o estado atual do dispositivo, enquanto que as escritas modificarão tal estado, refletidas em tempo real para o(s) gerente(s) que estiver(em) visualizando o objeto naquele momento.

Atualmente, são definidos três tipos de MIB: MIB II, MIB experimental e MIB privada. A MIB II foi originalmente especificada na RFC1213 [55], como uma evolução da MIB I, apresentada originalmente na RFC1066 [56]. Esta evolução consistiu no uso da MIB baseado na pilha de protocolos

TCP/IP, explicando-se, na MIB I, o conjunto de informações necessárias para a implementação de práticas de monitoramento e controle de redes baseadas no conjunto de protocolos TCP/IP [36].

Além disso, a MIB II também fornece informações de gerenciamento sobre um dado equipamento. Através disto, podemos obter estatísticas como, por exemplo, número de pacotes, ou *bytes* (também chamados de octetos) transmitidos por cada uma das interfaces do equipamento, totais de envios bem-sucedidos e com falha de pacotes, estado da(s) interface(s), etc.

Na MIB experimental, os objetos que a integram estão em fase de desenvolvimento e/ou teste, possuindo normalmente características mais específicas sobre a tecnologia dos meios de transmissão e equipamentos empregados [36].

Por fim, a MIB privada é a modalidade onde os componentes fornecem estatísticas específicas dos equipamentos gerenciados, tais como configuração e colisões; e também é possível reinicializar e/ou desabilitar uma ou mais portas de um roteador gerenciado pela MIB em questão [36].

Como as MIB's experiental e privada são MIB's I ou II customizadas para fins específicos dos fabricantes que as utilizam, não existem RFC's publicadas para estas bases.

- Construção

  As regras de construção da MIB são descritas na *Structure of Management Information* - SMI. A SMI é um conjunto de documentos que definem [36][38]:

  - Forma de identificação e agrupamento das estatísticas;
  - Sintaxes permitidas para elas;
  - Tipos de dados permitidos;
  - Lista de equipamentos gerenciados e os objetos que integram cada um deles;
  - Objetos gerenciados e seus respectivos comportamentos.

Desta forma, cabe ao conjunto de documentos SMI construir a estrutura de gerenciamento de objetos e equipamentos, enquanto que cabe à MIB apresentar esta estrutura construída [38].

Os objetos constituintes de uma MIB são especificados pelo documento *Abstract Syntax Notation One* - ASN.1, encarregado de fornecer uma descrição abstrata destes objetos, descrição esta que desconsidera a estrutura e as restrições do equipamento no qual a MIB é implementada. Para cada objeto são definidas as seguintes instâncias:

- **Object Name**: Nome do objeto, composto por uma string de texto curto;
- ***Object Identifier* - OID**: Identificador do objeto, formado por números que são separados por pontos. *Exemplo: .1.3.6.1.2.1.2.2.1.10*;
- **Syntax**: sintaxe do objeto, descreve o formato, ou o valor, da informação. Pode ser de um tipo simples (inteiro, *string* de octetos (*bytes*), OID ou nulo); ou uma sintaxe de aplicação (endereço IP, contador, medida específica, intervalo de tempo ou incompreensível ao usuário);
- **Definição**: Descrição textual do que é o objeto em questão;
- **Acesso**: Tipo de permissão que pode ser definida, em relação à manipulação de conteúdo do objeto, podendo ser: somente leitura, leitura e escrita ou não acessível.

- Estrutura

A figura 2.16 apresenta a estrutura definida para representar a estrutura lógica da MIB, com seus identificadores e nomes de cada componente.

O nó raiz não possui nenhuma denominação, mas possui pelo menos três subníveis:

- **Nó .0**: Administrado originalmente pelo *Comité Consultatif International Téléphonique et Télégraphique* - CCITT, rebatizado

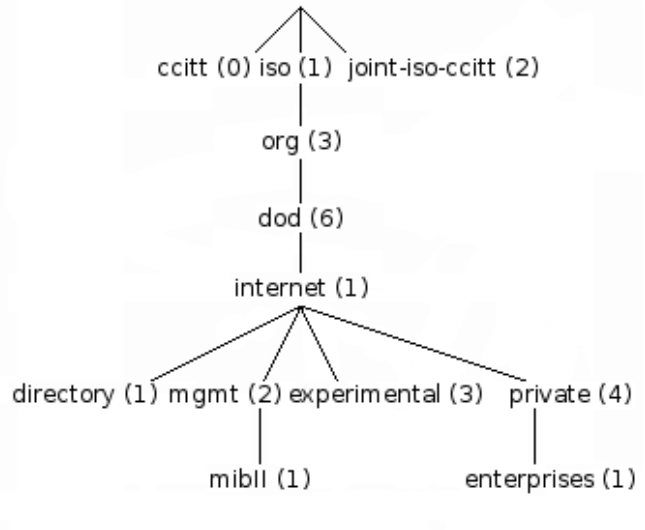


Figura 2.16: Estrutura lógica de uma MIB

posteriormente como *Union Internationale des Télécommunications* - UIT;

- **Nó .1**: Administrado pelo *International Organization for Standartization* - ISO;
- **Nó .2**: Administrado em conjunto pelas organizações UIT (antigo CCITT) e ISO.

Sob o nó .1 fica o nó .1.3, chamado *org*, que pode ser utilizado por outras instituições. Abaixo de *org* fica o *dod* (.1.3.6), pertencente ao departamento de defesa dos EUA. O departamento de defesa dos EUA, por sua vez, alocou um nó para a comunidade *internet* (.1.3.6.1), que é administrado pela organização *International Activities Board* - IAB, e abaixo deste estão os nós *directory* (.1.3.6.1.1), *management* (.1.3.6.1.2), *experimental* (.1.3.6.1.3) e *private* (.1.3.6.1.4).

Sob *directory* estão as informações sobre o serviço de diretórios OSI [57].

Sob *management* estão as informações de gerenciamento, e sob este está o nó responsável pela MIB II (.1.3.6.1.2.1).

Sob *experimental* estão os nós das MIBs experimentais.

Sob *private* fica o nó *enterprises* (.1.3.6.1.4.1) e sob este ficam os nós das indústrias de equipamentos gerenciados na MIB, como o nó da IBM (.1.3.6.1.4.1.2) e o da Cisco (.1.3.6.1.4.1.9), dentre outros [36][58].

- MIB II

  1. **Organização**

     Abaixo da sub-árvore *MIB II* estão os objetos usados para obter informações específicas dos dispositivos da rede. Os principais objetos estão divididos em 10 grupos, presentes na tabela 3.1;

  2. **Tipos de dados**

     Na sub-árvore *MIB II*, cada objeto contém um determinado tipo de dado, especificado antes do valor nominal contido no objeto propriamente dito, quando da verificação do nó em questão. Os tipos de dados considerados pelo documento SMI são [38]:

     - **Integer**: Número inteiro, normalmente utilizando 32 *bits*. De acordo com a RFC1155 [59], o valor zero (0) não deve ser utilizado para este tipo de dado; caso contrário o objeto em questão não será listado na MIB. Na versão SMIv2, também existe o dado do tipo Integer32, equivalente ao Integer convencional [38];
     - **String**: É uma string contendo zero ou mais octetos (*bytes*). Utilizada para especificar textos, ou ainda endereços físicos de um ou mais objetos;
     - **Counter**: Número de 32 *bits*, sempre definido como zero, a cada vez em que ocorre *overflow* no contador, ou seja, o valor máximo de $2^{32}-1$ é excedido. Nunca sofre decréscimo, exceto para os casos de *overflow*, ou reinicialização do equipamento; sendo utilizado para contabilizar quaisquer informações de totais de um equipamento, como, por exemplo, totais de pacotes recebidos e enviados. Na versão SMIv2, também existe o dado do tipo Counter32, equivalente ao Counter convencional;

Tabela 2.1: Grupos de informações da sub-árvore *MIB II*

Grupo	Informação
system (1)	informações básicas do sistema
interfaces (2)	interfaces de rede
at (3)	tradução de endereços
ip (4)	protocolo IP
icmp (5)	protocolo ICMP
tcp (6)	protocolo TCP
udp (7)	protocolo UDP
egp (8)	protocolo EGP
transmission (10)	meios de transmissão
snmp (11)	protocolo SNMP

- **Object Identifier**: String composta por números e pontos, utilizada como representação do caminho para chegar até um dado objeto da MIB;
- **Null**: Representa um objeto que não esteja sendo utilizado pelo protocolo SNMP;
- **Sequence**: Define listas de dados diferentes existentes no documento ASN.1. Pode conter zero ou mais tipos;
- **Sequence Of**: Define objetos gerenciados formados por dados do tipo Sequence;
- **IpAddress**: Utilizado para representar endereços de rede (IP's), conforme o padrão IPv4 (32 *bits*). As versões existentes do documento SMI (SMIv1 e SMIv2) não processam endereços IP segundo o futuro padrão de endereços IPv6 (128 *bits*), problema este a ser resolvido pela próxima versão do documento SMI (*SMI Next Generation* - SMING);
- **NetworkAddress**: Idem ao IpAddress, mas sendo capaz de representar tipos de endereços de rede diferentes do IPv4;
- **Gauge**: Mesmo tipo de dado que o tipo Counter, com a diferença de poder aumentar e diminuir seu valor, independente da ocorrência ou não de *overflow* ou reinicialização do equipamento. Na versão SMIv2 do documento SMI, também

existe o dado do tipo Gauge32, equivalente ao Gauge convencional;

- **TimeTicks**: Dado numérico de tamanho igual aos dados Counter e Gauge, normalmente utilizado para medir tempo corrente a partir de algum evento específico do equipamento, como, por exemplo, sua última reinicialização;
- **Opaque**: Habilita o armazenamento de codificações do documento ASN.1 em um objeto do tipo String;
- **Unsigned32 - exclusivo da versão SMIv2**: Utilizado para representar valores numéricos de 0 até $2^{32} - 1$;
- **Counter64 - exclusivo da versão SMIv2**: Apresenta o mesmo princípio de funcionamento que o Counter32, mas com 64 *bits* de tamanho, variando assim de 0 até $2^{64} - 1$. Utilizado quando números de 32 *bits* se mostram ineficazes como contadores, ou seja, o estado de *overflow* é alcançado muito rapidamente pela estatística em questão;
- **Bits - exclusivo da versão SMIv2**: Listagem de *bits* não negativos do objeto gerenciado.

3. **Exemplos**

   No apêndice B, estão listados alguns exemplos de nós existentes na sub-árvore MIB II.

2.2.5.2 O Protocolo SNMP

O protocolo SNMP é utilizado para monitoramento de desempenho e funcionamento de equipamentos de rede, tais como roteadores, *switches*, *servidores*, etc [60].

Ao se desenvolver este protocolo, os principais objetivos eram flexibilidade de uso do SNMP e facilidade de sua implementação. Sua especificação pode ser encontrada na RFC1157, publicada em 1990 [37].

O funcionamento do protocolo SNMP se baseia em dois dispositivos: o agente e o gerente.

- **Agente**: Executado na máquina gerenciada pelo protocolo SNMP, guardando as estatísticas de gerência da máquina, estatísticas estas estruturadas em uma base de dados MIB. Suas funções principais são:
  - Atender às requisições enviadas pelo gerente (*servidor*);
  - Enviar informações de gerenciamento ao gerente (*servidor*), quando programado para tal.
- **Gerente**: Executado a partir de uma máquina atuando como um *servidor*, permitindo obtenção e envio de estatísticas do gerenciamento através da comunicação com o(s) agente(s).

O gerente enxerga cada máquina (agente) como um conjunto de dados e variáveis representando estatísticas do estado atual desta, estatísticas essas disponíveis ao gerente para consulta e/ou edição. Cada máquina gerenciada via SNMP deverá possuir uma MIB associada a si, contendo as estatísticas do equipamento monitorado.

A figura 2.17 apresenta a ideia básica da comunicação entre agentes e gerentes via protocolo SNMP.

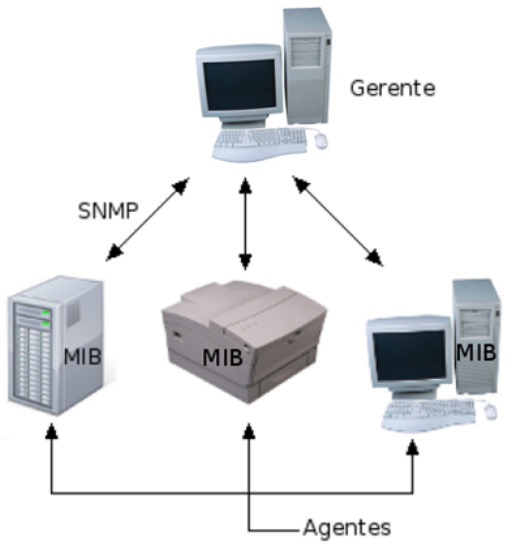


Figura 2.17: Funcionamento do protocolo SNMP

O protocolo SNMP opera na camada de aplicação da rede [15][37]. Obtém, via requisições de um dado gerente a um ou mais agentes, estatísticas de equipamentos conectados a determinados computadores operando como *servidores*. Esta coleta de dados é feita utilizando-se o protocolo de transporte UDP. O tipo de informação mais significativo que pode ser obtido via SNMP consiste na base de dados MIB, que estrutura hierarquicamente os dados de todas as interfaces agregadas ao equipamento de rede pesquisado, organizando assim todas as estatísticas relacionadas a este equipamento em especial.

Os aplicativos do protocolo SNMP se baseiam no mecanismo de busca/alteração, ou seja, estão disponíveis as operações de alteração de um valor de uma estatística, de obtenção dos valores e suas variações. A quantidade de aplicativos disponibilizada é reduzida, o que facilita sua implementação, tornando-a mais simples, estável e flexível. Esta limitação de aplicativos disponíveis ajuda ainda a reduzir o tráfego de mensagens de gerenciamento através da rede. Já a simplicidade do protocolo SNMP permite que este seja utilizado em diversas aplicações de gerenciamento de redes; e sua estabilidade e flexibilidade permitem a possibilidade de acrescentar estatísticas novas à base MIB do equipamento gerenciado [36].

Para realizar o monitoramento de equipamentos rede via SNMP, o gerente envia continuamente mensagens SNMP aos agentes gerenciados, a fim de obter suas respectivas estatísticas.

Os campos de uma mensagem SNMP estão estruturados da seguinte forma:

1. **Mensagem SNMP**: Trata-se da mensagem na sua visão mais geral, baseada em três partes principais. São elas:

    - **version**: Versão do protocolo;
    - **community**: Uma senha criada para o controle de acesso e/ou edição às estatísticas de um agente via SNMP;
    - **SNMP PDU**: Espaço onde ficam registradas as configurações de mensagem, de acordo com seu respectivo tipo, tipo este que pode ser *GetRequest*, *GetNextRequest*, *SetRequest*, *GetResponse* e *Trap*.

2. **GetRequestPDU, GetNextRequestPDU e SetRequestPDU**: Mensagens de envio e recepção de requisições do protocolo SNMP, por parte de um dado gerente. A estrutura do espaço SNMP PDU para este tipo de mensagem possui os seguintes campos:

   - **PDU Type**: Tipo de mensagem SNMP (neste caso, podem ser *GetRequestPDU*, *GetNextRequestPDU* ou *SetRequestPDU*) enviada pelo gerente;
   - **request-id**: Identificador da requisição;
   - **error-status**: Indica se ocorreu erro no processamento da mensagem. Para mensagens do tipo *request* (*GetRequestPDU*, *GetNextRequestPDU* e *SetRequestPDU*), esta variável apresentará sempre valor 0;
   - **error-index**: Quando um *error-status* é diferente de 0, este campo fornece informações adicionais sobre a variável da mensagem SNMP que gerou erro em error-status. Para mensagens do tipo *request* (*GetRequestPDU*, *GetNextRequestPDU* e *SetRequestPDU*), esta variável apresentará sempre valor 0;
   - **variable-bindings**: Uma lista de nomes de variáveis e seus respectivos valores.

3. **GetResponsePDU**: Mensagens de obtenção de resposta a requisições SNMP, para um dado gerente. A estrutura do espaço SNMP PDU para este tipo de mensagem possui os mesmos campos que aparecem nas mensagens de envio e recepção de requisições, com a diferença de que os campos *error-status* e *error-index* podem apresentar valores diferentes de 0;

4. **TrapPDU**: Mensagens de notificações SNMP, enviadas automaticamente para um gerente, com ou sem requisição prévia por parte deste; enquanto que os outros tipos de mensagem são enviados apenas quando ocorre requisição de dados por parte do gerente. A estrutura do espaço SNMP PDU para este tipo de mensagem possui os seguintes campos:

- **PDU Type**: Tipo de mensagem SNMP (neste caso, *TrapPDU*) enviada para o gerente;
- **enterprise**: Tipo do objeto gerador da mensagem *trap*;
- **generic-trap**: Mensagem *trap* genérica, ou mensagem padrão de notificação do protocolo SNMP;
- **specific-trap**: Código da mensagem *trap*;
- **time-stamp**: Intervalo de tempo entre a última (re)inicialização da entidade de rede (agente) e a geração da mensagem *trap*.

5. **variable-bindings**: Listagem de nomes de variáveis e seus valores. Esta listagem segue sempre o padrão de colocar primeiro o nome variável, imediatamente seguido pelo valor contido na mesma.

A figura 2.18 apresenta a estrutura de uma mensagem SNMP que é enviada, quando da requisição de estatísticas sobre um dado agente na rede [57]:

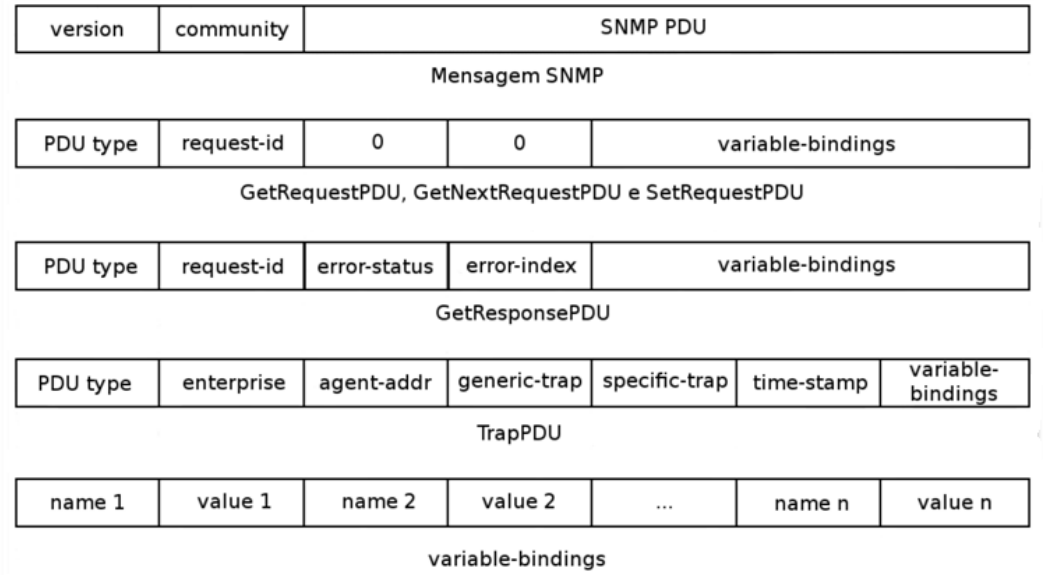


Figura 2.18: Formato de uma mensagem SNMP

A primeira linha da figura 2.18 mostra a estrutura padrão de qualquer mensagem SNMP, com os campos de versão, *community* e SNMP PDU.

As três linhas seguintes apresentam os formatos do campo SNMP PDU para os cinco tipos possíveis de mensagem SNMP (os tipos *GetRequestPDU*, *GetNextRequestPDU* ou *SetRequestPDU* apresentam a mesma forma padrão de SNMP PDU). Por fim, a última linha exibe a estrutura do campo variable-bindings, presente em todos os tipos de mensagem SNMP.

O gerenciamento de redes através do protocolo SNMP é conhecido como modelo de gerenciamento SNMP, ou gerenciamento SNMP, que permite o acompanhamento do estado atual da rede monitorada, em tempo real.

A seguir, seguem exemplos da implementação prática do protocolo SNMP. Executou-se o aplicativo *snmpwalk*, via *prompt*, retornando todas as estatísticas dos equipamentos de rede associados a um dado *servidor*, passado como parâmetro através de seu endereço IP.

Um exemplo de implementação via *prompt* do aplicativo *snmpwalk* pode ser definido da seguinte forma:

snmpwalk [COMMON FLAGS] [OPTIONS] HOST [OID]

Conforme pode ser visto no apêndice C. Para ilustrar este trabalho, serão apresentadas duas formas de execução do aplicativo *snmpwalk*: Uma execução simples, a partir de um dado endereço IP, retornando todos os elementos de rede conectados ao equipamento associado a este endereço; e uma execução mais específica, onde, a partir de um dado *host*, busca-se um atributo específico de um dado elemento de rede conectado a este *host*.

Por medida de simplicidade, os exemplos com o aplicativo *snmpwalk* geral serão rodados a partir de um computador pessoal, com poucos elementos de rede conectados a si. O comando executado terá a seguinte forma:

snmpwalk -v 2c -c public localhost

Tendo apresentado o resultado abaixo:

root@antonio-desktop:~# snmpwalk -v 2c -c public localhost
SNMPv2-MIB::sysDescr.0 = STRING: Linux antonio-desktop 2.6.31-

14-generic #48-Ubuntu SMP Fri Oct 16 14:04:26 UTC 2009 i686
(...)
SNMPv2-MIB::sysName.0 = STRING: antonio-desktop
(...)
SNMPv2-MIB::sysORLastChange.0 = Timeticks: (2) 0:00:00.02
SNMPv2-MIB::sysORID.1 = OID: SNMP-FRAMEWORK-MIB::
snmpFrameworkMIBCompliance
(...)
SNMPv2-MIB::sysORDescr.1 = STRING: The SNMP Management Architecture MIB.
(...)

Exemplificando alguns valores obtidos no caso acima:

- A variável *sysDescr* (.1.3.6.1.2.1.1.1) retornou a descrição do equipamento gerenciado, incluindo o sistema operacional e a data de início da operação do mesmo;

- A variável *sysName* (.1.3.6.1.2.1.1.5) retornou o nome admininstrativo dado ao sistema operacional do equipamento gerenciado;

- A variável *sysORLastChange* (.1.3.6.1.2.1.1.8) retornou o tempo transcorrido desde a última modificação ocorrida em alguma instância do equipamento;

- A variável *sysORID.1* (.1.3.6.1.2.1.1.9.1.2.1) retornou um dos identificadores de funcionalidades que o equipamento em questão é capaz de realizar. Neste caso, o identificador relativo à base de dados MIB;

- A variável *sysORDescr.1* (.1.3.6.1.2.1.1.9.1.3.1) retornou uma das descrições de funcionalidades que o equipamento em questão é capaz de realizar. Neste caso, uma referência à MIB.

No segundo exemplo, são obtidas as interfaces conectadas a um dado equipamento - cujos endereço IP e *community* reais foram omitidos por

questões de segurança - como exemplo do funcionamento do aplicativo *snmpwalk* para obtenção de dados específicos. A execução do aplicativo neste exemplo possui a seguinte forma:

snmpwalk -v 2c -c public 10.0.94.244 .1.3.6.1.2.1.2.2.1.2

Apresentando o resultado abaixo:

root@antonio-desktop:~# snmpwalk -v 2c -c public 10.0.94.244 .1.3.6.1.2.1.2.2.1.2
IF-MIB::ifDescr.1 = STRING: Serial3/0
(...)
IF-MIB::ifDescr.3 = STRING: FastEthernet0/0
(...)
IF-MIB::ifDescr.12 = STRING: Ethernet2/0
(...)
IF-MIB::ifDescr.24 = STRING: Null0
IF-MIB::ifDescr.25 = STRING: Loopback0
(...)
IF-MIB::ifDescr.27 = STRING: Multilink100

É interessante observar no exemplo acima que o aplicativo *snmpwalk*, ao listar as interfaces de um equipamento, consegue detectar tanto as interfaces reais, ou seja, as que possuem estrutura física, como as dos tipos Serial, FastEthernet ou Ethernet; quanto as interfaces virtuais, que possuem somente estrutura lógica, como as de tipo Null, Loopback ou Multilink.

2.2.6 O protocolo Spanning Tree - STP

O protocolo Spanning Tree é implementado em equipamentos que reencaminham pacotes dentro de redes. Sua principal função é impedir que pacotes transitem infinitamente entre os equipamentos de rede [61]. Para compreender melhor seu funcionamento, é preciso introduzir um breve histórico

sobre redes de computadores.

A primeira rede de computadores foi desenvolvida pelo governo dos Estados Unidos durante o período da história conhecido como Guerra Fria. A intenção era conectar os diversos pontos de presença do departamento de defesa americano e algumas instituições de pesquisa. Nessa ocasião, os Estados Unidos pretendiam criar um sistema de comunicação alternativo, caso o sistema de comunicação primário sofresse algum tipo de ataque, por parte do bloco Comunista. Com o tempo, outros departamentos e instituições foram se conectando àquela rede, que recebeu o nome de ARPANET [62].

Ao final dos anos 70, ocorreu a necessidade de atualização do protocolo de comutação de pacotes, devido a dimensão que o *backbone* ARPANET adquiriu.

O *backbone* ARPANET substituiu o seu protocolo de comunicação original pelo TCP/IP, que começou a ser utilizado ainda nos anos 70, sendo, até hoje, o protocolo padrão para comunicação na Internet.

A figura 2.19 apresenta o mapa de rede do *backbone* ARPANET no começo dos anos 80.

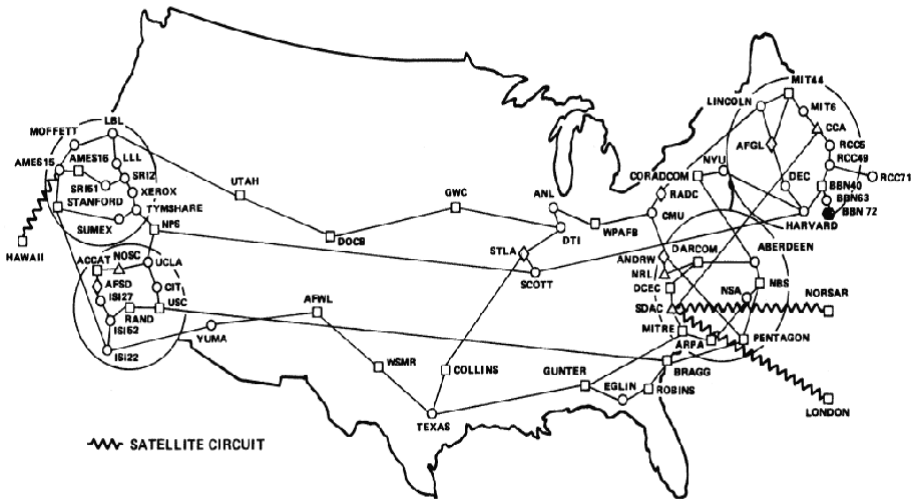


Figura 2.19: *Backbone* ARPANET (1980) - obtido em www.let.leidenuniv.nl

No início dos anos 80, alguns fabricantes se reuniram e criaram uma organização chamada ISO. Esta organização, em conjunto com outros órgãos, criou padrões para a fabricação de equipamentos e propuseram padrões para o tratamento da comunicação em redes de computadores. O modelo OSI [15] padronizou a forma como os dados a serem transmitidos deveriam ser encapsulados em pacotes e como estes pacotes seriam enviados, recebidos e interpretados dentro de uma arquitetura em camadas. Ao todo, são sete camadas, e cada protocolo de comunicação opera em uma determinada camada.

A primeira camada do modelo OSI, chamada física, trata das interfaces dos elementos de uma rede e a conexão física estabelecida entre eles. A camada seguinte, de enlace, é uma camada de protocolos que provêm comunicação entre dois pontos adjacentes em uma rede ou entre pontos em redes diferentes.

O *switch* é um equipamento que atua nesta camada, além de segmentar uma rede de computadores, ele possui a tarefa de reencaminhar pacotes entre diferentes domínios. Esta técnica é conhecida na literatura como *bridging* [61], que permite, a dois domínios que utilizam protocolos de comunicação diferentes, a comunicação através do MAC Address (endereço de máquina, ou endereço físico). Entretanto, quando há de mais de dois *switches* na rede e quando estes equipamentos formam um anel, existe a possibilidade de ocorrência de um fenômeno conhecido como *loop* de rede, um problema que pode causar a queda da comunicação. O *loop* de rede é caracterizado quando um mesmo pacote permanece trafegando infinitamente entre as interfaces dos *switches*.

Para solucionar este problema, o protocolo Spanning Tree impede a ocorrência de *loops* de rede, configurando cada interface de um *switch* em estado *forwarding* ou *blocking* [15][61]. A figura 2.20(a) apresenta quatro *switches* interconectados sem o protocolo Spanning Tree (cenário com *loop* de rede). A figura 2.20(b) mostra a configuração das conexões *forwarding* com o protocolo Spanning Tree, quando do bloqueio do tráfego entre os dois *switches* à direita. A figura 2.20(c) apresenta um caminho alternativo, caso, no exemplo (b), haja um problema de comunicação entre os *switches* mais abaixo na figura.

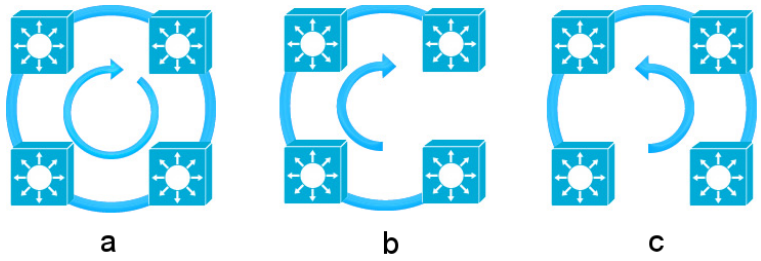


Figura 2.20: Funcionamento do protocolo Spanning Tree

2.2.6.1 Árvore de Custo Mínimo - MST

O estudo da Teoria de Grafos convencionou como árvore um grafo não-direcionado (havendo uma aresta que conecta dois vértices i e j quaisquer, ela vale tanto como aresta ij, como aresta ji), conexo (todos os vértices do grafo podem ser alcançados a partir de um dado vértice qualquer por qualquer método de busca em grafos) e acíclico (não existem ciclos dentro do grafo, ou seja, existe um único caminho possível de ser realizado entre dois vértices quaisquer deste grafo).

A estrutura de uma árvore é simples de ser implementada, computacionalmente falando, devido à simplicidade de suas características, permitindo assim uma gama de aplicações para as árvores [4]. Exemplos da simplicidade estrutural das árvores podem ser observados em fatos como:

- Toda árvore com n vértices terá n - *1* arestas;
- um grafo não-direcionado somente poderá ser considerado como uma árvore se e somente se, dados dois vértices u, v, houver somente um caminho ligando-os.

O protocolo Spanning Tree deriva da Teoria dos Grafos, implementando um algoritmo que propõe encontrar o caminho com o menor custo possível entre todos os vértices de uma dada rede, em relação a um desses pontos, chamado de nó raiz. Pode se tratar de custo operacional, financeiro, tempo de processamento, de distância física entre pontos da rede, etc [63].

Este algoritmo monta uma árvore a partir de uma rede conexa. Redes deste tipo são caracterizadas pelo fato de todos os seus vértices se "enxer-

garem", ou seja, existem um ou mais caminhos ligando dois vértices quaisquer dentro da rede em questão.

Na literatura especializada, as árvores resultantes desta operação são chamadas de Árvores de Custo Mínimo, ou *Minimum Spanning Tree* - MST. Alguns algoritmos podem ser usados para a solução MST de uma rede, como por exemplo, os algoritmos de Kruskal [64] e Prim [65]. Os dois algoritmos atingem os mesmos resultados, porém com algumas diferenças em suas implementações.

Uma Árvore de Custo Mínimo não deve conter ciclos, pois isto aumenta seu custo, custo este que pode ser removido, sem prejuízo à criação da árvore, a qual deve possuir o menor valor de custo possível.

O processo do geração da Árvore de Custo Mínimo de uma dada rede também é conhecido como convergência [66].

2.2.6.2 Algoritmos empregados

Aqui serão abordados os dois algoritmos utilizados para o cálculo da Árvore de Custo Mínimo. São eles: os algoritmos de Kruskal e de Prim.

Para o entendimento destes algoritmos, deve-se introduzir duas definições importantes sobre grafos:

- Um terminal isolado é um vértice que não apresenta conexão com nenhum outro vértice do grafo;
- Um fragmento é um conjunto de vértices, onde há conexões suficientes entre eles, de modo que, a partir de um dado vértice, pode-se alcançar qualquer outro vértice deste fragmento [65].

Conforme apresentado na figura 2.21, os vértices **2**, **4** e **9** são exemplos de terminais isolados; contudo, temos também dois fragmentos: um composto pelos vértices **1**, **5**, **6** e **7**, e outro composto pelos vértices **3** e **8**.

- Algoritmo de Kruskal

  A ideia deste algoritmo consiste em selecionar, a cada iteração, a aresta com menor custo no grafo original, devendo obedecer às seguintes condições:

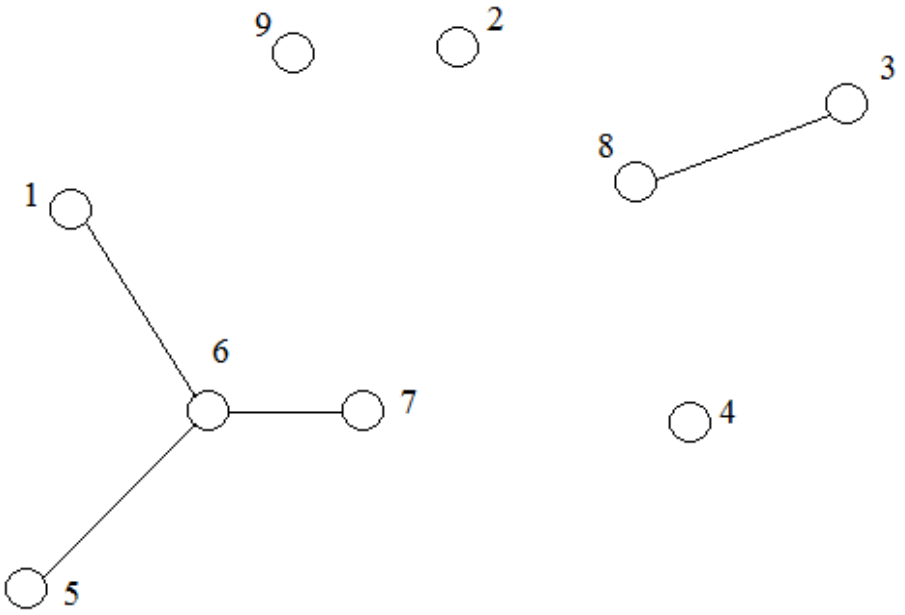


Figura 2.21: Exemplos de fragmentos e terminais isolados

– Não ter sido escolhida anteriormente;
– Não gerar ciclos na Árvore de Custo Mínimo.

O algoritmo pode ser resumido no conjunto de passos abaixo:

1. Seja um grafo com n vértices e m arestas; onde, para cada aresta, haja um valor numérico associado, referente ao seu respectivo custo;
2. Cria-se então um fragmento para cada vértice do grafo;
3. Enquanto houver dois ou mais fragmentos no grafo, fazer:
   (a) Selecionar a aresta com menor custo ligando dois vértices de fragmentos distintos;
   (b) A nova aresta será então incorporada à Árvore de Custo Mínimo; e um dos fragmentos ligados por ela será removido, com todos os seus vértices sendo então incorporados ao outro fragmento existente na ligação.

Deve-se observar que, buscando apenas por arestas que liguem vértices de fragmentos diferentes, se garante a não ocorrência de ciclos na Árvore de Custo Mínimo. Isto se deve ao fato de que, para haver ciclos, seria necessário conectar dois vértices que estejam dentro de um mesmo fragmento, através de mais de um caminho.

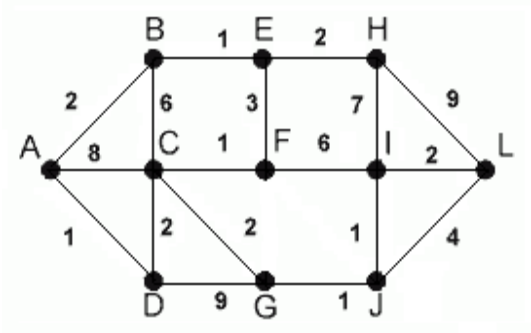


Figura 2.22: Exemplo de grafo conexo

Aplicando-se o algoritmo de Kruskal à figura 2.22, teremos a realização dos seguintes passos:

1. Criação de um fragmento para cada vértice do grafo;
2. Escolha da aresta com menor custo no grafo, que ligue vértices pertencentes a fragmentos distintos. Neste caso, poderão ser escolhidas as arestas **AD**, **BE**, **CF**, **GJ** ou **IJ** (ambas com custo 1). Será escolhida a aresta **AD**;
3. Vértice **D** incorporado ao fragmento de **A**, gerando assim o fragmento **{A, D}**;
4. Escolha da aresta com menor custo no grafo, que ligue vértices pertencentes a fragmentos distintos. Neste caso, poderão ser escolhidas as arestas **BE**, **CF**, **GJ** ou **IJ** (ambas com custo 1). Será escolhida a aresta **BE**;
5. Vértice **E** incorporado ao fragmento de **B**, gerando assim o fragmento **{B, E}**;
6. Escolha da aresta com menor custo no grafo, que ligue vértices pertencentes a fragmentos distintos. Neste caso, poderão ser es-

colhidas as arestas **CF**, **GJ** ou **IJ** (ambas com custo 1). Será escolhida a aresta **CF**;

7. Vértice **F** incorporado ao fragmento de **C**, gerando assim o fragmento **{C, F}**;
8. Escolha da aresta com menor custo no grafo, que ligue vértices pertencentes a fragmentos distintos. Neste caso, poderão ser escolhidas as arestas **GJ** ou **IJ** (ambas com custo 1). Será escolhida a aresta **GJ**;
9. Vértice **J** incorporado ao fragmento de **G**, gerando assim o fragmento **{G, J}**;
10. Escolha da aresta com menor custo no grafo, que ligue vértices pertencentes a fragmentos distintos. Neste caso, será escolhida a aresta **IJ**. (custo 1);
11. Vértice **I** incorporado ao fragmento de **J**, gerando assim o fragmento **{G, I, J}**;
12. Escolha da aresta com menor custo no grafo, que ligue vértices pertencentes a fragmentos distintos. Neste caso, poderão ser escolhidas as arestas **AB**, **CD**, **CG**, **EH** ou **IL** (ambas com custo 2). Será escolhida a aresta **AB**;
13. Vértice **B** incorporado ao fragmento de **A**, gerando assim o fragmento **{A, B, D, E}**;
14. Escolha da aresta com menor custo no grafo, que ligue vértices pertencentes a fragmentos distintos. Neste caso, poderão ser escolhidas as arestas **CD**, **CG**, **EH** ou **IL** (ambas com custo 2). Será escolhida a aresta **CD**;
15. Vértice **D** incorporado ao fragmento de **C**, gerando assim o fragmento **{A, B, C, D, E, F}**;
16. Escolha da aresta com menor custo no grafo, que ligue vértices pertencentes a fragmentos distintos. Neste caso, poderão ser escolhidas as arestas **CG**, **EH** ou **IL** (ambas com custo 2). Será escolhida a aresta **CG**;

17. Vértice **G** incorporado ao fragmento de **C**, gerando assim o fragmento **{A, B, C, D, E, F, G, I, J}**;
18. Escolha da aresta com menor custo no grafo, que ligue vértices pertencentes a fragmentos distintos. Neste caso, poderão ser escolhidas as arestas **EH** ou **IL** (ambas com custo 2). Será escolhida a aresta **EH**;
19. Vértice **H** incorporado ao fragmento de **E**, gerando assim o fragmento **{A, B, C, D, E, F, G, H, I, J}**;
20. Escolha da aresta com menor custo no grafo, que ligue vértices pertencentes a fragmentos distintos. Neste caso, será escolhida a aresta **IL** (custo 2);
21. Vértice **L** incorporado ao fragmento de **I**, gerando assim o fragmento **{A, B, C, D, E, F, G, H, I, J, L}**;
22. Como todos os vértices do grafo estão incluídos em um único fragmento, a geração da Árvore de Custo Mínimo se encerra aqui.

A figura 2.23, apresenta, em vermelho, as arestas da Árvore de Custo Mínimo criada pelo algoritmo de Kruskal descrito nesta sessão.

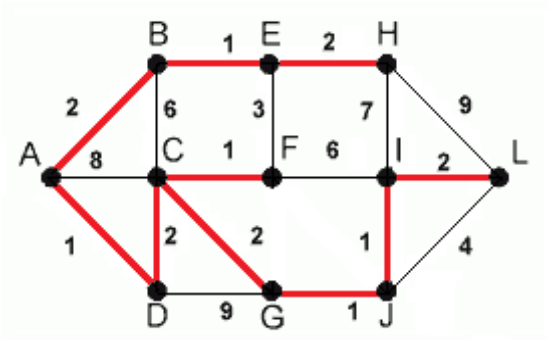


Figura 2.23: Resultado do algoritmo de Kruskal

Já a figura 2.24 apresenta a Árvore de Custo Mínimo gerada para o grafo da figura 2.22.

- Algoritmo de Prim

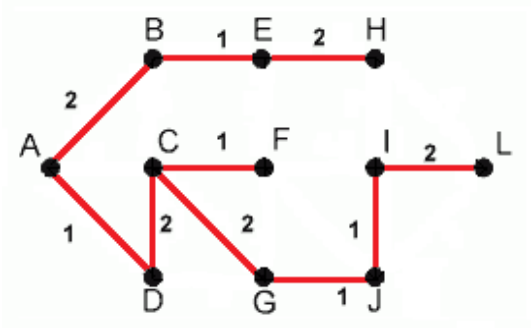


Figura 2.24: Árvore de Custo Mínimo (Algoritmo de Kruskal)

A construção deste algoritmo é resumida pelo seguinte conjunto de passos:

1. Seja um dado grafo com n vértices e m arestas; onde, para cada aresta, haja um valor numérico associado, referente ao seu custo;
2. Escolhe-se um vértice **u** como raiz, colocando-o como o único vértice do fragmento **A**, que reúne os vértices alcançados pela busca;
3. Enquanto houverem vértices do grafo fora de **A**, fazer:
   (a) Para cada vértice de **A**, procura-se pela aresta de menor custo que conecte **A** a algum vértice externo a **A**. Ao final deste processo, a aresta com menor custo dentre todas as encontradas será escolhida para entrar na Árvore de Custo Mínimo;
   (b) A nova aresta será então inserida na Árvore de Custo Mínimo; e o vértice externo a **A** que é conectado a **A** através dela será adicionado ao conjunto de vértices do fragmento **A**.

Deve-se observar que, buscando arestas que liguem os vértices de **A** a vértices externos a **A**, garante-se a não ocorrência de ciclos no grafo da Árvore de Custo Mínimo, da mesma forma que no algoritmo de Kruskal [65].

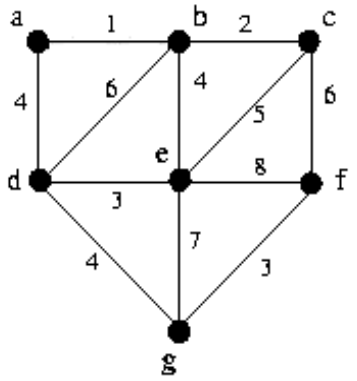


Figura 2.25: Exemplo de grafo, para demonstração do algoritmo de Prim

Aplicando-se o algoritmo de Prim à figura 2.25, iniciando a busca a partir do vértice **a**, teremos a realização dos seguintes passos:

1. Inclusão do vértice **a** ao fragmento **A**, inicialmente vazio;
2. Escolha de uma aresta com custo mínimo que ligue **a** a qualquer outro vértice não pertencente a **A**. Neste caso, a aresta que satisfaz a condição estabelecida no passo 3(a) é **ab** (custo 1);
3. Inclusão do vértice **b** ao fragmento **A**;
4. Escolha de uma aresta com custo mínimo que ligue o conjunto **{a, b}** a qualquer outro vértice não pertencente a **A**. Neste caso, a aresta que satisfaz a condição estabelecida no passo 3(a) é **bc** (custo 2);
5. Inclusão do vértice **c** ao fragmento **A**;
6. Escolha de uma aresta com custo mínimo que ligue o conjunto **{a, b, c}** a qualquer outro vértice não pertencente a **A**. Como as arestas **ad** e **be** apresentam o menor custo possível neste passo (custo 4), qualquer uma das duas poderá ser escolhida. Escolheremos então a aresta **ad**;
7. Inclusão do vértice **d** ao fragmento **A**;
8. Escolha de uma aresta com custo mínimo que ligue o conjunto **{a, b, c, d}** a qualquer outro vértice não pertencente a **A**. Neste

caso, a aresta que satisfaz a condição estabelecida no passo 3(a) é **de** (custo 3);

9. Inclusão do vértice **e** ao fragmento **A**;
10. Escolha de uma aresta com custo mínimo que ligue o conjunto **{a, b, c, d, e}** a qualquer outro vértice não pertencente a **A**. Neste caso, a aresta que satisfaz a condição estabelecida no passo 3(a) é **dg** (custo 4);
11. Inclusão do vértice **g** ao fragmento **A**;
12. Escolha de uma aresta com custo mínimo que ligue o conjunto **{a, b, c, d, e, g}** a qualquer outro vértice não pertencente a **A**. Neste caso, a aresta que satisfaz a condição estabelecida no passo 3(a) é **fg** (custo 3);
13. Inclusão do vértice **f** ao fragmento **A**;
14. Como todos os vértices do grafo estão incluídos em **A**, a geração da Árvore de Custo Mínimo se encerra aqui.

A figura 2.26 apresenta a Árvore de Custo Mínimo gerada pelo algoritmo de Prim com base no grafo da figura 2.25.

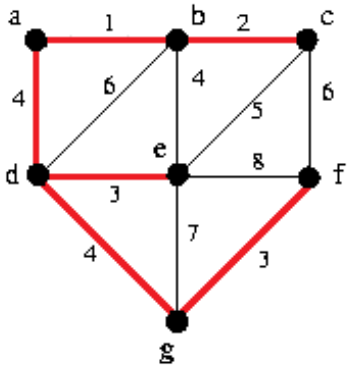


Figura 2.26: Árvore de Custo Mínimo gerada pelo algoritmo de Prim

Já a figura 2.27 apresenta a Árvore de Custo Mínimo gerada para o grafo da figura 2.25.

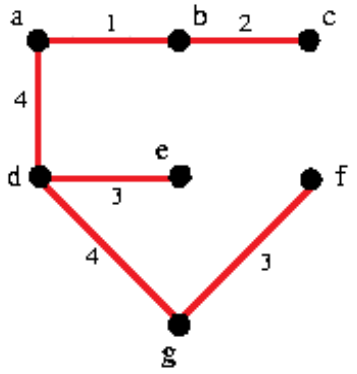


Figura 2.27: Árvore de Custo Mínimo (Algoritmo de Prim)

2.2.6.3 Versões

Inicialmente, a extinta empresa *Digital Equipment Corporation* - DEC desenvolveu a primeira versão do protocolo Spanning Tree. Posteriormente, o protocolo foi adequado ao padrão IEEE, levando à criação de sua versão 802.1d, em 1990, com posterior atualização em 1998 [67]. Todos os equipamentos de rede fabricados pela empresa Cisco utilizam versões do protocolo Spanning Tree com padrão IEEE, padrão este não compatível com a versão original criada pela DEC [66].

Cabe observar que todas as versões posteriores à 802.1d do Spanning Tree seguem o padrão IEEE.

A versão seguinte do protocolo Spanning Tree é conhecida como *Rapid Spanning Tree Protocol* - RSTP, ou 802.1w, criada em 2001 [67]. Apresenta as seguintes melhorias, em relação à versão 802.1d [66]:

- ***PortFast***: Exclui, do processo de convergência do protocolo Spanning Tree, portas que, sabidamente, não causarão *loops* na rede. Normalmente utilizado para portas conectadas a computadores atuando como terminais em uma rede;

- ***UplinkFast***: Esta funcionalidade faz com que, se um dado caminho de rede utilizado por um *switch* deixe de funcionar, este *switch* tenha previamente definido um caminho alternativo para substituir o caminho anterior. Recurso destinado a *uplinks* (conexões entre *switches*);

- ***BackboneFast***: Detecta inconsistências na topologia de uma rede, devendo ser aplicada a todos os *switches* desta.

Uma limitação da versão 802.1d está no fato de que todos os elementos de uma dada rede estão dentro de uma única instância Spanning Tree, conhecida como *Common Spanning Tree* - CST, deixando-os todos com o mesmo ponto de bloqueio na topologia para prevenção de *loops* de rede, o que centraliza excessivamente a gerência da rede.

Como forma de se contornar tal limitação, a empresa Cisco desenvolveu as versões *Per-Vlan Spanning Tree* - PVST e *Per-Vlan Spanning Tree Plus* - PVST+ (também conhecido como 802.1q, e criado em 1998), possibilitando que os *switches* de uma dada rede criem automaticamente instâncias STP (no caso do protocolo PVST), ou RSTP (no caso do protocolo PVST+) por cada sub-rede (VLAN). Com isto, é possível estabelecer um ponto de bloqueio por VLAN, utilizando diversos *switches* como nós raiz de cada VLAN [67].

No entanto, a implementação dos protocolos PVST e PVST+ apresenta consumo elevado de recursos de rede para se fazer o cálculo das Árvores de Custo Mínimo (convergência).

Para resolver este novo problema, foi criada, em 2002, a versão *Multiple Instance Spanning Tree* - MISTP, também conhecida como *Multiple Spanning Tree* - MSTP (802.1s), onde é possível mapear várias VLAN's, utilizando para isto um número reduzido de instâncias Spanning Tree independentes umas das outras, otimizando o uso de *links* e de CPU's. Desta forma é possível criar, para todas as sub-redes que compartilhem de uma mesma topologia, uma instância Spanning Tree padronizada, reduzindo assim a utilização de recursos de rede [67].

2.2.6.4 Mensagens BPDU

Para realizar o processo de convergência em uma dada rede, o protocolo Spanning Tree faz com que os equipamentos desta rede troquem entre si periodicamente mensagens contendo informações relacionadas ao estado atual de cada equipamento, mantendo cada ponto de rede sempre atualizado, em relação à situação atual de seus vizinhos.

Estas mensagens são conhecidas como mensagens *Bridge Protocol Data Units* - BPDU, e se dividem em dois tipos: configuração, utilizados para gerar a Árvore de Custo Mínimo; e notificação de mudança de topologia (*Topology Change Notification* - TCN), empregados para redesenhar a Árvore de Custo Mínimo, sempre que isto for necessário [66] [68].

A mensagem BPDU de configuração apresenta os seguintes campos [68] [69]:

- **Protocol ID**: Utiliza 16 *bits*, e identifica o protocolo de rede utilizado;
- **Version**: Utiliza 8 *bits*, e identifica a versão do protocolo Spanning Tree utilizada;
- **Message Type**: Utiliza 8 *bits*, e identifica qual o tipo de mensagem BPDU enviada;
- **Flags**: Utilizam 8 *bits*, e identificam os índices responsáveis por sinalizar mudanças de topologia na rede. São eles:
    1. ***Topology Change Nofitication* - TC (1 *bit*)**: utilizado pelo *switch* raiz para reconhecimento de mudança na topologia;
    2. **Proposal (1 *bit*)**: caso a porta para a qual a mensagem BPDU foi enviada esteja com status *blocking*, este índice, caso esteja habilitado, irá propor o desbloqueio da porta, causando mudança de topologia. Utilizado no protocolo RSTP;
    3. **Port Role (2 *bits*)**: aponta o status atual da porta do *switch* emissor da mensagem. Utilizado no protocolo RSTP;
    4. **Learning (1 *bit*)**: caso esteja habilitado, indica que a porta do *switch* emissor da mensagem se encontra com status *learning*. Utilizado no protocolo RSTP;
    5. **Forwarding (1 *bit*)**: caso esteja habilitado, indica que a porta do *switch* emissor da mensagem se encontra com status *forwarding*. Utilizado no protocolo RSTP;

6. **Agreement (1 *bit*)**: estará habilitado, caso a porta para a qual a mensagem BPDU foi enviada tenha recebido e aceito proposta de desbloqueio. Utilizado no protocolo RSTP;
7. ***Topology Change Notification Acknowledgment* - TCA (1 *bit*)**: comunica ao *switch* emissor da mensagem BPDU original, que os dados da topologia enviados foram lidos e salvos pelo *switch* destinatário.

- **Root ID**: Utiliza 64 *bits*, e apresenta o número identificador do equipamento atualmente utilizado como raiz pelo protocolo Spanning Tree;
- **Root Path Cost**: Utiliza 32 *bits*, e indica o custo acumulado desde a raiz até o equipamento emissor da mensagem;
- **Bridge ID**: Utiliza 64 *bits*, e é o número identificador gerado para o *switch* emissor da mensagem;
- **Port ID**: Utiliza 16 *bits*, e contém o número identificador da porta emissora da mensagem BPDU corrente;
- **Max Age**: Utiliza 16 *bits*, e determina o intervalo máximo de tempo de espera, em milisegundos, para que a mensagem BPDU possa ser recebida pelo equipamento receptor da mensagem. Caso este tempo seja excedido, a mensagem será descartada, e o *switch* receptor interpretará isto como uma mudança de topologia na rede;
- **Forward Delay**: Utiliza 16 *bits*, e indica o intervalo de tempo em que a porta deverá permanecer em um determinado status;
- **Message Age**: Utiliza 16 *bits*, e o intervalo de tempo transcorrido, em milisegundos, que uma dada mensagem BPDU levou para alcançar seu destino;
- **Hello Time**: Utiliza 16 *bits*, e identifica o tempo gasto para a geração da mensagem BPDU.

A mensagem BPDU TCN possui somente os campos Protocol ID, Version e Message Type, o qual assume o valor 128, quando se trata de uma mensagem BPDU TCN. Para mensagens BPDU de configuração, o campo Message Type assume valor 0 [68].

A figura 2.28 apresenta um exemplo de mensagem BPDU de configuração.

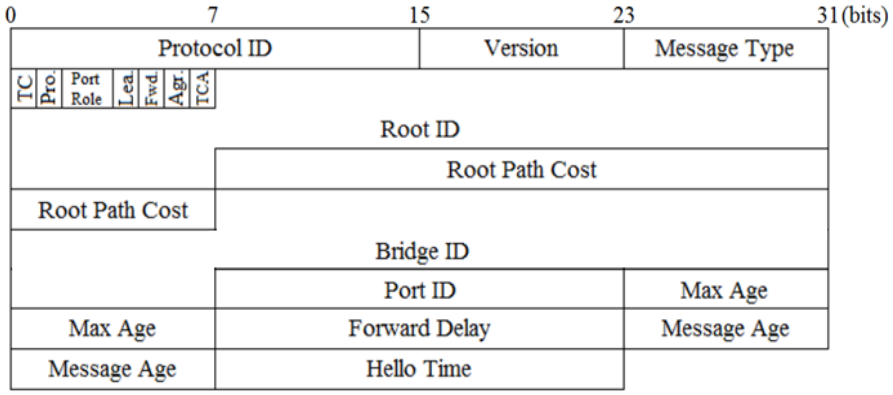


Figura 2.28: Mensagem BPDU

2.3 Aplicativos

2.3.1 O aplicativo PING

O aplicativo *Packet Internet Grouper* - PING tem como finalidade fazer a verificação de conectividade entre dois endereços IP quaisquer, retornando o tempo de resposta do endereço IP de destino, medido em milisegundos. Assim, o uso do aplicativo PING é um teste importante para o gerenciamento de redes, permitindo também estimar uma medida da velocidade de transmissão de dados entre o computador emissor e o receptor da mensagem enviada [17].

O aplicativo PING costumava ser utilizado como um indicador de habilidade de uma máquina para enviar e receber datagramas IP, de modo que se for possível "pingar" um dado endereço IP, é então possível estabelecer uma conexão com o mesmo. No entanto, com a prática da filtragem de pacotes como medida de segurança, não é mais possível ter certeza da impossibilidade

de se estabelecer uma conexão entre dois dispositivos de rede, em caso de insucesso do aplicativo PING, já que muitos *firewalls* desabilitam datagramas ICMP. Um dos motivos para isto é a ocorrência de ataques baseados no protocolo ICMP, como o "PING OF DEATH", que utiliza o aplicativo PING com datagramas de grandes tamanhos, sobrecarregando o alvo [17].

O aplicativo PING envia datagramas ICMP de requisição para uma dada máquina, e espera o envio de um datagrama ICMP de resposta. O formato da mensagem ICMP, utilizada no aplicativo PING, é apresentado na figura 2.29, onde o tipo terá valor 0 (resposta) ou 8 (requisição) [17][22].

Tipo (0 ou 8)	Código (0)	Checksum
Identificador		Número de sequência
Dados opcionais		

Figura 2.29: Estrutura de datagrama ICMP para comunicação com PING

Um dos campos mais importantes para o estudo do aplicativo PING é o TTL, cuja descrição pode ser encontrada na seção 2.2.1.1 desta dissertação. O campo TTL impede, por exemplo, que um datagrama fique indefinidamente circulando pela rede, ao tentar alcançar seu destino. Normalmente, o campo TTL é definido com valor inicial de 60 saltos, considerado suficiente para os datagramas chegarem aos seus destinos. Em cada equipamento de rede por onde um dado datagrama passa, seu respectivo valor do campo TTL é decrementado em uma unidade. Caso o campo TTL chegue a zero, uma mensagem ICMP é enviada à máquina de origem e o pacote é descartado, exceto se o datagrama tiver alcançado seu destino naquele exato momento, evitando-se assim a criação de *loops* e consequentemente um possível congestionamento de dados na rede [17].

Um exemplo que mostra o caminho percorrido pelos datagramas ICMP enviados pelo PING é apresentado, na figura 2.30[17].

No caso de uma requisição PING, o campo de dados opcionais do datagrama ICMP é então utilizado para armazenar a data e horário em que o datagrama de requisição foi enviado. Quando a máquina remetente do

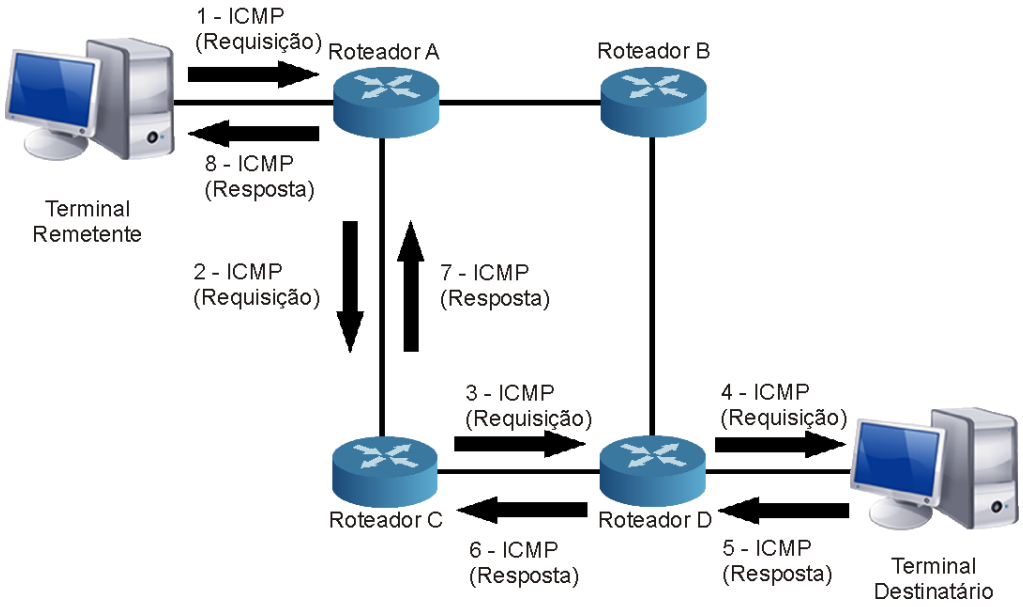


Figura 2.30: Percurso de um datagrama ICMP enviado via aplicativo PING

datagrama receber o datagrama de resposta, esta pode então medir o tempo transcorrido entre o início da emissão e o término da resposta, definido como o tempo de resposta do aplicativo PING, também conhecido como *Round Trip Time* - RTT [17]. O valor obtido como RTT deve ser utilizado como comparação entre diversas requisições PING enviadas, pois o aplicativo PING não possui prioridade em relação a outras tarefas do sistema operacional, ou seja, se houver uma outra tarefa pendente, esta será realizada antes do aplicativo PING, o que pode ocasionar um pico de tempo de resposta em certo instante – daí a necessidade de se ter várias requisições PING enviadas em certo intervalo de tempo, permitindo assim uma comparação mais correta do tempo de resposta [17].

Na resposta do aplicativo PING são apresentados 3 tempos de RTT, correspondentes aos tempos mínimo e máximo de resposta do grupo de requisições PING enviadas, mais o tempo médio de resposta do mesmo grupo; o tamanho do datagrama e o valor de TTL configurado no aplicativo PING. Grandes diferenças nos valores de RTT indicam rede congestionada ou com algum problema técnico em dado momento, como mau funcionamento de um roteador, por exemplo. O aplicativo PING utiliza datagramas pequenos,

para evitar que o tamanho destes comprometa o tempo de resposta, bem como evitar também congestionamentos desnecessários na rede [17].

Abaixo segue um exemplo do funcionamento do aplicativo PING, rodando em um sistema operacional Linux, baseado na distribuição Ubuntu, versão 9.10, utilizada neste projeto (maiores detalhes sobre este sistema operacional serão discutidos no capítulo 3):

root@antonio-desktop:~# ping 100.0.94.244
PING 100.0.94.244 (100.0.94.244) 56(84) *bytes* of data.
64 *bytes* from 100.0.94.244: icmp_seq=1 ttl=128 time=17.3 ms
64 *bytes* from 100.0.94.244: icmp_seq=2 ttl=128 time=16.9 ms
64 *bytes* from 100.0.94.244: icmp_seq=3 ttl=128 time=16.6 ms
64 *bytes* from 100.0.94.244: icmp_seq=4 ttl=128 time=15.8 ms
64 *bytes* from 100.0.94.244: icmp_seq=5 ttl=128 time=15.9 ms
64 *bytes* from 100.0.94.244: icmp_seq=6 ttl=128 time=15.6 ms
64 *bytes* from 100.0.94.244: icmp_seq=7 ttl=128 time=17.1 ms
64 *bytes* from 100.0.94.244: icmp_seq=8 ttl=128 time=16.3 ms
64 *bytes* from 100.0.94.244: icmp_seq=9 ttl=128 time=17.1 ms
64 *bytes* from 100.0.94.244: icmp_seq=10 ttl=128 time=17.2 ms
64 *bytes* from 100.0.94.244: icmp_seq=11 ttl=128 time=16.9 ms

— 100.0.94.244 ping statistics –
11 packets transmitted, 11 received, 0% packet loss, time 10021ms
rtt min/avg/max/mdev = 15.661/16.652/17.321/0.578 ms

No exemplo acima, foram enviados 11 datagramas de 64 *bytes* cada um, para um equipamento de rede possuidor do endereço IP 100.0.94.244. Todos foram recebidos com sucesso e as respectivas mensagens de resposta bem-sucedida foram enviadas para o emissor. O valor *default* de TTL está configurado em 128 saltos, ou seja, se em até 128 saltos na rede, o datagrama não alcançar seu destino, o mesmo será descartado e uma mensagem ICMP de erro será enviada ao equipamento emissor. Os tempos de resposta (RTT)

foram, em milisegundos:

- Menor tempo de resposta: 15,661;
- Maior tempo de resposta: 17,321;
- Tempo médio de resposta: 16,652.

Por questões de segurança, o verdadeiro endereço IP utilizado neste exemplo foi omitido.

2.3.2 O aplicativo Traceroute

O aplicativo Traceroute tem como propósito fornecer como resposta à sua utilização, o caminho percorrido, do terminal de onde a requisição foi enviada, até o terminal passado como parâmetro de entrada. O terminal de destino pode ser passado como endereço IP, ou ainda no formato de endereço HTTP (endereço de página WEB).

O aplicativo Traceroute funciona com o envio de 3 segmentos UDP, os quais estarão com o campo TTL configurado com o valor 1. Ao chegar ao primeiro equipamento de rede intermediário, o valor do campo TTL será zerado, acarretando o envio de uma mensagem ICMP de controle para o terminal remetente, informando o ponto de rede onde o valor do campo TTL tornou-se igual a zero.

A seguir, o procedimento listado no parágrafo anterior é repetido, com uma diferença: o valor do campo TTL desta vez estará configurado com o valor 2. Com isso, serão realizados dois saltos na rede, até que a mensagem ICMP seja novamente enviada ao terminal remetente, dando-lhe assim o conhecimento sobre o próximo ponto intermediário de rede no caminho.

Este procedimento será repetido, com sucessivos incrementos no campo TTL do segmento UDP, até que os segmentos UDP alcancem o terminal destinatário da mensagem, especificado como parâmetro de entrada no aplicativo Traceroute. Neste caso, a mensagem ICMP mudará de tempo excedido para porta inacessível [17].

A figura 2.31 ilustra o conceito apresentado no parágrafo anterior:

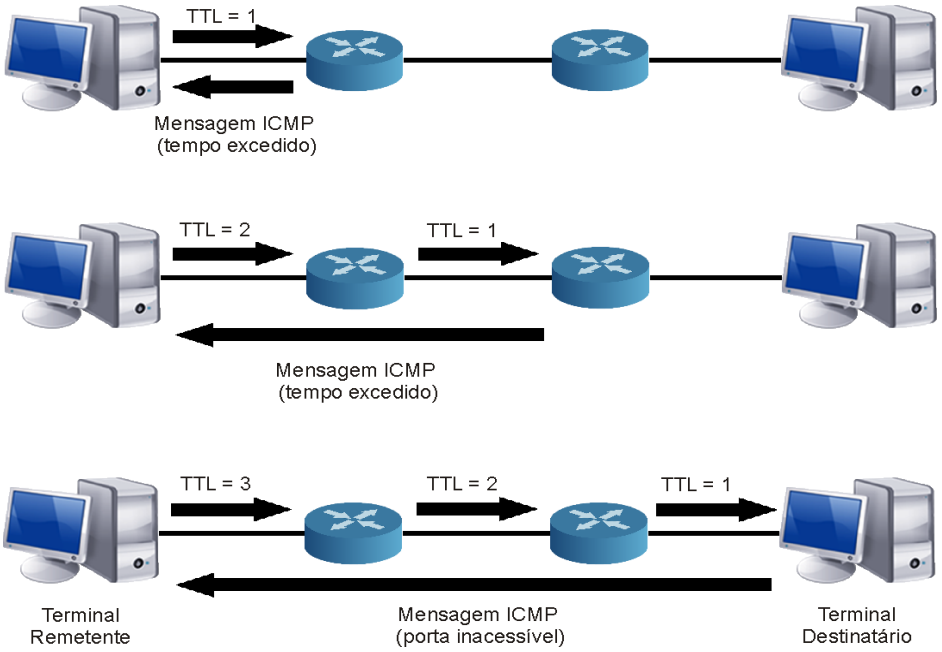


Figura 2.31: Conceito de funcionamento do aplicativo Traceroute

Já a figura 2.32 apresenta um exemplo da mensagem ICMP, enviada na utilização do aplicativo Traceroute:

Tipo(3-11)	Código(0-3)	Checksum
Identificador		Número de sequência
Dados opcionais		

Figura 2.32: Mensagem ICMP enviada no uso do aplicativo Traceroute

Dependendo do tipo de mensagem enviada (tempo excedido ou porta inacessível), os valores dos campos tipo e código poderão ser, respectivamente, 11 e 0 (em caso de tempo excedido); ou 3 e 3 (em caso de porta inacessível).

A cada salto realizado na execução do aplicativo Traceroute, será exibido o endereço IP do ponto de rede intermediário retornado na mensagem ICMP de tempo excedido, juntamente com os tempos de resposta (RTT's) de cada segmento UDP. Para cada vez que o tempo de *timeout* de 3 segundos for

excedido por um segmento, o valor de RTT retornado para este segmento virá, na linha de código, substituído por um asterisco (*). Caso a maioria dos segmentos UDP enviados no salto apresentarem erro de time-out, a execução do aplicativo Traceroute é finalizada sem sucesso [17].

Para exemplificar o uso do aplicativo Traceroute, serão apresentados dois exemplos práticos de sua utilização. O primeiro exemplo consiste na utilização do aplicativo em uma distribuição Ubuntu, versão 10.04, para acessar o equipamento de rede onde funciona o *servidor* www.google.com.br (endereço IP 74.125.229.216). Foi obtido o resultado a seguir:

root@antonio-desktop:~# traceroute 74.125.229.216
traceroute to 74.125.229.216 (74.125.229.216), 30 hops max, 60 *byte* packets
1 200.20.94.58 (200.20.94.58) 0.568 ms 0.484 ms 0.408 ms
2 200.143.254.138 (200.143.254.138) 0.334 ms 0.553 ms 0.481 ms
3 200.143.252.69 (200.143.252.69) 8.382 ms 8.239 ms 8.164 ms
4 187.16.216.55 (187.16.216.55) 9.080 ms 8.941 ms 8.867 ms
5 209.85.243.198 (209.85.243.198) 9.613 ms 9.543 ms 9.470 ms
6 209.85.249.47 (209.85.249.47) 117.573 ms 116.918 ms 158.815 ms
7 216.239.46.94 (216.239.46.94) 117.746 ms 117.674 ms 117.601 ms
8 74.125.229.216 (74.125.229.216) 117.517 ms 116.911 ms 116.816 ms

No segundo exemplo, para acessar ao mesmo equipamento do exemplo, o aplicativo Traceroute será rodado em um sistema operacional Windows 7. A principal diferença está na sintaxe da linha de comando a ser utilizada na execução do aplicativo, onde se utiliza *tracert*, no lugar do *traceroute* utilizado em sistemas Linux. Segue o resultado obtido:

C:\Users\Jéssica&Antonio>tracert www.google.com.br

Rastreando a rota para www.google.com.br [74.125.229.216]
com no máximo 30 saltos:

1 1 ms 1 ms 1 ms 192.168.254.254
2 * * * Esgotado o tempo limite do pedido.
3 38 ms 37 ms 37 ms 200.149.112.30
4 43 ms 41 ms 43 ms 200.199.62.157
5 45 ms 46 ms 46 ms 200.223.55.22
6 62 ms 74 ms 46 ms 72.14.197.21
7 47 ms 45 ms 46 ms 209.85.243.202
8 155 ms 156 ms 156 ms 209.85.249.47
9 157 ms 155 ms 156 ms 216.239.46.94
10 153 ms 153 ms 153 ms mia04s05-in-f24.1e100.net [74.125.229.216]

Rastreamento concluído.

Cabe observar que, em ambos os casos, o máximo de saltos (*hops*) padrão é 30, embora este valor possa ser modificado, caso o usuário o julgue insuficiente para sua demanda. Para sistemas operacionais Linux, também é possível configurar o tamanho dos segmentos UDP enviados, especificando o tamanho desejado, em *bytes*. Por padrão, além do tamanho especificado, serão incluídos mais 8 *bytes* de cabeçalho nos segmentos.

Caso nenhum valor de tamanho seja passado, o aplicativo Traceroute enviará segmentos UDP de 60 *bytes*.

Capítulo 3

Metodologia

3.1 Etapa 1

O objetivo principal desta etapa é implementar um ambiente integrado de monitoramento e gerência dos equipamentos de rede do *backbone* da Redecomep/RedeRio. Será empregado um modelo padrão de gerenciamento de redes composto por:

- **Gerente**: foi adotada uma plataforma WEB baseada no conjunto LAMP: sistema operacional Linux, *servidor* WEB Apache, banco de dados MySQL e linguagem de programação WEB PHP;

- **Base de informações gerenciais (MIB)**: são os objetos, variáveis, parâmetros dos agentes que devem ser tratados durante o processo;

- **Protocolo de gerência (SNMP)**: este protocolo define mensagens a serem trocadas entre o gerente (estação de gerenciamento) e o agente (equipamento de rede);

- **Agente**: *software* específico da função de gerenciamento que o dispositivo a ser gerenciado/monitorado (roteadores, comutadores, *servidores*, etc) possui instalado. Neste caso, baseado no protocolo SNMP.

Dentro do exposto acima, foi feito nesta etapa o desenvolvimento de um sistema de banco de dados com informações administrativas, técnicas e operacionais das diversas instituições filiadas à Redecomep/RedeRio e de todos os equipamentos envolvidos neste *backbone*. Este desenvolvimento foi realizado como implementação prática relacionada à utilização do protocolo SNMP e da base de dados MIB.

Também foi desenvolvido um sistema de monitoramento e gerência de redes para a Redecomep/RedeRio, onde são monitorados todos os equipamentos integrantes desta rede, e suas respectivas interfaces; bem como é feito o controle das instituições que se encontram cadastradas neste sistema, relacionando cada instituição a uma interface de um equipamento de rede.

Este sistema inclui ainda informações numéricas e gráficas de parâmetros relevantes para gerência de equipamentos e de rede, tais como: tráfego em

interfaces, temperatura e taxa de utilização de CPU's de equipamentos, estatísticas de erros, etc.

O aplicativo desenvolvido contempla o seguinte conjunto de funcionalidades:

- Login
- Instituições
  - Detalhe de cada instituição
    - Dados cadastrais
    - Dados técnicos
    - Gráfico de tráfego da interface
      - Gráficos por período
        1. Último Dia
        2. Última Semana
        3. Último Mês
        4. Último Ano
    - Teste de conectividade do endereço IP de rede da instituição
    - Listagem do histórico de ocorrências da instituição
      - Editar ocorrência (se não estiver concluída)
  - Incluir instituição
  - Editar instituição
  - Excluir instituição
  - Incluir ocorrência
  - Lista completa de instituições
  - Lista completa de e-mails
- Equipamentos
  - Detalhe do equipamento
    - Dados do equipamento

        * Ver interfaces do equipamento
            · Detalhes da interface
            · Gráfico de tráfego
                1. Gráficos por período
                    Último Dia
                    Última Semana
                    Último Mês
                    Último Ano
    - Incluir equipamento
    - Editar equipamento
    - Excluir equipamento
- Monitoramento e gerência de rede
    - Utilização das CPU's dos equipamentos
        * Gráficos por período
            · Último Dia
            · Última Semana
            · Último Mês
            · Último Ano
    - Temperaturas dos equipamentos
        * Gráficos por período
            · Último Dia
            · Última Semana
            · Último Mês
            · Último Ano
- Ocorrências pendentes
    - Listagem de ocorrências pendentes, com links para as instituições
- Conjuntos pessoais de gráficos

- Busca
  - Busca geral
  - Busca por Link Permanente (LP)
  - Busca por ocorrência
- Links e Utilitários
  - RedeRIO
  - Telefones úteis
  - Ping
  - Traceroute
  - Sobre a gerência GRRW
- Documentação
- Preferências
  - Conjunto de gráficos
    - Criar conjunto
    - Editar conjunto
    - Excluir conjunto
  - Dados pessoais de usuário
- Logout

Neste sistema, é possível se *log*ar com dois tipos de usuário: admin e visitante. O usuário do tipo admin tem acesso a todas as funcionalidades da aplicação, enquanto que o visitante só tem acesso às telas onde não é possível modificar dados de equipamentos, instituições, ocorrências etc. O sistema está disponível no endereço `http://200.20.94.129/antonio`.

3.2 Etapa 2

Esta etapa, mais complexa e original que a primeira, é relativa a engenharia de tráfego e inclui módulos que contemplem a capacidade de monitoramento e gerência do protocolo Spanning Tree. O objetivo é permitir a fácil observação dos fluxos de dados que são dependentes deste protocolo, possibilitando alterar em tempo real o comportamento destes fluxos através da interação direta na configuração dos equipamentos de rede cadastrados no sistema, onde este protocolo opera.

Nesta etapa, a ideia inicial consiste em simular uma possível configuração de rede a ser encontrada no *backbone* Redecomep. Dentro desta ideia de simulação de rede, o foco principal está em implementar o protocolo Spanning Tree, que, conforme visto na seção 2.2.6, busca obter o caminho com menor custo para se alcançar todos os nós de um dada rede, a partir de um nó convencionado como raiz da rede.

Estas simulações de rede foram realizadas com o programa Cisco Packet Tracer, em sua versão 5.3, que é capaz de simular diversas situações de rede da mesma forma como elas funcionariam numa implementação "física" de uma rede, ou seja, implementando equipamentos de rede (*switches*, roteadores, etc), as conexões entre eles (cabos de par trançado, coaxial, etc), bem como definir as configurações das interfaces dos equipamentos e como estas estarão conectadas entre si.

Em um segundo momento, foi realizada a implementação de uma interface WEB de monitoramento de equipamentos de rede, baseada no protocolo Spanning Tree. Esta interface possui dois objetivos principais:

- O primeiro consiste na apresentação textual dos dados obtidos pelo referido protocolo para cada equipamento (se é um nó raiz da Árvore de Custo Mínimo, seu custo operacional, quais interfaces de rede estão operacionais, etc.);

- O segundo trata da exibição de um mapa de rede, uma imagem apresentando como o protocolo Spanning Tree montou a Árvore de Custo Mínimo de uma dada rede, exibindo como os equipamentos estão conec-

tados uns aos outros, quais interfaces de rede são utilizadas nestas conexões e os custos destas conexões.

3.3 Métodos utilizados

Nesta seção, serão apresentados os ambientes e ferramentas utilizados para a implementação prática dos sistemas implantados neste projeto, bem como a forma como esta implementação foi realizada.

3.3.1 Plataforma LAMP

Para a implementação prática da primeira etapa, foi montada uma plataforma LAMP - acrônimo para os aplicativos Linux, Apache, MySQL e PHP - as tecnologias empregadas neste momento. A distribuição utilizada do Linux é a Ubuntu, versão 9.10; em conjunto com as versões 2.2.12 do Apache, 5.1.37 do MySQL, e 5 do PHP. Este projeto também utilizou o aplicativo PHPMyAdmin, versão 3.2.2.1deb1, que oferece uma interface via *browser* para o usuário administrar o banco MySQL, sendo assim mais amigável e intuitivo para uso, quando comparado com aplicativos convencionais de gerência de bancos de dados.

3.3.1.1 Ubuntu

Os sistemas operacionais baseados na tecnologia Linux são, simultaneamente, um *kernel* (também conhecido como núcleo do sistema operacional) e o sistema operacional propriamente dito a rodar sobre o *kernel*. A tecnologia Linux foi criada em 1991 pelo finlandês Linus Torvalds, e hoje é mantida por uma comunidade mundial de desenvolvedores, indo desde programadores individuais até grandes empresas (tais como IBM, HP e Hitachi), coordenada pela instituição Linux Foundation.

A tecnologia Linux utiliza a licença de *software* livre GPL, permitindo assim que qualquer pessoa interessada possa utilizar ou distribuir esta tecnologia, sem nenhum custo. Em conjunto com outros aplicativos gratuitos, como o KDE, o GNOME, o Apache, o Firefox, os *software*s do sistema GNU e

o OpenOffice.org, é possível se criar um ambiente gráfico similar ao utilizado pelo sistema operacional Windows [70].

O Ubuntu, por sua vez, é um sistema operacional baseado em Linux, portanto de código aberto (qualquer pessoa pode acessar e desenvolver funcionalidades para este) e gratuito (*freeware*).

Uma instalação padrão deve levar menos de 30 minutos. Uma vez que esteja instalado, o sistema operacional Ubuntu já está pronto para uso, possuindo duas versões principais do sistema: a versão *desktop* e a versão servidor. Na versão *desktop*, baseada em interfaces gráficas, existe um conjunto completo de aplicativos básicos previamente determinado, tais como internet, imagens, jogos, etc. Por outro lado, na versão servidor, em modo texto, mas também acessível via terminal na versão *desktop*, é possível configurar o servidor, de modo a instalar aplicativos que venham a ser necessários, modificar configurações do sistema, ou mesmo remover itens desnecessários às demandas do usuário [71].

Em razão da versão 9.10 do Ubuntu ter se tornado obsoleta, não possuindo mais suporte, por parte da comunidade de desenvolvedores Linux, foi necessário atualizar para a versão 10.04 LTS, para o sistema de monitoramento descrito nesta dissertação funcionar adequadamente.

3.3.1.2 MySQL

Em razão do aumento crescente da demanda por armazenamento de dados, incluindo-se nisto a necessidade do compartilhamento destes em ambientes de rede, como a Internet, os desenvolvedores David Axmark, Allan Larsson e Michael "Monty" Widenius deram início, durante os anos 90, à criação de uma interface, baseada na linguagem SQL, para realizar as atividades de manipulação/ gerência de dados. O resultado deste desenvolvimento levou à criação ao aplicativo MySQL.

O aplicativo MySQL é um sistema gerenciador de bancos de dados (SGBD) do tipo relacional, onde os dados armazenados são estruturados em abstrações conhecidas como tabelas, que preservam relacionamentos entre si, onde um ou mais dados de cada uma das tabelas faz referência a outras

tabelas, para obter outros dados destas. A prática de criar tabelas e estabelecer relacionamentos entre elas evita a ocorrência de informações redundantes armazenadas, ocupando desnecessariamente espaço físico em disco [72].

Para efetuar operações próprias de manipulação de dados, tais como consultas, inserções, alterações e remoções de elementos de tabelas, ou mesmo tabelas inteiras, o aplicativo MySQL utiliza a linguagem *Structured Query Language* - SQL (Linguagem de Consulta Estruturada) [72].

O aplicativo MySQL também é caracterizado pelo fato de ser distribuído através de duas licenças: uma licença livre, seguindo o padrão de licenças livres utilizado pela maior parte dos aplicativos que operam em distribuições Linux; e uma licença comercial, com a qual é possível utilizar o aplicativo MySQL em aplicações de tamanho maior, atendendo a grandes sistemas desenvolvidos por empresas [72].

Dentro do sistema operacional Ubuntu, além de se poder utilizar o aplicativo MySQL por intermédio de interfaces WEB, como o aplicativo PHP-MyAdmin, apresentado na seção 3.3.1, MySQL pode também ser utilizado através de linhas de comando, na aplicação Terminal do sistema operacional Ubuntu. Para isto, uma vez que o aplicativo MySQL esteja instalado, deverá ser executado o comando a seguir:

mysql −−user=user_name −−password=your_password db_name

Assim, o usuário poderá acessar um dado banco de dados, bem como executar comandos SQL.

Como exemplos de comandos SQL, pode-se destacar:

- **create table−database nome_da_tabela_ou_banco_de_dados ...**: cria tabelas e/ou bancos de dados no aplicativo MySQL. No caso da criação de tabelas, deverão ser especificados ainda os campos desta tabela, com informações sobre nome e tipo de dados de cada campo;

- **alter table−database nome_da_tabela_ou_banco_de_dados ...**: modifica tabelas e/ou bancos de dados no aplicativo MySQL. No caso da modificação de tabelas, deverão ser especificados ainda os cam-

pos a serem incluídos, modificados ou removidos desta tabela, com informações como nome e tipo de dados de cada campo, em caso de inclusão ou alteração;

- **drop table−database nome_da_tabela_ou_banco_de_dados**: remove tabelas e/ou bancos de dados no aplicativo MySQL, juntamente com qualquer conteúdo que a tabela/banco eventualmente possua;

- **select ... from ...** [**where ...**] [**order by ...**] **...**: consulta um ou mais campos em uma tabela do banco, ou em duas ou mais tabelas unidas através de campos específicos. Dentro destas consultas, pode-se ainda determinar quais registros das tabelas deverão ser exibidos, a partir dos valores que os mesmos possuírem (cláusula *where*), bem como a forma como estes serão ordenados (cláusula *order by*);

- **insert into tabela (registro1, registro2, ...) values (valor1, valor2, ...)**: insere novos registros em uma dada tabela;

- **update tabela set registro1=valor1, registro2=valor2, ...** [**where ...**]: edita um ou mais campos, em um ou mais registros em uma dada tabela, podendo se condicionar tal modificação a uma dada condição (cláusula *where*);

- **delete from tabela** [**where ...**]: apaga um ou mais registros em uma dada tabela, podendo se condicionar tal exclusão a uma dada condição (cláusula *where*).

3.3.1.3 Apache

O *servidor* Apache HTTP é um *servidor* WEB adaptado para dar suporte a aplicações *on-line*, rodando sobre o protocolo HTTP, compatível com a versão 1.1 e posteriores deste protocolo [73]. Seguindo a linha de outros *softwares* livres, o Apache é um aplicativo onde qualquer pessoa interessada em colaborar pode desenvolver correções e/ou melhorias ao mesmo. Também se caracteriza por ser gratuito.

Antes da criação do *servidor* Apche, em Fevereiro de 1995, o *daemon* de domínimo público HTTP desenvolvido por Robert McCool na Universidade de Illinois era o *servidor* WEB mais utilizado em todo o planeta. Contudo, a saída de Robert da universidade em 1994 paralisou o desenvolvimento deste *servidor*, enquanto outros desenvolvedores criavam suas próprias extensões que necessitavam de um aplicativo em comum para unificá-las.

Alguns destes desenvolvedores se uniram para a formação do grupo original de desenvolvimento do atual *servidor* Apache, a unificação mencionada no parágrafo anterior.

As primeiras versões do *servidor* Apache representaram um avanço no objetivo de unificar o projeto de McCool, no entanto, ainda necessitavam de revisão e refinamentos para se tornarem um aplicativo funcional.

Em meados de 1995, a equipe desenvolvedora implementou uma série de funcionalidades para versões 0.7.x do *servidor* Apache, resultando numa nova arquitetura para o *servidor*, conhecida pelo nome de *Shambhala*, incluindo, dentre outras coisas, estrutura modular, interfaces para programação de aplicativos (API's) para mais extensibilidade do aplicativo e algoritmos para alocação dinâmica de memória. Estas implementações, em conjunto com o *servidor* Apache já existente, resultaram no Apache Server versão 0.8.8, disponibilizado em Agosto de 1995.

Posteriormente, após diversos testes, e a inclusão de outras melhorias, em Dezembro do mesmo ano, foi liberada a versão 1 do *servidor* Apache [74].

3.3.1.4 PHP

PHP, acrônimo para *Hypertext Preprocessor - Pré-processador de Hipertexto*, é uma linguagem de programação interpretada, utilizada para desenvolvimento de páginas WEB. A sintaxe PHP é similar a outras linguagens de programação, tais como C, Java e Perl, tendo como principal objetivo permitir a criação de páginas que sejam geradas de forma dinâmica, de acordo com os dados transmitidos para elas. O código PHP é delimitado por tags iniciais e finais <?php e ?>, que permitem ao usuário alternar a escrita de código com outras linguagens, como, por exemplo, HTML.

Diferentemente do que ocorre em linguagens como JavaScript - onde o *cliente* que solicitou o carregamento da página consegue visualizar diretamente o código que gerou a página - em PHP, o código-fonte é executado no *servidor* onde a plataforma PHP está instalada, gerando código HTML que, por sua vez é enviado para o *cliente* responsável pela solicitação do carregamento da página. O *cliente* recebe então os resultados desta execução, no formato HTML, sem visualizar o código PHP original [75].

3.3.1.5 PHPMyAdmin

O aplicativo PHPMyAdmin é uma ferramenta do tipo *freeware* (gratuita) desenvolvida na linguagem PHP, com o propósito de administrar, através de uma interface WEB (*browser*), bancos de dados construídos por meio de aplicativos MySQL, de modo que a tarefa de gerência destas bases de dados possa ser realizada de forma mais amigável e intuitiva, se comparada com outros aplicativos de gerência de bancos de dados.

O aplicativo PHPMyAdmin fornece suporte às principais operações de manipulação de dados e tabelas existentes em aplicativos MySQL, como criação/edição/remoção de tabelas, consulta/criação/edição/remoção de dados em tabelas, etc., operações estas que podem ser realizadas tanto através de comandos SQL, quanto pela própria interface do aplicativo PHPMyAdmin, que permite ao usuário montar sua estrutura de dados sem precisar obrigatoriamente dominar a linguagem SQL [76].

O aplicativo PHPMyAdmin também fornece ferramentas de importação e exportação de dados, desde registros isolados até bancos de dados inteiros, dependendo das necessidades do usuário. Estas atividades podem ser realizadas através de arquivos em formato .CSV, .SQL, etc [76].

Para versões do aplicativo PHPMyAdmin a partir da versão 3, são exigidos, como requisitos de uso, os seguintes aplicativos:

- Versão 5 ou superior do aplicativo MySQL;
- Versão 5.2 ou superior do aplicativo para a linguagem PHP.

O aplicativo PHPMyAdmin começou a ser desenvolvido em 1998 por Tobias Ratschiller, quando este decidiu criar um aplicativo WEB em PHP,

para gerenciamento de bancos de dados MySQL, inspirado no aplicativo MySQL-WEBadmin, de autoria de Peter Kuppelwieser.

Ratschiller, em virtude de falta de tempo para se dedicar a este aplicativo, teve que desistir de dar continuidade ao mesmo. Contudo, o aplicativo PHPMyAdmin havia se tornado um dos mais populares aplicativos de gerência WEB de bancos de dados, com um considerável número de usuários e colaboradores. Para coordenar o desenvolvimento de novas funcionalidades, os desenvolvedores Olivier Müller, Marc Delisle e Loïc Chapeaux registraram o aplicativo PHPMyAdmin no projeto SourceForge.net, assumindo, em 2001, a responsabilidade pelo desenvolvimento desta ferramenta [76].

A figura 3.1 apresenta a interface padrão do aplicativo PHPMyAdmin.

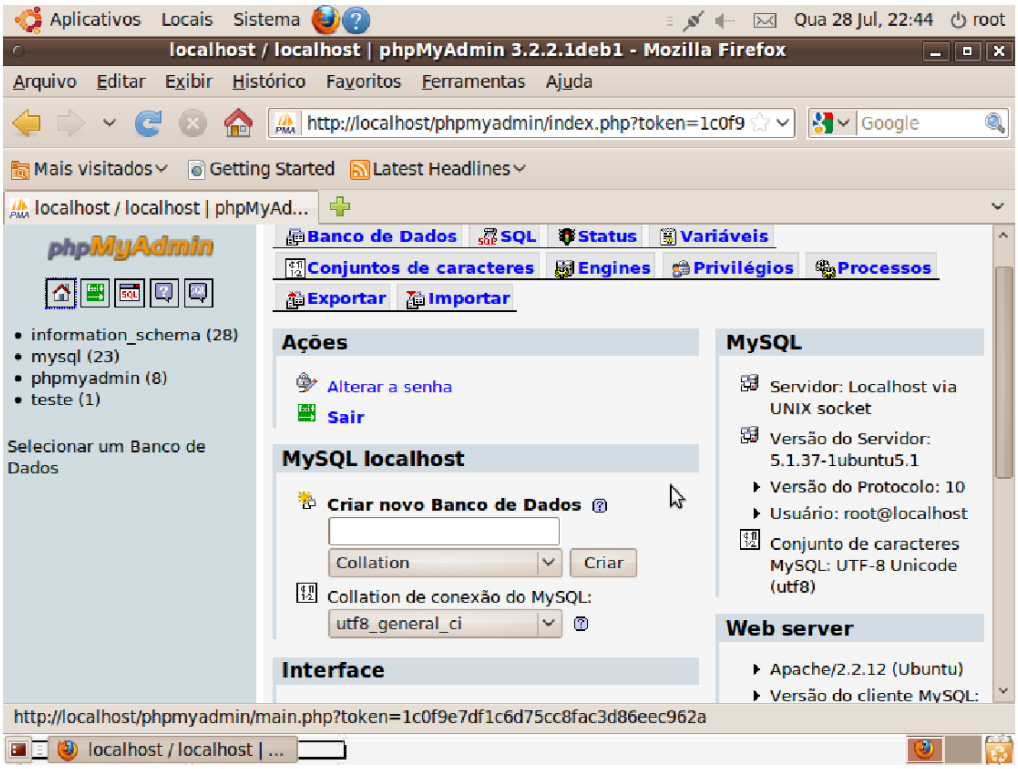


Figura 3.1: Interface do aplicativo PHPMyAdmin

3.3.2 Aplicativos adicionais para etapa 1

Para a etapa 1, também foi necessária a instalação de aplicativos extras, a fim de integrar a plataforma LAMP ao protocolo SNMP, e ainda para a geração dos gráficos de forma mais rápida e eficaz. Os aplicativos instalados foram os seguintes:

- **SNMP**
- **PHP-SNMP**
- **MRTG**
- **RRDTool**

3.3.2.1 SNMP

Pacote de aplicativos disponível para distribuições Linux, com o objetivo de habilitar a utilização, através de prompt de comando, dos aplicativos básicos do protocolo SNMP.

3.3.2.2 PHP-SNMP

Pacote de aplicativos disponível para distribuições Linux, com o objetivo de realizar a intergação entre o protocolo SNMP e a linguagem de programação PHP, habilitando assim o uso de comandos PHP relacionados ao SNMP.

3.3.2.3 MRTG

O aplicativo *Multi Router Traffic Grapher* - MRTG (Gerador de gráficos para tráfego de múltiplos roteadores) é uma ferramenta gratuita de monitoração de entrada e saída de dados diversos em equipamentos de rede, que gera páginas HTML com gráficos de dados coletados via protocolo SNMP (suportado pelo aplicativo MRTG ou por *scripts* externos ao MRTG), e gravados em arquivos de imagens PNG.

O aplicativo MRTG costuma ser mais utilizado para monitorar tráfego de dados em equipamentos de rede, mas pode apresentar qualquer estatística, caso o equipamento de rede forneça os dados da mesma. O aplicativo MRTG pode ser utilizado tanto em sistemas operacionais Windows quanto em distribuições Linux.

Este aplicativo foi desenvolvido por Tobias Oetiker e Dave Rand, utilizando a linguagem Perl para a coleta de dados, e também um módulo escrito na linguagem C para a geração dos gráficos.

Em 1994, Oetiker estava desenvolvendo um site de Internet onde o *link* de rede disponível para o mundo exterior ao *servidor* que hospedava o site possuia largura de banda de somente 64 KBps, o que gerou um interesse sobre como seria a performance deste *link*. Então, Oetiker desenvolveu uma página WEB onde um gráfico de tráfego de rede deste *link* era atualizado periodicamente. Esta atividade envolveu ainda a criação, com a linguagem de programação Perl, de um aplicativo chamado MRTG-1.0, disponibilizado para utilização em meados de 1995.

Um problema observado na primeira versão do aplicativo MRTG era a baixa velocidade de processamento de informações a serem coletadas para o gráfico; sendo descoberto posteriormente que isto era ocasionado pela baixa eficiência de execução do aplicativo, juntamente com o fato do mesmo ter sido totalmente escrito na linguagem Perl.

Como primeira tentativa de solucionar o problema, implementaram-se algumas mudanças no código-fonte original, de forma a otimizar sua velocidade de processamento, sem sucesso. Como segunda tentativa, Dave Rand decidiu reescrever as partes mais críticas, em termos de tempo de processamento, do código MRTG na linguagem C, o que acarretou em um aplicativo aproximadamente 40 vezes mais rápido que o original, o que levaria posteriormente à criação da versão 2 do aplicativo MRTG.

Após o desenvolvimento do aplicativo MRTG-2, Oetiker e Rand começaram a disponibilizar cópias em versão beta do aplicativo, com aplicativos adicionais para correção de erros detectados pelos usuários. Para esta dissertação, está sendo utilizada a versão 2.9.17 do aplicativo MRTG.

O aplicativo MRTG faz a coleta de dados de uma dada estatística sempre

medindo 2 valores desta. Por exemplo, em se tratando de um caso de tráfego de rede, estes valores podem corresponder às taxas de tráfego de entrada e saída de uma interface de rede do equipamento monitorado, usualmente medidas em *bits* por segundo (Bps), ou em *megabits* por segundo (MBps).

A leitura dos dados é feita através do protocolo SNMP, onde o mesmo buscará o dado através do identificador OID associado à estatística na base MIB do equipamento monitorado; ou através de algum *script* que retorne um formato padrão para o dado coletado. Por padrão, o aplicativo MRTG coleta, a cada 5 minutos, os dados de monitoramento junto ao equipamento monitorado, contudo, é possível modificar este intervalo de tempo no arquivo de configuração (.CFG) do MRTG, lembrando que o intervalo de tempo no arquivo .CFG é medido em segundos.

Um ponto relevante do aplicativo MRTG é o fato de que este guarda os dados coletados em arquivos de *log*, utilizados para a geração dos gráficos finais, e que estes arquivos nunca crescem mais do que o limite estabelecido para a correta geração dos gráficos, evitando assim que os arquivos de *log* ocupem um espaço demasiadamente grande no *servidor* onde o MRTG estiver instalado.

Esta ferramenta de controle de tamanho de arquivos é feita através da implementação, no aplicativo MRTG, de um algoritmo de consolidação de dados, que preserva nos arquivos de *log* apenas as informações consideradas mais significativas nos últimos dois anos de coleta de dados de uma dada interface/equipamento de rede.

Paralelamente à coleta de dados, o MRTG atualiza, no mesmo intervalo de tempo utilizado para a monitoração, uma página HTML com 4 gráficos de monitoramento da estatística desejada para cada interface do equipamento monitorado, ou do próprio equipamento, se for o caso (gráficos diário, semanal, mensal e anual), podendo suprimir algum deles, caso não seja necessário. Esta modificação deverá ser feita no arquivo de configuração (arquivo .CFG).

A figura 3.2 apresenta um exemplo de página WEB gerada pelo aplicativo MRTG, com alguns gráficos de monitoramento de tráfego de dados através de algumas interfaces de rede. Os gráficos na cor verde representam as taxas

de entrada de dados nas interfaces de rede do equipamento monitorado; já os gráficos na cor azul, representam as taxas de saída de dados nas interfaces de rede deste mesmo equipamento.

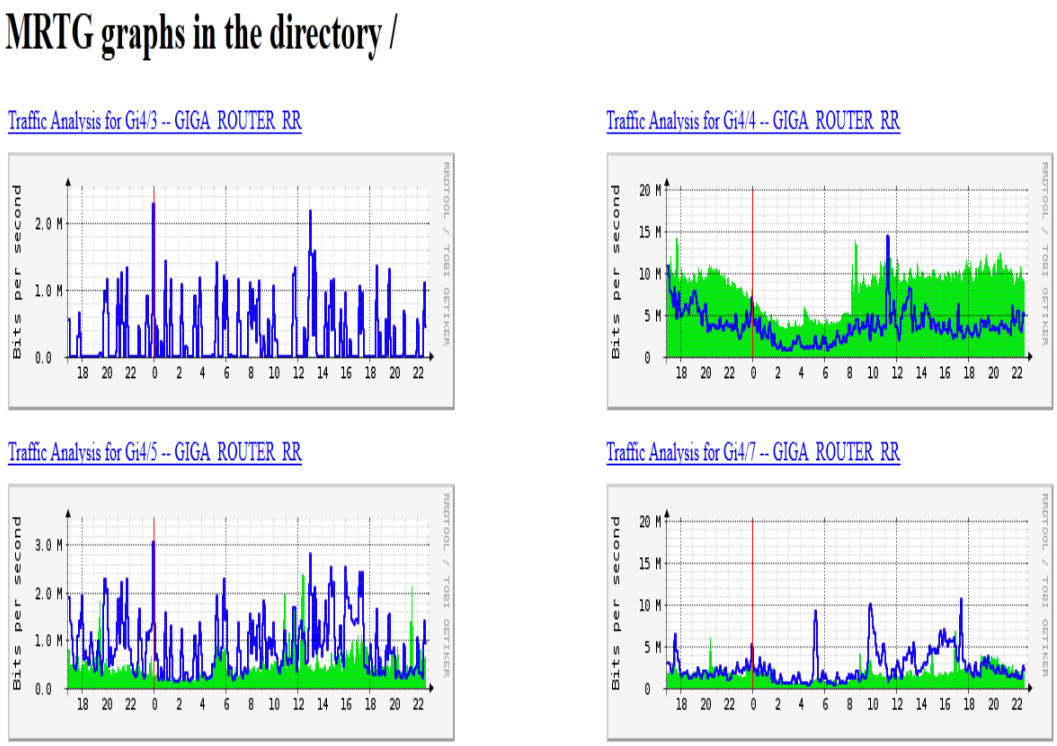


Figura 3.2: Página de gráficos gerada pelo aplicativo MRTG

O arquivo .CFG, responsável pela configuração do aplicativo MRTG, é gerado pela ferramenta *cfgmaker*, nativa da instalação padrão do MRTG. Mas, conforme observado nos parágrafos anteriores, este arquivo também pode ser editado manualmente pelo usuário [7].

3.3.2.4 RRDTool

O aplicativo RRDTool consiste em um banco de dados baseado no algoritmo Round-Robin de execução de processos computacionais. Assim como o aplicativo MRTG, este aplicativo também foi criado por Tobias Oetiker sob licença GNU GPL, o que o caracteriza como um software livre.

O algoritmo Round-Robin é um dos mais antigos algoritmos de escalonamento existentes. É muito utilizado, e foi projetado para sistemas do tipo *time-sharing*, onde o sistema operacional otimiza o tempo ocioso de execução de um processo, alocando este tempo para outro processo.

A idéia do algoritmo consiste em definir certa unidade de tempo, chamada de *timeslice* ou *quantum*. Todos os processos pendentes são então armazenados em uma fila circular. O escalonador da CPU percorre esta fila, reservando a utilização da CPU para cada processo durante um *quantum*. Mais precisamente, o escalonador obtém o primeiro processo da fila e procede à sua execução. Se um *quantum* for insuficiente para o processo ter sua execução finalizada, este é interrompido e inserido no fim da fila. Do contrário, a CPU é liberada para que o próximo processo seja executado. Em ambos os casos, após a liberação da CPU, um novo processo é escolhido na fila. Novos processos do sistema são inseridos no fim da fila, após o último processo interrompido pelo algoritmo Round-Robin.

Quando um processo é retirado da fila para a CPU, ocorre um evento chamado troca de contexto, aumentando o tempo prático de execução deste processo.

O tempo total de execução de um dado processo via Round-Robin é estimado através da prática conhecida como *polling*.

Suponhamos que existam n processos numa fila de tarefas a cumprir, e que o *quantum* entre duas interrupções de execuções de processos é q. No melhor caso, cada processo utilizaria $1/n$ do tempo total de um ciclo da CPU, ciclo este dividido em n partes de q unidades de tempo cada, e esperaria no máximo $n*q$ unidades de tempo para voltar à CPU. Um cálculo mais próximo da realidade para este tempo de espera seria $n(q + o)$, onde o é tempo da troca de contexto entre o fim de um *quantum* e o início do *quantum* seguinte (tempo de duração de uma interrupção, na prática). O tempo da troca de contexto deve ser pequeno, em relação ao *quantum*.

A figura 3.3 apresenta um exemplo simples do funcionamento do algoritmo Round-Robin, onde a CPU inicialmente executa, durante um certo *quantum*, um dado processo P1 (a), passando posteriormente a executar um processo P2 (b), e assim sucessivamente.

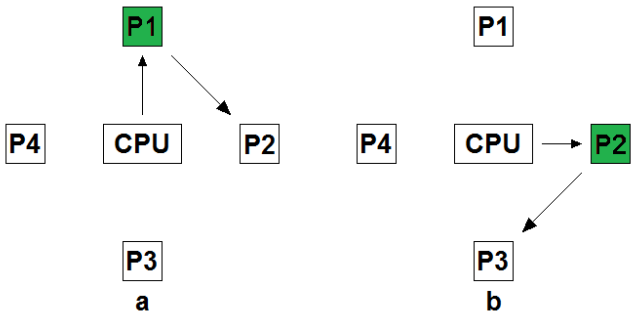


Figura 3.3: Funcionamento do algoritmo Round-Robin

O desempenho do algoritmo Round-Robin está condicionado ao *quantum* destinado para cada execução individual de processo. Se este for muito grande, o algoritmo Round-Robin adota um comportamento similar ao do algoritmo *First In First Out* - FIFO, onde todos os processos são executados em ordem sequencial, e um dado processo só pode ser executado quando todos os que estão à sua frente tiverem sido concluídos (no caso do Round-Robin todos, ou pelo menos a maior parte dos processos serão executados completamente em um único *quantum*).

Por outro lado, se o *quantum* for muito pequeno, o algoritmo passa a se comportar como um algortimo do tipo *processor sharing*, onde o tráfego é visto como se fosse fluido, de tal modo que todos os processos seriam executados de modo quase simultâneo, o que poderia fazer com que o tempo de execução de processos aumentasse consideravelmente, somando-se a isto o significativo aumento do valor de o, em razão da quantidade elevada de trocas de contexto no escalonamento de processos.

Com isto, é importante estimar o tempo médio que um dado processo levará para ser finalizado. Para tal, deve ser calculado o Tempo Médio de Finalização, obtido por:

$$(\sum_{i=1}^{N} T_i)/N$$

Onde T_i é o tempo necessário para finalizar o i-ésimo dos N processos finalizados pelo sistema operacional [77].

Com isto, é possível se estimar quanto tempo a CPU do *servidor* está levando para executar cada processo, permitindo assim que se determine um *quantum* ótimo, para que o maior número possível de processos sejam finalizados com maior rapidez. Este *quantum* deve ser suficientemente grande, para que o tempo gasto em trocas de contexto não comprometa o tempo de execução; e suficientemente reduzido, para que a utilização da CPU não fique restrita a poucos processos.

O aplicativo RRDTool foi desenvolvido para armazenar séries de dados numéricos sobre estatísticas de tráfego de dados de equipamentos de rede, podendo também ser utilizado no armazenamento de qualquer outro tipo de dados, tais como temperatura, utilização de CPU's, etc., assim como ocorre com o aplicativo MRTG. RRD é um acrônimo para *Round-Robin Database* (base de dados round-robin).

A aquisição de dados via algoritmo Round-Robin no aplicativo RRDTool consiste no armazenamento de dados em arquivos de *log*, no formato .RRD, assim como ocorre no aplicativo MRTG. Isto funciona da seguinte forma: se, por exemplo, for necessário armazenar 1000 valores de uma dada medida, coletados em intervalos de 5 minutos entre cada coleta, o aplicativo RRDTool ira reservar espaço para estes valores, e criará um cabeçalho especificando o intervalo de tempo representado por este conjunto de valores. À medida em que novos valores sejam obtidos para esta medida, o histórico do aplicativo restringe o conjunto de valores "visíveis" aos 1000 mais recentes, evitando assim que o aumento dos dados ocupe mais espaço do que o que foi previamente reservado para este histórico.

Caso seja preciso se aumentar o tamanho do histórico, o aplicativo RRDTool deverá ser reconfigurado para o novo tamanho desejado.

Assim como acontece no aplicativo MRTG, o aplicativo RRDTool também é capaz de gerar gráficos dos dados armazenados durante a monitoração de equipamentos de rede. O aplicativo RRDTool possui ainda funções de consolidação, para informar dados de interesse do gráfico a ser monitorado (valor médio da estatística em um dado período de monitoramento, mínimo, máximo, etc.), bem como os intervalos de monitoramento do gráfico (último dia, última semana, último mês, último ano) [78].

O aplicativo RRDTool fornece ainda ao usuário uma funcionalidade para detectar possíveis comportamentos anormais em um dado monitoramento. Isto inclui [79]:

- Um algoritmo que estima possíveis valores futuros para uma série de dados qualquer, baseado no histórico de dados da mesma. Este algoritmo acumula os dados obtidos, formando três componentes, cuja soma é utilizada para a predição dos próximos dados:
  - Um valor básico esperado para cada elemento do conjunto de dados;
  - Um fator linear de crescimento estimado para os próximos valores a serem obtidos;
  - Um parâmetro relacionado à sazonalidade, ou seja, ao tamanho da discrepância entre os valores do conjunto de dados dentro de um dado período de tempo.

  O parâmetro da sazonalidade pode funcionar de forma aditiva ou multiplicativa. Na forma aditiva, a variação sazonal de valores se mantém constante ao longo do tempo; já na forma multiplicativa, esta variação cresce com o passar do tempo. Esta metodologia de predição de valores é conhecida como Modelo de Holt-Winters.

  À medida em que é feita a coleta de dados, cada um dos três componentes é recalculado através de suavização exponencial, onde então o algoritmo "aprende", com os dados coletados anteriormente, a forma provável que os mesmos devem se comportar em futuras aquisições de dados, fazendo assim a predição, conforme proposto pelo Modelo de Holt-Winters;

- Uma medida do desvio entre os dados estimados e os dados coletados. Esta medida é obtida através da utilização dos dados obtidos anteriormente, atribuindo pesos aos seus valores, de tal forma que os valores mais recentes têm maior peso no cálculo do desvio. A partir do histórico de dados e as predições originalmente estimadas para os

mesmos, são geradas faixas de confiança, que funcionam como uma medida de erro "aceitável" para as predições futuras, controlando assim se a discrepância entre um dado valor obtido e a predição proposta inicialmente para ele está acima do que pode ser considerado aceitável;

- Um mecanismo de decisão que define se um valor, ou conjunto de valores está ou não discrepante em relação ao esperado; o que configura ou não a existência de comportamento anormal em um monitoramento. O aplicativo RRDTool reporta como comportamento anormal a incidência de dados obtidos que excedem as faixas de confiança, ou ainda quando ocorre uma determinada quantidade de valores excessivamente discrepantes dentro de um determinado período de tempo.

Algumas configurações importantes a serem customizadas no arquivo .CFG são:

- **Options[_]: growright, *bits***: Definem, respectivamente, as opções de direcionamento de como o gráfico será desenhado e a unidade de medida a ser utilizada para definir como os dados serão colocados no gráfico. Neste exemplo, o gráfico será desenhado da esquerda para a direita, e os valores coletados serão contabilizados em *bits*;

- **Colours[_]: GREEN#00eb0c,BLUE#1000ff, DARK GREEN #006600,VIOLET#ff00ff**: Define as cores que serão utilizadas para cada estatística exibida no gráfico. Neste exemplo, a primeira estatística aparecerá na cor verde, a segunda na cor azul, e assim sucessivamente;

- **YLegend[_]: Bits per second**: Define o texto a ser exibido como legenda no eixo Y;

- **LegendI[_]: In**: Define a legenda correspondente à primeira estatística definida para aparecer no gráfico;

- **LegendO[_]: Out**: Define a legenda correspondente à segunda estatística definida para aparecer no gráfico;

- **WorkDir: /var/www/mrtg**: Define o diretório onde serão salvos os arquivos contendo os dados coletados (*logs* em formato .RRD, gráficos, etc.);

- **RunAsDaemon: Yes**: Define se o aplicativo MRTG buscará periodicamente os dados dos equipamentos monitorados, para atualização dos arquivos de *log* (.RRD) e dos gráficos de monitoramento. Neste exemplo, o aplicativo está configurado para fazer esta busca periódica;

- **LogFormat: rrdtool**: Define como o aplicativo MRTG irá armazenar os dados coletados. Neste exemplo, isto será feito através do aplicativo RRDTool, gerando arquivos .RRD;

- **Refresh: 300**: Define o tempo, em segundos, que o aplicativo MRTG levará para atualizar os gráficos de monitoramento na página WEB em que estiverem sendo exibidos;

- **Interval: 1**: Define o tempo, em minutos, que o aplicativo MRTG levará para buscar novas estatísticas dos equipamentos monitorados, contando a partir da última coleta de dados.

3.3.3 PHP-GD

A linguagem PHP não cria apenas códigos em linguagem HTML como saída de dados. Através da código PHP, também é possível criar e manipular arquivos de imagem em diversos formatos diferentes, tais como GIF, PNG, JPG, etc.

Para isto, é necessário utilizar a biblioteca de funções GD, que oferece uma série de rotinas prontas para uso, de modo a gerar dimensionamento de imagens, linhas, textos, formas geométricas diversas (preenchidas ou apenas seus respectivos contornos), bem como pode se editar cores diversas para todos estes elementos listados anteriormente [80].

A figura 3.4 apresenta um exemplo do funcionamento da biblioteca GD, para geração de imagens.

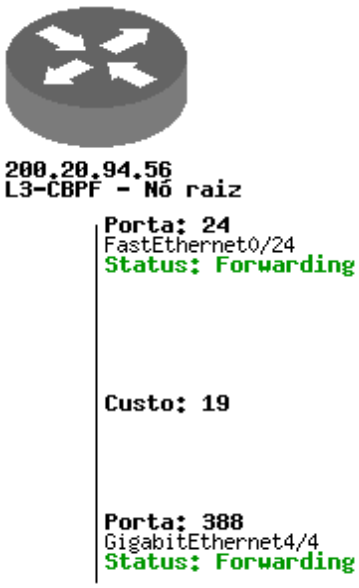


Figura 3.4: Geração de imagens com a biblioteca GD

No exemplo da imagem 3.4, inicialmente, para se criar a imagem, será utilizada a rotina **imagecreate(largura,altura)**, para se determinar as dimenões da imagem gerada, através dos argumentos de largura e altura passados na chamada da rotina.

Para definir as cores de elementos da imagem, utiliza-se a rotina **imagecolorallocate(imagem, red, green, blue)**, onde os argumentos serão a imagem onde tais cores serão empregadas; e, em seguida a cor a ser utilizada, que será definida por três parâmetros numéricos, correspondentes, respectivamente, às escalas de vermelho, verde e azul da cor. Cada um destes parâmetros de cor pode variar de 0 a 255, onde, quanto maior o valor, mais intensa é a tonalidade de cor. Por exemplo, parâmetros 0, 0 e 0 gerarão ausência de cor (cor preta), enquanto parâmetros 255, 255 e 255 gerarão intensidade máxima de verde, vermelho e azul (cor branca).

Para se criar as formas geométricas a serem utilizadas na imagem, serão utilizadas as seguintes rotinas:

- **imagefilledrectangle(imagem, posicao_x_inicial, posicao_y_inicial, posicao_x_final, posicao_y_final, cor)**: Responsável por preen-

cher um retângulo na imagem definida pelo parâmetro imagem, com a cor definida no parâmetro cor, iniciando o preenchimento com os parâmetros posicao_x_inicial e posicao_y_inicial definindo a localização, na figura, do ponto superior esquerdo do retângulo; e os parâmetros posicao_x_final e posicao_y_final definindo a localização, na figura, do ponto inferior direito do retângulo. A rotina **imagerectangle** funciona de forma similar à rotina **imagefilledrectangle**, se diferenciando desta pelo fato de preencher apenas o contorno do retângulo;

- **imagefilledellipse(imagem, posicao_x, posicao_y, largura, altura, cor)**: Responsável por preencher uma elipse na imagem definida pelo parâmetro imagem, com a cor definida no parâmetro cor, determinando o centro da elipse com os parâmetros posicao_x e posicao_y, determinando a largura e a altura da elipse com os parâmetros largura e altura. A rotina **imageellipse** funciona de forma similar à rotina **imagefilledellipse**, se diferenciando desta pelo fato de preencher apenas o contorno da elipse;

- **imagefilledpolygon(imagem, array(posicao_x_ponto1, posicao_y_ponto1, posicao_x_ponto2, posicao_y_ponto2, ...,posicao_x_pontoN, posicao_y_pontoN, N, color)**: Responsável por preencher um polígono na imagem definida pelo parâmetro imagem, com a cor definida no parâmetro cor, determinando cada um dos N pontos do polígono com os parâmetros posicao_y_ponto1, posicao_x_ponto2, posicao_y_ponto2, ..., posicao_x_pontoN, posicao_y_pontoN. A rotina **imagepolygon** funciona de forma similar à rotina **imagefilledpolygon**, se diferenciando desta pelo fato de preencher apenas o contorno do polígono.

Para se inserir textos na imagem, é utilizada a rotina **imagestring(imagem, tamanho_fonte, posicao_x, posicao_y, texto, cor)**, onde o texto especificado no parâmetro texto será inserido na imagem definida pelo parâmetro imagem, com a cor definida no parâmetro cor, com o tamanho definido pelo parâmetro tamanho_fonte, e a posição superior esquerda definida pelos parâmetros posicao_x e posicao_y.

Para se inserir linhas na imagem, é utilizada a rotina **imageline(imagem, posicao_x_inicial, posicao_y_inicial, posicao_x_final, posicao_y_final, cor)**, onde uma linha é inserida na imagem definida pelo parâmetro imagem, com a cor definida no parâmetro cor, ligando o ponto definido pelos parâmetros posicao_x_inicial e posicao_y_inicial ao ponto definido pelos parâmetros posicao_x_final e posicao_y_final.

Para se efetuar a digitalização da imagem, utiliza-se a rotina **imagepng(imagem)**, passando como parâmetro a imagem a ser digitalizada.

Por fim, para se liberar o espaço de memória utilizado para a criação de uma imagem, utiliza-se a rotina **imagedestroy(imagem)**, passando como parâmetro a imagem digitalizada.

3.3.4 Packet Tracer

O aplicativo Packet Tracer, desenvolvido pela empresa Cisco, é um *software* destinado à simulação de redes computacionais, de modo a perimitir a reprodução de possíveis situações e comportamentos de rede. Este aplicativo oferece versões virtuais de PC's, roteadores, *switches*, diversos tipos de cabos de rede, etc.

Também disponibiliza a implementação de comandos de aplicativo *Internetwork Operating System* - IOS (utilizado para configurar equipamentos de rede), para a configuração da rede a ser simulada; bem como uma interface para a visualização do tráfego de dados, de modo a se saber como os pacotes estão sendo enviados de um ponto a outro, e os status dos mesmos.

É possível utilizar este aplicativo tanto em sistemas operacionais Windows, quanto em sistemas baseados em Linux (Ubuntu, Fedora, etc.).

O aplicativo Packet Tracer oferece também suporte para simulação de diversos protocolos, viabilizando assim o estudo de comportamentos destes em uma dada rede simulada. Alguns exemplos de protocolos suportados pelo Packet Tracer são [81]:

- **HTTP**
- **TCP**

- **UDP**
- **IP**
- **ICMP**
- **STP**

A figura 3.5 apresenta a tela inicial do aplicativo Packet Tracer.

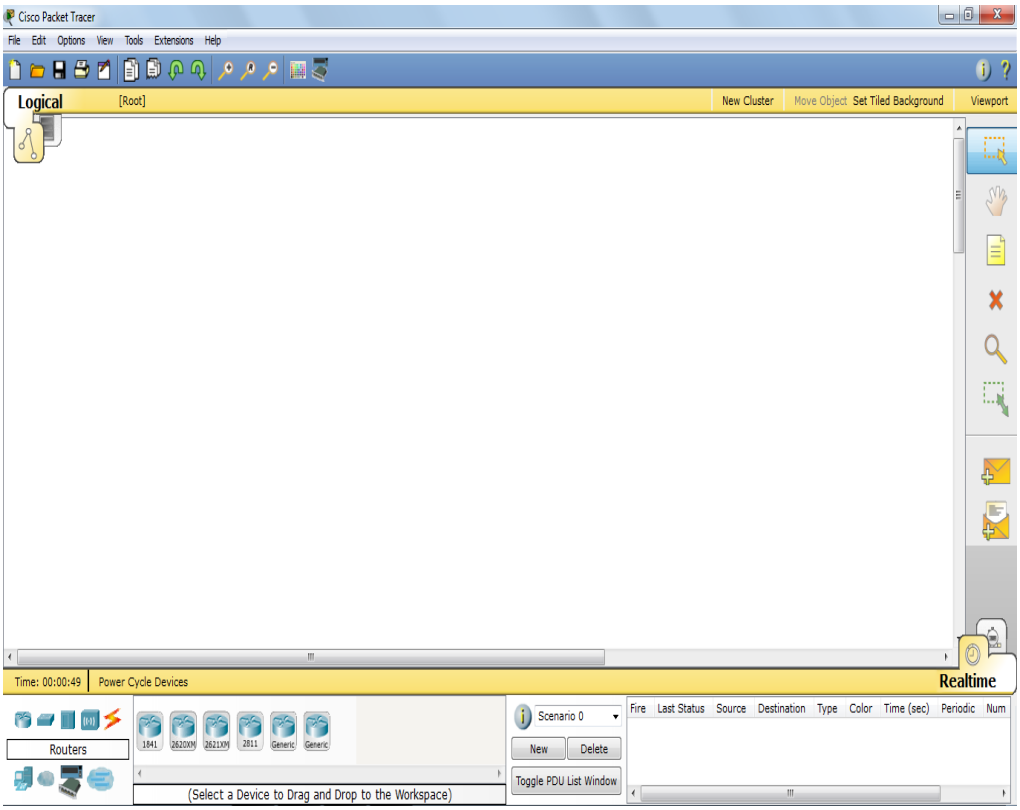


Figura 3.5: Tela inicial do aplicativo Packet Tracer

Conforme mencionado na seção 3.2, a versão utilizada do Packet Tracer é a 5.3. Como este aplicativo foi utilizado apenas para fins de simulação, o mesmo não foi instalado no *servidor* LAMP utilizado para o sistema final desenvolvido como objeto desta dissertação.

3.4 Implementação

Nesta seção, será apresentada a forma como as funcionalidades envolvidas neste projeto foram implementadas. Por motivos de segurança, alguns dados das telas que serão apresentadas aqui foram omitidos neste trabalho.

3.4.1 Etapa 1

Na etapa 1, conforme visto na seção 3.1, foi desenvolvido um aplicativo WEB baseado na tecnologia PHP com o conjunto de funcionalidades lá descrito. Este aplicativo conta ainda com algumas funções desenvolvidas em JavaScript, auxiliando o sistema em ocasiões de validação de dados a serem inseridos, editados ou excluídos do mesmo.

Ainda conforme mencionado na seção 3.1, existem dois níveis de acesso a este sistema: **admin** e **visitante**, onde o primeiro nível consegue modificar informações contidas no sistema, enquanto que o segundo tem permissão para somente consultar informações.

Dentro deste conjunto de funcionalidades, foram implementados os seguintes itens[1]:

- **Login**: Tela onde o usuário deverá inserir o *login* e a senha com os quais entrará no sistema;
- **Instituições**: Tela onde é exibida a listagem de instituições cadastradas no sistema, os dados delas (cadastrais e técnicos) e onde é possível inserir, editar ou excluir instituições do sistema. A figura 3.6 apresenta um exemplo de instituições listadas dentro do sistema;
  - **Detalhe de cada instituição**: Tela onde são exibidos os dados cadastrais e técnicos, juntamente com o gráfico de monitoramento de tráfego da interface utilizada pela instituição, caso hajam dados

[1]Os códigos-fonte do sistema implementado e descrito nesta dissertação não foram anexados neste documento, em razão do número excessivo de páginas que tal anexo geraria. No entanto, o mesmo estará disponível para quaisquer interessados em acessá-lo, bastando entrar em contato com o autor desta dissertação, a fim de se obtê-los.

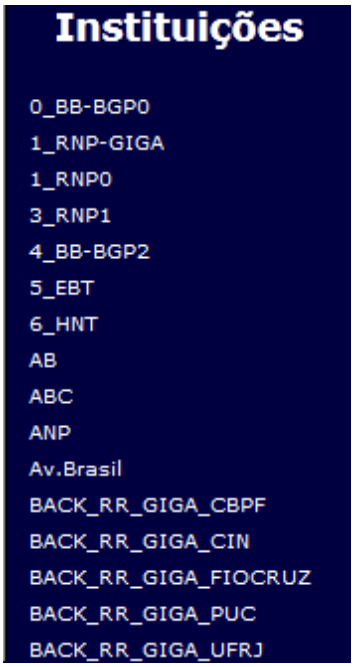


Figura 3.6: Exemplo de listagem de instituições cadastradas

a serem exibidos no gráfico, e o histórico de ocorrências relacionado à instituição, se houverem ocorrências registradas;

* **Dados cadastrais**: Relação de informações básicas sobre a instituição cadastrada. São elas:
  - **Sigla**: Sigla que identifica a instituição na listagem de instituições;
  - **Nome da instituição**: Nome completo da instituição cadastrada;
  - **Status**: Status da instituição no sitema. Pode ser ativa ou inativa;
  - **Endereço**: Endereço da instituição;
  - **Página Web**: Site oficial da instituição;
  - **Nome(s) de contato**: Listagem de pessoas da instituição que podem ser contactadas pelo sistema;
  - **Telefone**: Listagem de telefones de contato da instituição;
  - **E-mail**: Listagem de e-mails de contato da instituição.

A figura 3.7 apresenta um exemplo de como ficam os dados cadastrais de uma dada instituição;

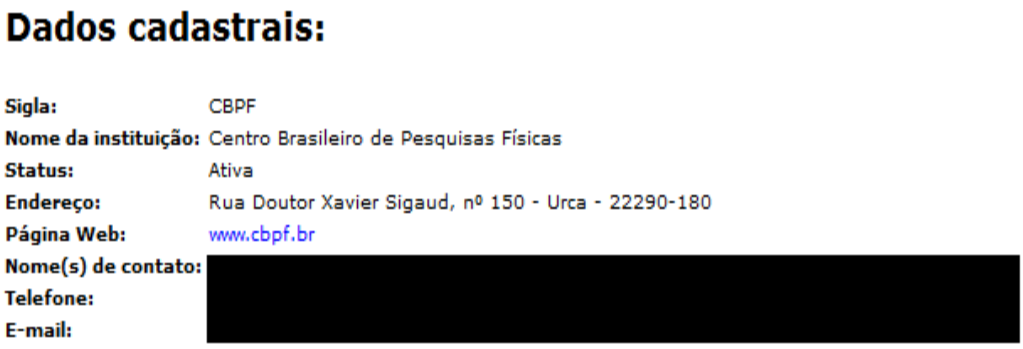


Figura 3.7: Listagem de dados cadastrais de instituição

* **Dados técnicos**: Relação de informações técnicas sobre a instituição cadastrada. São elas:
  - **Link/LP**: *Link* permanente de banda de conexão da instituição;
  - **Roteador**: Roteador utilizado pela instituição;
  - **IP do roteador**: Endereço IP do roteador utilizado pela instituição;
  - **Equipamento**: Equipamento utilizado pela instituição;
  - **IP do equipamento**: Endereço IP do equipamento utilizado pela instituição;
  - **Interface**: Número da interface de equipamento utilizada pela instituição;
  - **Status**: Status operacional da interface. Pode ser "*up*" (interface ativa), ou "*down*" (interface inativa);
  - **Tipo**: Nome da interface utilizada;
  - **IP da rede**: Endereço IP da rede utilizada pela interface;
  - **IP interno**: Endereço IP interno da interface.

  A figura 3.8 apresenta um exemplo de como ficam os dados técnicos de uma dada instituição;

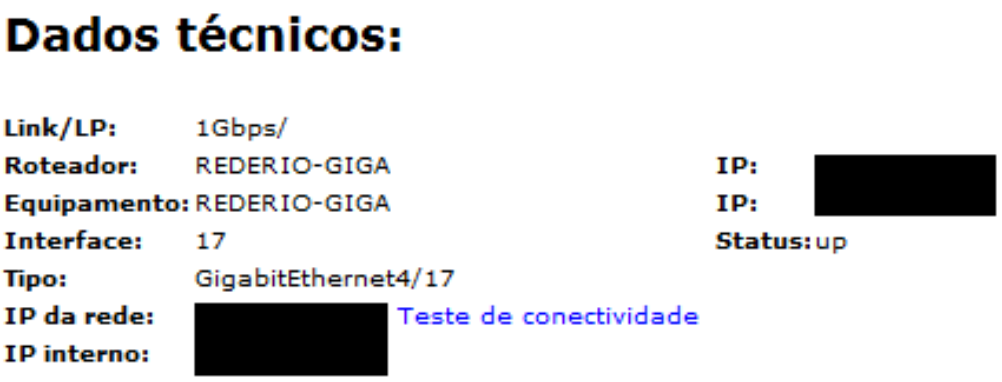


Figura 3.8: Listagem de dados técnicos de instituição

* **Gráfico de tráfego da interface**: Gráfico de monitoramento de tráfego de entrada e saída de *bytes* (octetos) da interface, medido em *bits* por segundo - Bps. A escala de monitoramento do aplicativo MRTG é adaptada à ordem de grandeza do tráfego, de acordo com o valor máximo atingido por este, ou seja, pode ser mensurada em *bits*, *Kilobits*, *Megabits* ou *Gigabits* por segundo. O gráfico padrão apresenta sempre o monitoramento nas últimas 24 horas (gráfico diário);
  - **Gráficos por período**: Ao se clicar no gráfico principal, é aberta uma nova janela que exibe os gráficos de monitoramento de tráfego da interface em quatro períodos distintos. São eles:
    1. **Último Dia**
    2. **Última Semana**
    3. **Último Mês**
    4. **Último Ano**

    A figura 3.9 apresenta um exemplo de gráficos de monitoramento de tráfego. Neste exemplo, podemos ver os gráficos diário e semanal de uma interface.
* **Teste de conectividade do endereço de rede da instituição**: Ferramenta para teste de conectividade básica com

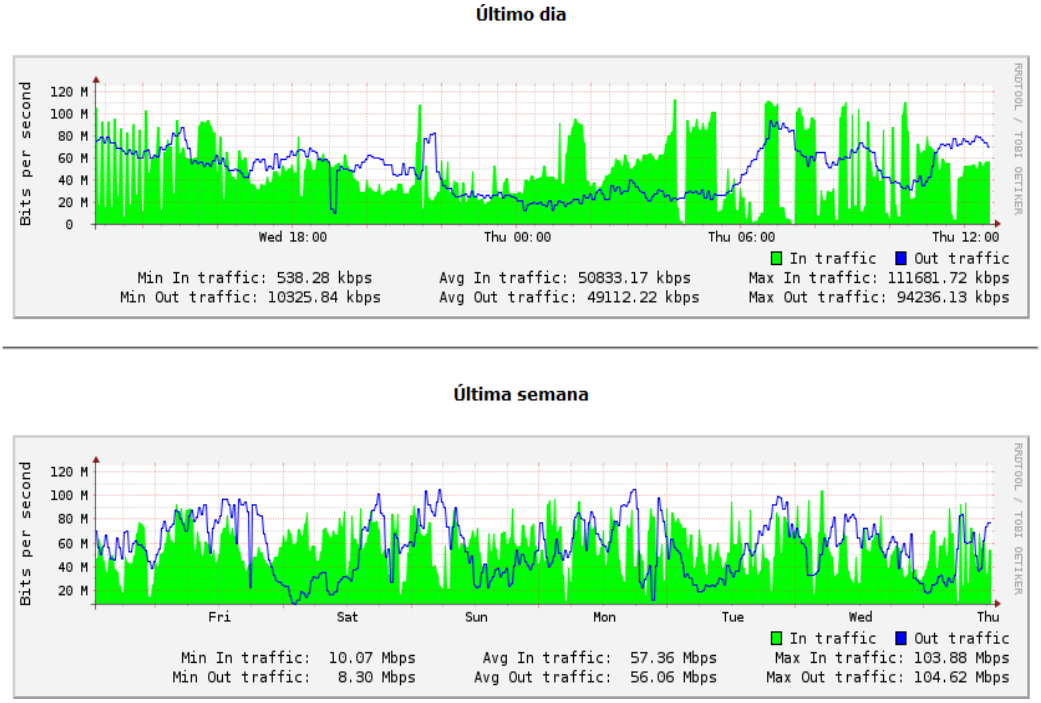


Figura 3.9: Exemplos de gráficos de monitoramento de tráfego

os endereços IP's interno e externo, relacionados ao endereço de rede utilizado pela interface, através do aplicativo PING. Nesta interface WEB, o usuário pode configurar a quantidade de *pings* a serem enviados, e o tamanho, em *bytes*, dos pacotes "pingados". Na página de resultados, aparecerão os totais de *pings* efetuados com sucesso e com falha, e os tempos de resposta mínimo, máximo e médio, em milisegundos. Caso o usuário não configure os valores de total de *pings* e tamanho de pacotes, o sistema adotará os valores padrão de 3 *pings* e 24 *bytes* para tamanho de pacote.

As figuras 3.10 e 3.11 apresentam o funcionamento do teste de conectividade;

* **Listagem do histórico de ocorrências da instituição**: Tela onde é exibida a listagem de ocorrências cadastradas em uma instituição, ordenando-as por dia e hora, iniciando pela ocorrência mais recente, até a mais antiga. Nesta tela, são

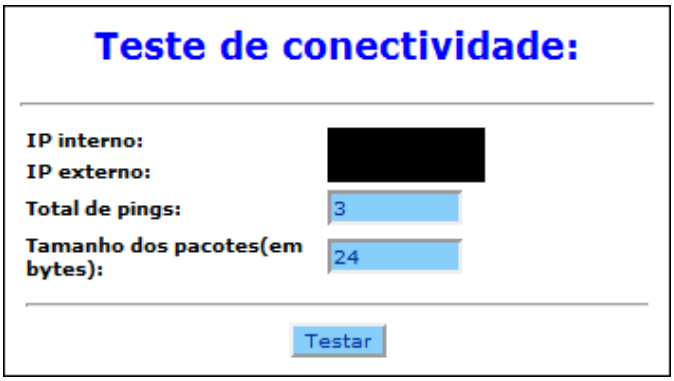


Figura 3.10: Interface de entrada do teste de conectividade

exibidas, além de dia e hora da ocorrência, a descrição da ocorrência e o status atual da mesma, que pode ser "Pendente" ou "Concluída". A figura 3.12 apresenta um exemplo de como ocorrências são listadas no sistema.

- **Editar ocorrência (se não estiver concluída)**: Caso o status da ocorrência não esteja como concluído, é possível realizar modificações na descrição nesta ocorrência em especial.
  A figura 3.13 apresenta a tela de edição de ocorrências de uma instituição no sistema.
  Nesta tela, os campos Data e Descrição são de preenchimento obrigatório. Caso não sejam preenchidos, será apresentada a mensagem abaixo, ilustrada pela figura 3.14.

– **Incluir instituição**: Conjunto com duas telas, para a inclusão de uma instituição. Na primeira tela, são cadastrados todos os dados cadastrais e técnicos, exceto o dado de tipo de interface, que será cadastrado na tela seguinte, uma vez que tenha sido escolhido o equipamento com quem a instituição será vinculada.

  Na segunda tela, caso seja selecionado algum equipamento de rede na tela anterior, serão então carregadas numa *combobox* todas as interfaces do equipamento disponíveis para uso, num trabalho conjunto entre o aplicativo *snmpwalk*, listando as interfaces; e a

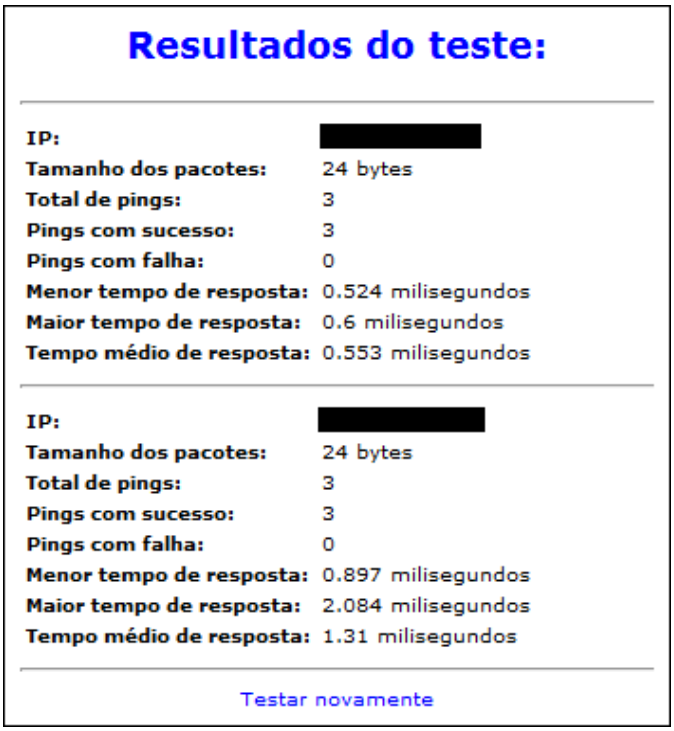


Figura 3.11: Interface de saída do teste de conectividade

base de dados, responsável pela restrição às interfaces já utilizadas para alguma outra instituição.

Caso nenhum equipamento seja selecionado na primeira tela, serão exibidos dois campos de texto, de preenchimento opcional: um para o número, e outro para o nome da interface.

As figuras 3.15, 3.16 e 3.17 apresentam as telas de inclusão de instituições. A figura 3.16 representa a segunda tela, quando nenhum equipamento é selecionado na primeira tela de inclusão de instituições; já a figura 3.17, representa a segunda tela quando há algum equipamento selecionado.

As figuras 3.18 e 3.19 apresentam as mensagens que indicam a relação de campos de preenchimento obrigatório na inclusão de instituições. Lembrando que a seleção de interface será obrigatória somente se algum equipamento for selecionado na primeira tela de inclusão;

Figura 3.12: Exemplos de listagem de ocorrências de uma instituição

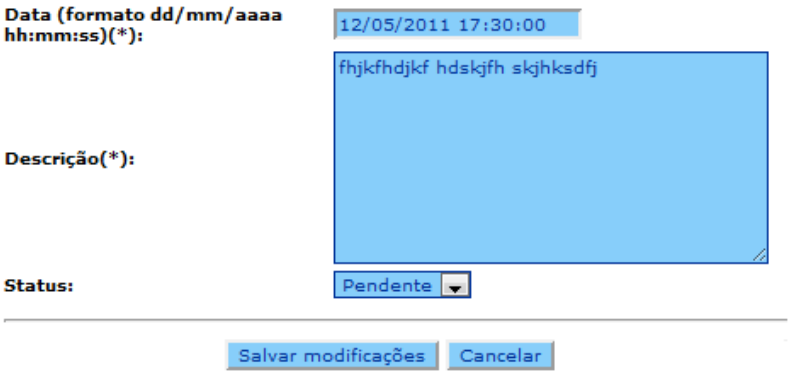


Figura 3.13: Tela de edição de ocorrências pendentes

- **Editar instituição**: Idem ao comando de incluir instituição, com a diferença de vir com os dados já cadastrados carregados nos campos, estando sujeitos a alterações;
- **Excluir instituição**: Remove uma instituição do sistema, juntamente com todas as ocorrências associadas à mesma;
- **Incluir ocorrência**: Incluir uma nova ocorrência, associada à instituição corrente. Uma nova ocorrência será sempre incluída com o status "Pendente". A figura 3.20 apresenta a tela de inclusão de ocorrências de uma instituição no sistema;
- **Lista completa de instituições**: Relação de instituições cadastradas, agrupadas por equipamento cadastrado no sistema;

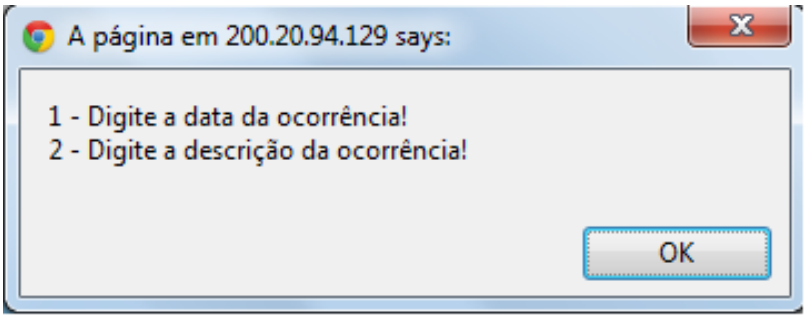


Figura 3.14: Campos obrigatórios na edição de ocorrência

  - **Lista completa de e-mails**: Listagem de todos os e-mails cadastrados no sistema.

    A figura 3.21 apresenta as funcionalidades do sistema relacionadas às instituições.

- **Equipamentos**: Tela onde é exibida a listagem de equipamentos de rede cadastrados no sistema, os dados técnicos dos mesmos, as interfaces de rede associadas ao equipamento corrente e onde é possível inserir, editar ou excluir equipamentos. A figura 3.22 apresenta um exemplo de equipamentos listados dentro do sistema;

  - **Detalhe de cada equipamento**: Tela onde são exibidos os dados técnicos de cada equipamento, e a relação de interfaces que lhe estão associadas, caso haja uma ou mais interfaces disponíveis para visualização;

    * **Dados do equipamento**: Relação de informações técnicas sobre o equipamento cadastrado. São elas:

      · **Nome do equipamento**: Nome com o qual o equipamento foi cadastrado no sistema;

      · **IP do equipamento**: Endereço IP do equipamento cadastrado;

      · **Monitorando interfaces?**: Variável que controla se o sistema irá ou não monitorar o tráfego de entrada e saída de dados nas interfaces do equipamento cadastrado;

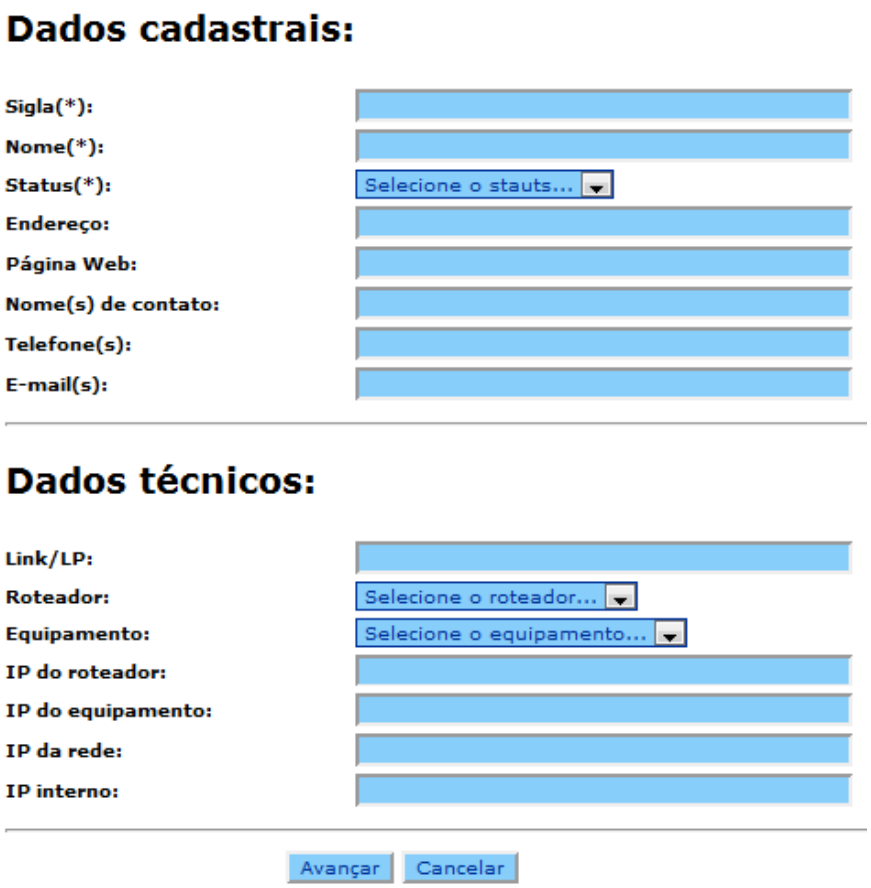


Figura 3.15: Incluir instituição (1)

- **Monitorando CPU?**: Variável que controla se o sistema irá ou não monitorar a taxa de utilização da CPU do equipamento cadastrado;
- **Monitorando temperatura?**: Variável que controla se o sistema irá ou não monitorar a temperatura do equipamento cadastrado;
- **OID ifIndex**: Identificador OID utilizado para obter os índices das interfaces associadas ao equipamento;
- **OID ifDesc**: Identificador OID utilizado para obter os nomes das interfaces associadas ao equipamento;
- **OID ifInOctets**: Identificador OID utilizado para obter

Figura 3.16: Incluir instituição (2)

Figura 3.17: Incluir instituição (3)

os totais de *bytes* recebidos pelas interfaces associadas ao equipamento;

- **OID ifOutOctets**: Identificador OID utilizado para obter os totais de *bytes* enviados pelas interfaces associadas ao equipamento;
- **OID ifAdminStatus**: Identificador OID utilizado para obter os status operacionais esperados para as interfaces associadas ao equipamento;

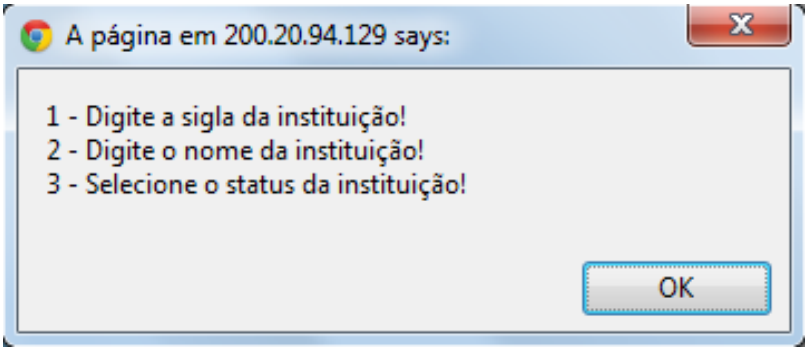


Figura 3.18: Campos obrigatórios na inclusão de instituição (1)

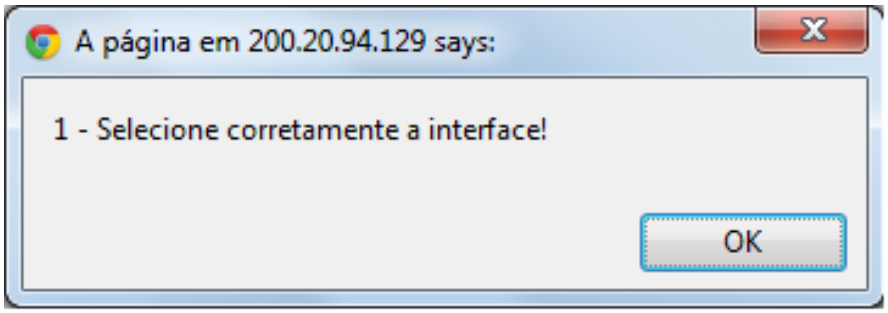


Figura 3.19: Campos obrigatórios na inclusão de instituição (2)

Figura 3.20: Tela de inclusão de ocorrências

- **OID ifOperStatus**: Identificador OID utilizado para obter os status operacionais correntes para as interfaces associadas ao equipamento;
- **OID CPU 1 Minuto**: Identificador OID utilizado para obter a utilização média da CPU por minuto do equipamento;
- **OID CPU 5 Minutos**: Identificador OID utilizado para obter a utilização média da CPU a cada 5 minutos do equipamento;
- **OID Temperatura In**: Identificador OID utilizado para obter a temperatura nas interfaces de entrada do equipamento;
- **OID Temperatura Out**: Identificador OID utilizado

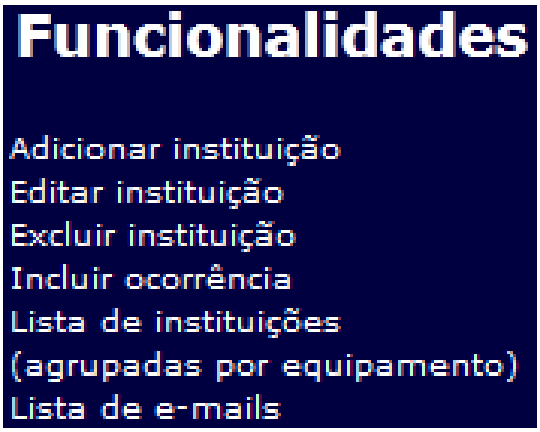


Figura 3.21: Tela de funcionalidades para instituições

para obter a temperatura nas interfaces de saída do equipamento;

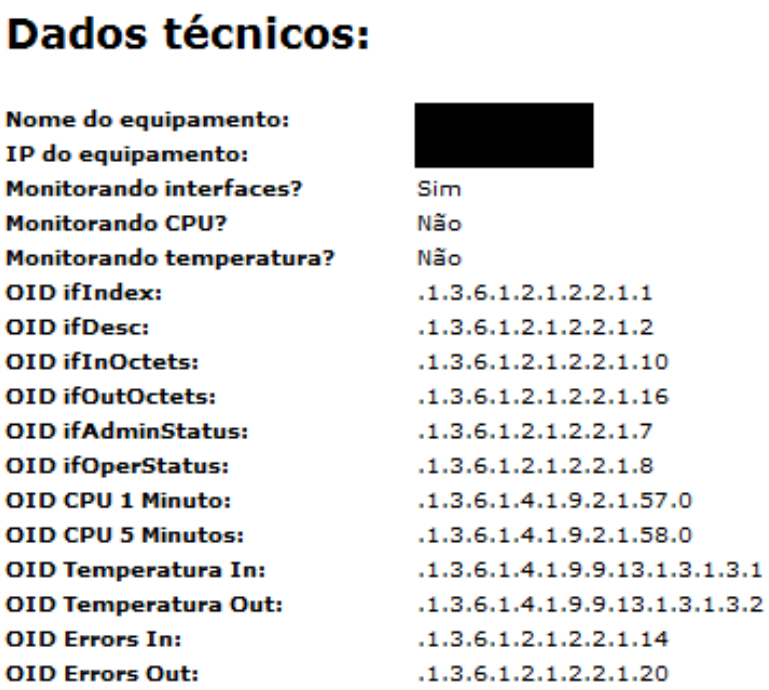
- **OID Errors In**: Identificador OID utilizado para obter os totais de *bytes* recebidos com erro pelas interfaces associadas ao equipamento;
- **OID Errors Out**: Identificador OID utilizado para obter os totais de *bytes* enviados com erro pelas interfaces associadas ao equipamento.

* **Ver interfaces do equipamento**: Listagem de interfaces associadas ao equipamento, com a descrição configurada no equipamento (ifDescr2); e a instituição associada à interface, no sistema. Esta listagem de interfaces será visualizada somente se a variável **Monitorando interfaces?** estiver definida com o valor "Sim".
  A figura 3.23 apresenta um exemplo de como ficam os dados técnicos de um dado equipamento.
  Já a figura 3.24 apresenta um exemplo de como são listadas as interfaces de um dado equipamento.
  - **Detalhes da interface**: Tela onde são exibidos os dados técnicos e o gráfico de monitoramento de tráfego (caso hajam dados a serem exibidos) da interface em questão.

Figura 3.22: Exemplo de listagem de equipamentos cadastrados

Estes dados são:

1. **ifIndex**: Identificador OID utilizado para obter o índice da interface associada ao equipamento;
2. **ifDescr**: Identificador OID utilizado para obter o nome da interface associada ao equipamento;
3. **ifType**: Identificador OID utilizado para obter o nome do tipo de interface associada ao equipamento;
4. **ifMtu**: Identificador OID utilizado para obter o tamanho máximo permitido, em *bytes*, que um pacote de dados pode ter para ser recebido/enviado pela interface associada ao equipamento;
5. **ifSpeed**: Identificador OID utilizado para obter a largura de banda, em *bits* por segundo, configurada para a interface associada ao equipamento;
6. **ifPhysAddress**: Identificador OID utilizado para obter o endereço físico da interface associada ao equipamento;
7. **ifAdminStatus**: Identificador OID utilizado para obter o status operacional esperado para a interface associada



Dados técnicos:

Nome do equipamento:	[redacted]
IP do equipamento:	
Monitorando interfaces?	Sim
Monitorando CPU?	Não
Monitorando temperatura?	Não
OID ifIndex:	.1.3.6.1.2.1.2.2.1.1
OID ifDesc:	.1.3.6.1.2.1.2.2.1.2
OID ifInOctets:	.1.3.6.1.2.1.2.2.1.10
OID ifOutOctets:	.1.3.6.1.2.1.2.2.1.16
OID ifAdminStatus:	.1.3.6.1.2.1.2.2.1.7
OID ifOperStatus:	.1.3.6.1.2.1.2.2.1.8
OID CPU 1 Minuto:	.1.3.6.1.4.1.9.2.1.57.0
OID CPU 5 Minutos:	.1.3.6.1.4.1.9.2.1.58.0
OID Temperatura In:	.1.3.6.1.4.1.9.9.13.1.3.1.3.1
OID Temperatura Out:	.1.3.6.1.4.1.9.9.13.1.3.1.3.2
OID Errors In:	.1.3.6.1.2.1.2.2.1.14
OID Errors Out:	.1.3.6.1.2.1.2.2.1.20

Figura 3.23: Listagem de dados técnicos de equipamento

ao equipamento;

8. **ifOperStatus**: Identificador OID utilizado para obter o status operacional corrente para a interface associada ao equipamento;
9. **ifLastChange**: Identificador OID utilizado para obter o tempo passado desde a última vez em que a interface associada ao equipamento foi inicializada em seu atual status operacional;
10. **ifInOctets**: Identificador OID utilizado para obter o total de *bytes* recebidos pela interface associada ao equipamento;
11. **ifInUcastPkts**: Identificador OID utilizado para obter o total de pacotes enviados, a partir da interface associada ao equipamento, pela camada atual para outra mais alta (mais próxima da camada de aplicação) que

Interfaces:

#	Nome	ifDescr2	Instituição
1	Serial3/0	ON - 34Mbps	ON 34
2	Serial3/1	UFRRJ - 34Mbps	UFRRJ
3	FastEthernet0/0		BB-RNP
4	Serial1/0	EMBRAPA GUARATIBA - 2Mbps	
5	Serial1/1	EMBRAPA GUARATIBA - 2Mbps	CPII L2
6	Serial1/2	IFRJ REALENGO - 2Mbps	
7	Serial1/3	INES - 2Mbps link 2	INES L2
8	Serial1/4	IFRJ PINHEIRAL - 2Mbps	CEFET/Quimica
9	Serial1/5	IFRJ PINHEIRAL - 2Mbps	CEFETQL3
10	Serial1/6		
11	Serial1/7		
12	Ethernet2/0	Gerencia OI - 64Kbps	OI
13	Ethernet2/1		
14	Ethernet2/2		
15	Ethernet2/3		
16	Serial4/0	UNED - Maria da Graca - 2Mbps	UNED_MariadaGraça
17	Serial4/1	DATASUS - 2Mbps	DATASUS
18	Serial4/2	INCL - 1Mbps	INCL
19	Serial4/3	INT - 2Mbps	
20	Serial4/4	UNED Nova Iguacu - 2Mbps	UNED_NovaIguaçu
21	Serial4/5		UFRRJ2
22	Serial4/6		UFRRJ3
23	Serial4/7		UFRRJ4
24	Null0		
25	Loopback0	CEFETQ - S1/4-5 - 4Mbps	
26	Loopback3	Gerencia OI	
27	Multilink100	EMBRAPA GUARATIBA - 2x2Mbps - S1/0 e S1/1	EMBRAPA_GUARATIBA
28	Multilink200	IFRJ PINHEIRAL - 2x2Mbps - S1/4 e S1/5	

Figura 3.24: Listagem de interfaces de equipamento

não receberam endereçamento para esta nova camada;

12. **ifInNUcastPkts**: Identificador OID utilizado para obter o total de pacotes enviados, a partir da interface associada ao equipamento, pela camada atual para outra mais alta (mais próxima da camada de aplicação) que receberam endereçamento para esta nova camada;

13. **ifInDiscards**: Identificador OID utilizado para obter o total de pacotes recebidos e descartados pela interface associada ao equipamento, mesmo sem apresentar

nenhum tipo de erro em seu recebimento. Tal descarte pode ocorrer em razão de eventual necessidade de se liberar espaço no armazenamento de pacotes a serem processados pela interface;

14. **ifInErrors**: Identificador OID utilizado para obter o total de *bytes* recebidos com erro pela interface associada ao equipamento;
15. **ifInUnknownProtos**: Identificador OID utilizado para obter o total de pacotes recebidos e descartados pela interface associada ao equipamento, em razão dos mesmos utilizarem protocolos de rede não reconhecidos pela interface;
16. **ifOutOctets**: Identificador OID utilizado para obter o total de *bytes* enviados pela interface associada ao equipamento;
17. **ifOutUcastPkts**: Identificador OID utilizado para obter o total de pacotes recebidos pela interface associada ao equipamento, enviados para outra camada mais alta que a da interface (mais próxima da camada de aplicação) que não receberam endereçamento para a camada da interface;
18. **ifOutNUcastPkts**: Identificador OID utilizado para obter o total de pacotes recebidos pela interface associada ao equipamento, enviados para outra camada mais alta que a da interface (mais próxima da camada de aplicação) que receberam endereçamento para a camada da interface;
19. **ifOutDiscards**: Identificador OID utilizado para obter o total de pacotes enviados e descartados pela interface associada ao equipamento, mesmo sem apresentar nenhum tipo de erro em seu recebimento. Tal descarte pode ocorrer em razão de eventual necessidade de se

liberar espaço no armazenamento de pacotes a serem processados pela interface;

20. **ifOutErrors**: Identificador OID utilizado para obter o total de *bytes* enviados com erro pela interface associada ao equipamento;
21. **ifOutQLen**: Identificador OID utilizado para obter o tamanho, em total de pacotes, da fila de envio de pacotes da interface associada ao equipamento;
22. **ifSpecific**: Identificador OID utilizado para obter referência para as definições da base MIB, específica para a mídia utilizada na construção da interface associada ao equipamento.

    A figura 3.25 apresenta um exemplo de como ficam os dados de uma dada interface.

· **Gráfico de tráfego**: Gráfico de monitoramento de tráfego de entrada e saída de *bytes* (octetos) da interface, medido em *bits* por segundo - Bps. A escala de monitoramento do aplicativo MRTG é adaptada à ordem de grandeza do tráfego, de acordo com o valor máximo atingido por este, ou seja, pode ser mensurada em *bits*, Kilobits, Megabtis ou Gigabits por segundo. O gráfico padrão apresenta sempre o monitoramento nas últimas 24 horas (gráfico diário).

1. **Gráficos por período**: Ao se clicar no gráfico principal, é aberta uma nova janela que exibe os gráficos de monitoramento de tráfego da interface em quatro períodos distintos. São eles:

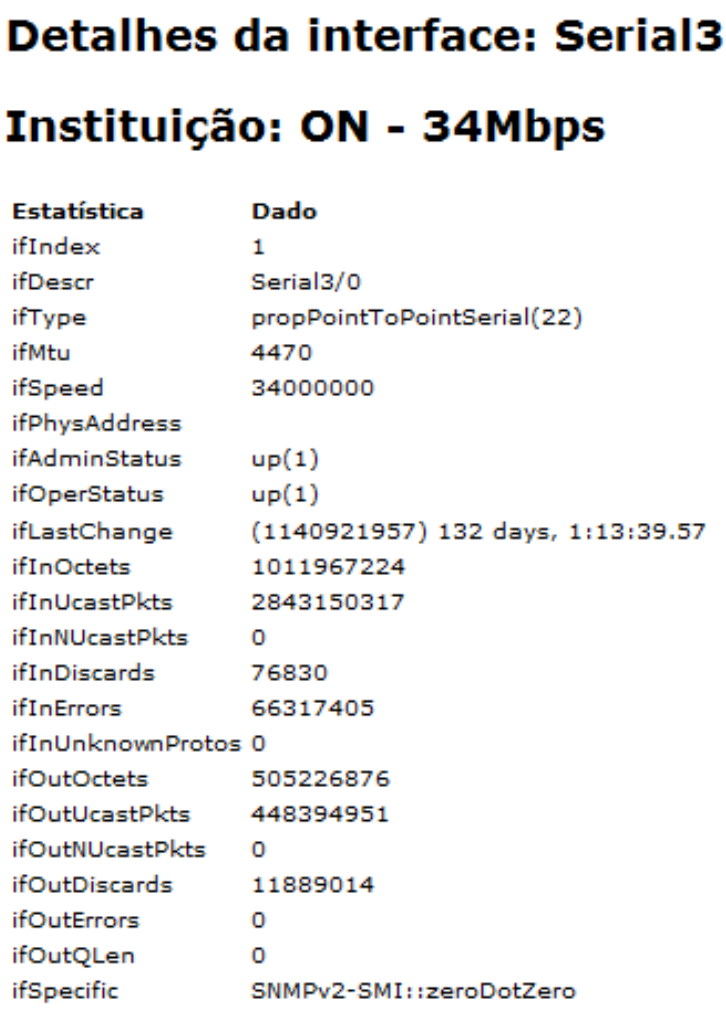
   Último Dia

   Última Semana

   Último Mês

   Último Ano

– **Incluir equipamento**: Tela para a inclusão de um equipamento



Detalhes da interface: Serial3/0

Instituição: ON - 34Mbps

Estatística	Dado
ifIndex	1
ifDescr	Serial3/0
ifType	propPointToPointSerial(22)
ifMtu	4470
ifSpeed	34000000
ifPhysAddress	
ifAdminStatus	up(1)
ifOperStatus	up(1)
ifLastChange	(1140921957) 132 days, 1:13:39.57
ifInOctets	1011967224
ifInUcastPkts	2843150317
ifInNUcastPkts	0
ifInDiscards	76830
ifInErrors	66317405
ifInUnknownProtos	0
ifOutOctets	505226876
ifOutUcastPkts	448394951
ifOutNUcastPkts	0
ifOutDiscards	11889014
ifOutErrors	0
ifOutQLen	0
ifSpecific	SNMPv2-SMI::zeroDotZero

Figura 3.25: Listagem de dados técnicos de equipamento

de rede. Nesta tela, são cadastrados o nome do equipamento, seu endereço IP e sua *community* utilizada para acesso, via protocolo SNMP, das estatísticas do equipamento. Os demais dados do equipamento são preenchidos automaticamente em sua criação, podendo ser editados posteriormente pelo usuário.

A figura 3.26 apresenta a tela de inclusão de equipamentos.

A figura 3.27 apresenta a mensagem que indica a relação de campos de preenchimento obrigatório na inclusão de equipamentos;

– **Editar equipamento**: Idem ao comando de incluir equipamento,

Figura 3.26: Incluir equipamento

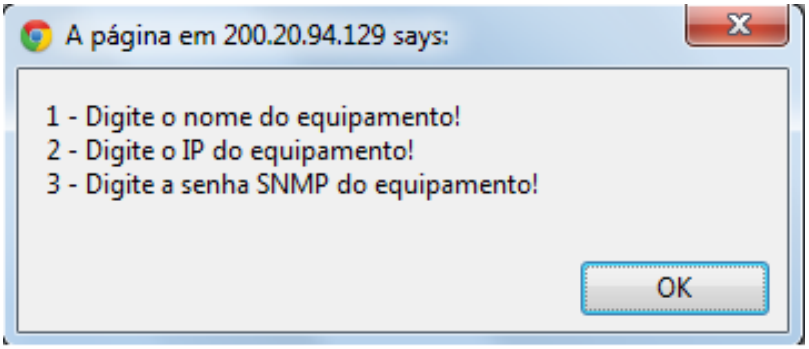


Figura 3.27: Campos obrigatórios na inclusão de equipamento

com a diferença de vir com os dados já cadastrados carregados nos campos, estando sujeitos a alterações. Além disto, diferentemente do que ocorre na inclusão, é possível editar a *community* utilizada para acesso às estatísticas (todas ou parte delas) do equipamento; e também todos os dados do equipamento listados na tela de detalhe.

A figura 3.28 apresenta a tela de edição de equipamentos.

A figura 3.29 apresenta a mensagem que indica a relação de campos de preenchimento obrigatório na edição de equipamentos;

– **Excluir equipamento**: Remove um equipamento do cadastro de equipamentos do sistema. Para se excluir um equipamento, é preciso que este não possua nenhuma instituição cadastrada associada às suas interfaces.

A figura 3.30 apresenta as funcionalidades do sistema que estão relacionadas aos equipamentos.

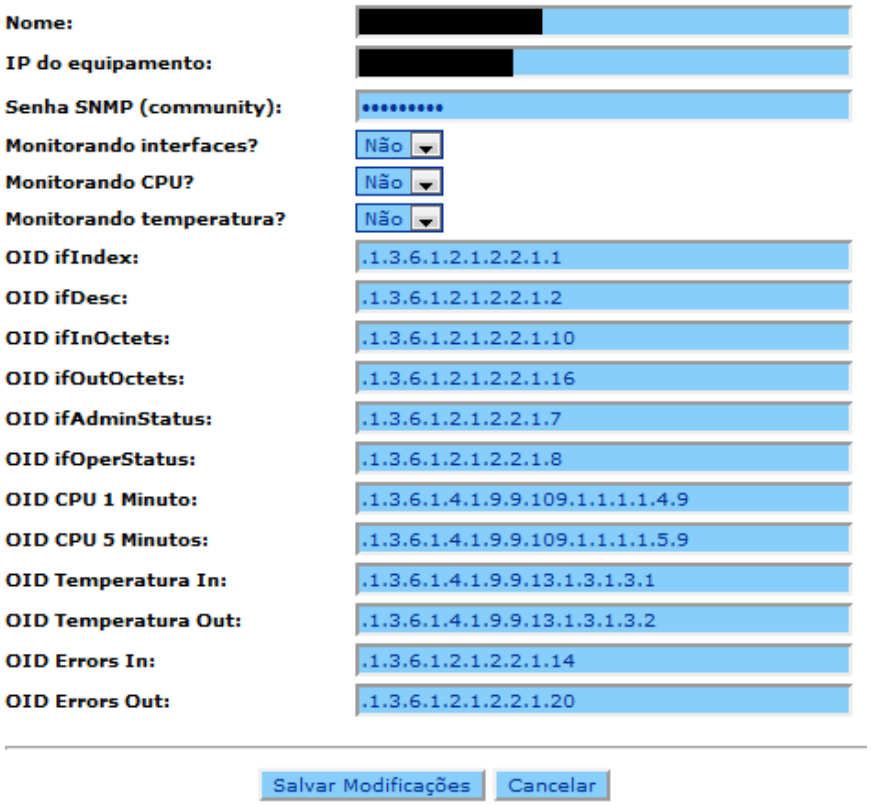


Figura 3.28: Editar equipamento

- **Monitoramento e gerência de rede**: Tela onde são disponibilizadas as funcionalidades de monitoramento da utilização das CPU's dos equipamentos cadastrados, e das temperaturas dos mesmos. Este monitoramento somente será realizado para os equipamentos onde as variáveis **Monitorando CPU?** e/ou **Monitorando temperatura?** estiverem definidas como "Sim".

  A figura 3.31 apresenta a listagem de monitoramentos disponíveis no sistema;

  - **Utilização das CPU's dos equipamentos**: Listagem de gráficos de monitoramento de utilização de CPU, em termos de percentual, dos equipamentos cadastrados no sistema, para os equipamen-

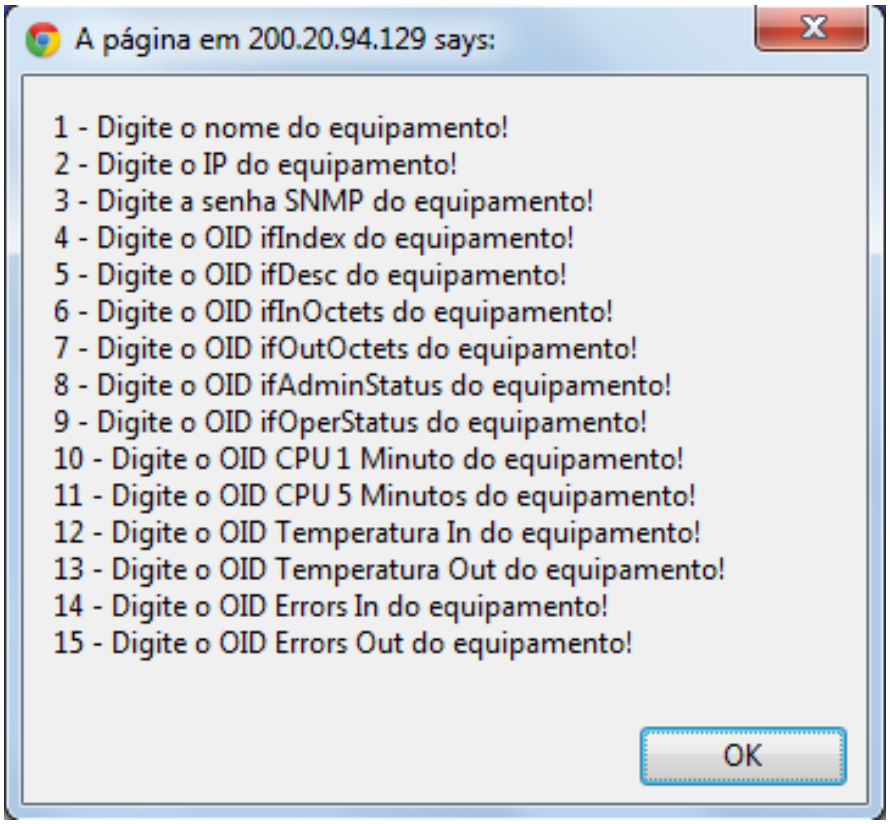


Figura 3.29: Campos obrigatórios na edição de equipamento

tos configurados para ter este monitoramento em especial. Estes gráficos apresentam a utilização média das CPU's por minuto, e por intervalos dos últimos 5 minutos.

A figura 3.32 apresenta um exemplo de gráfico de monitoramento de CPU de um dado equipamento de rede disponível no sistema;

* **Gráficos por período**: Ao se clicar no gráfico principal, é aberta um a nova janela que exibe os gráficos de monitoramento de utilização de CPU em quatro períodos distintos. São eles:
  - **Último Dia**
  - **Última Semana**
  - **Último Mês**
  - **Último Ano**

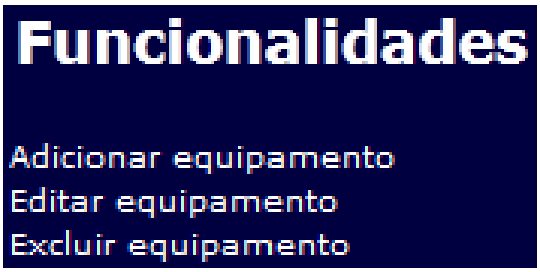


Figura 3.30: Tela de funcionalidades para equipamentos

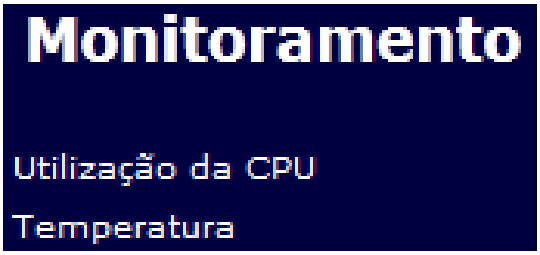


Figura 3.31: Monitoramentos disponíveis no sistema

– **Temperaturas dos equipamentos**: Listagem de gráficos de monitoramento de temperatura, medida em graus Celsius, dos equipamentos cadastrados no sistema, para os equipamentos configurados para ter este monitoramento em especial. Estes gráficos apresentam as temperaturas nas interfaces de entrada e saída de dados do equipamento.

  A figura 3.33 apresenta um exemplo de gráfico de monitoramento de temperatura de um dado equipamento de rede disponível no

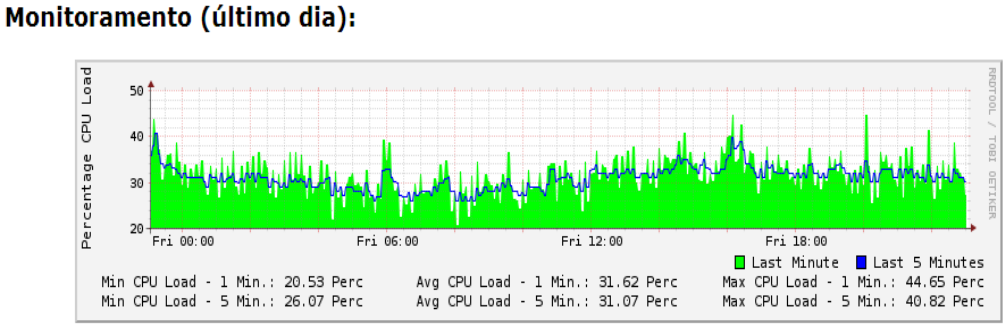


Figura 3.32: Monitoramento de CPU

sistema.

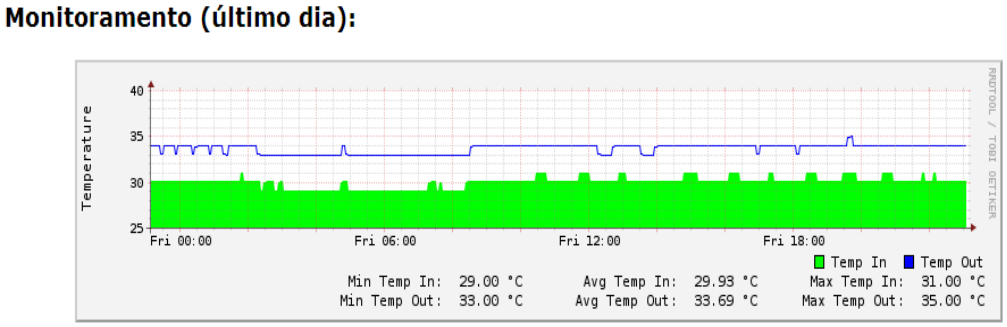


Figura 3.33: Monitoramento de temperatura

  * **Gráficos por período**: Ao se clicar no gráfico principal, é aberta um a nova janela que exibe os gráficos de monitoramento de temperatura em quatro períodos distintos. São eles:
    - **Último Dia**
    - **Última Semana**
    - **Último Mês**
    - **Último Ano**

- **Ocorrências pendentes**: Tela onde é disponibilizada a listagem completa de ocorrências cadastradas no sistema com status "Pendente";

  – **Listagem de ocorrências pendentes, com *link*s para as instituições**: Ao se clicar na ocorrência pendente, o sistema redireciona o usuário para a tela de detalhe da instituição, onde é possível então se editar a mesma.

  A figura 3.34 apresenta um exemplo de listagem de ocorrências pendentes cadastradas no sistema.

- **Conjuntos pessoais de gráficos**: Tela onde é disponibilizada a listagem completa de conjuntos de gráficos criados para monitorar o tráfego de algumas interfaces de equipamentos de rede, de acordo com o que o usuário deste conjunto em especial desejar monitorar. Ao clicar em um

Figura 3.34: Listagem de ocorrências pendentes

destes conjuntos, os gráficos monitorados pelo mesmo serão exibidos na tela.

A figura 3.35 apresenta um exemplo de listagem de conjuntos de gráficos cadastrados no sistema.

Já a figura 3.36 apresenta um exemplo de detalhamento de um dado conjunto de gráficos cadastrado no sistema;

- **Busca**: Tela onde são disponibilizadas algumas ferramentas de busca de informações dentro do sistema. São estas ferramentas:

  - **Busca geral**: O texto passado como parâmetro de busca será procurado nos campos de sigla, nome, endereço, página WEB, contato, telefone e e-mail de todas as instituições cadastradas no sistema. Para as instituições onde esta busca achar um os mais resultados compatíveis nos campos listados anteriormente, tal relação de campos será apresentada na página de resultados, destacando tais ocorrências em **negrito**. Haverá ainda um *link*, através do campo de sigla, para a tela de detalhe da instituição;

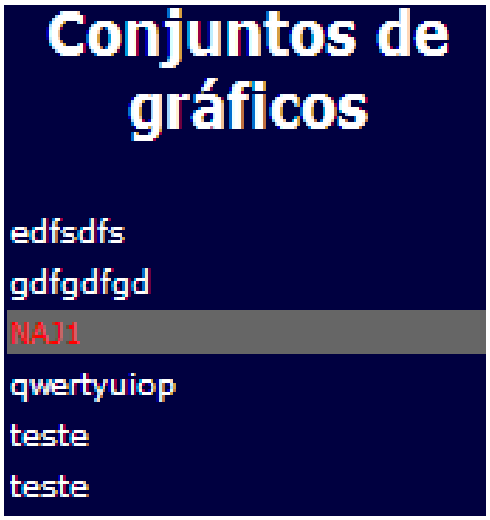


Figura 3.35: Listagem de conjuntos de gráficos

– **Busca por Link Permanente (LP)**: O texto passado como parâmetro de busca será procurado no campo **Link/LP** de todas as instituições cadastradas no sistema. Para as instituições onde esta busca achar resultados compatíveis, será apresentada na página de resultados a relação de siglas de cada instituição, juntamente com o Link Permanente de cada uma delas, destacando as ocorrências do critério de busca em **negrito**. Haverá ainda um *link*, através do campo de sigla, para a tela de detalhe da instituição;

– **Busca por ocorrência**: O texto passado como parâmetro de busca será procurado no campo identificador do número da ocorrência, conforme este estiver cadastrado no banco de dados, se for o caso. Sendo também possível buscar todas as ocorrências cadastradas no sistema, caso o usuário selecione a opção "Buscar todas". O sistema então retornará a ocorrência buscada individualmente, ou buscará todas as ocorrências cadastradas, dependendo do que for passado como parâmetro pelo usuário. Para cada ocorrência retornada na busca, constarão os dados de número identificador, sigla da instituição vinculada, data, descrição e sta-

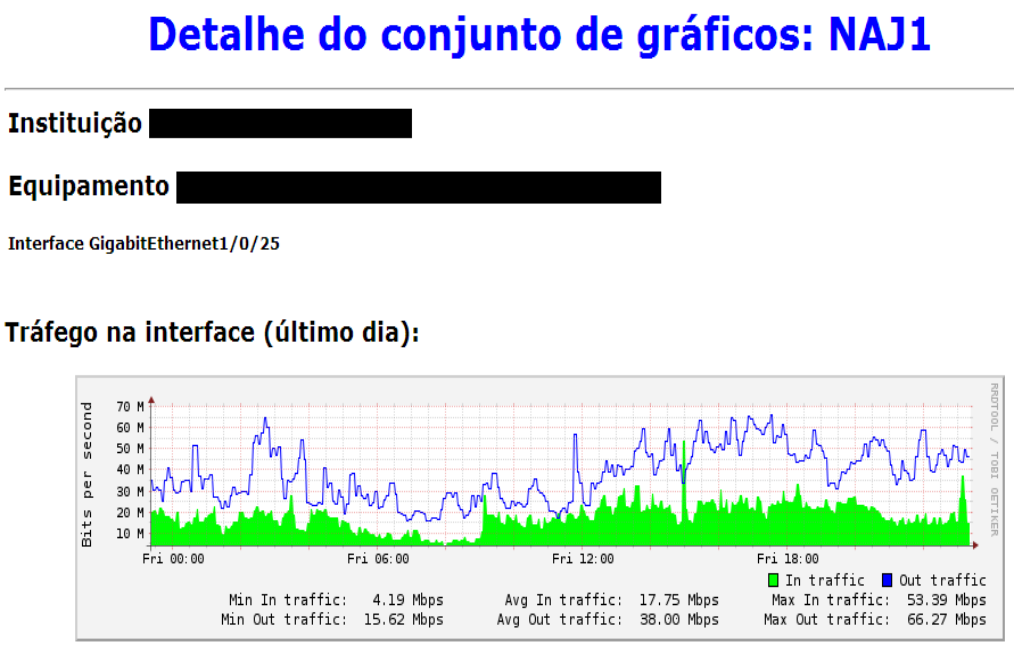


Figura 3.36: Detalhe de um conjunto de gráficos

tus atual da ocorrência. Haverá ainda um *link*, através do campo de sigla, para a tela de detalhe da instituição.

A figura 3.37 apresenta os critérios de busca disponíveis no sistema;

- **Links e Utilitários**: Tela onde são disponibilizados alguns *link*s de contato e ferramentas de gerência de rede disponibilizadas atualmente pela RedeRio. São eles:
  - **RedeRIO**: *Link* para o site oficial da RedeRio;
  - **Telefones úteis**: Telefones de contato com as empresas Embratel e Telemar (Oi), para questões relacionadas aos serviços de provedores de Internet disponibilizados por ambas;
  - **Ping**: Ferramenta de Ping disponibilizada pelo site oficial da RedeRio. Seu funcionamento se baseia no aplicativo PING, assim como a funcionalidade **Teste de conectividade do IP de rede da instituição** desenvolvida para o sistema de monitoramento descrito nesta dissertação, com a diferença que, neste teste

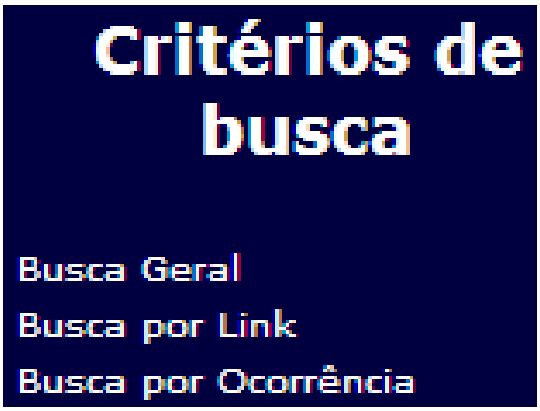


Figura 3.37: Critérios de busca disponíveis

disponibilizado pela RedeRio, serão sempre enviados 10 pacotes de 72 *bytes* cada;

- **Traceroute**: Ferramenta de Traceroute disponibilizada pelo site oficial da RedeRio. Seu funcionamento se baseia no aplicativo Traceroute, utilizado para determinar um caminho de rede disponível entre o *host* que iniciou a requisição e o *host* passado como parâmetro, apresentando ainda a relação de *hosts* percorridos entre a origem e o destino, formando assim o "caminho" de rede entre estes dois *hosts*;
- **Sobre a gerência GRRW**: Descrição geral sobre os objetivos do sistema de monitoramento e gerência de redes explicado nesta dissertação.

- **Documentação**: Tela onde são disponibilizados documentos relativos ao projeto RedeRio (futuramente Redecomep);

- **Preferências**: Tela de configurações de preferências de usuário, para edição de conjuntos de gráficos de interesse e criação/edição de *login*s e senhas de acesso ao sistema de monitoramento.

  A figura 3.38 apresenta a listagem de preferências disponíveis no sistema;

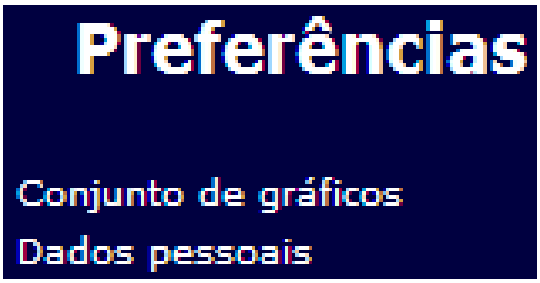


Figura 3.38: Preferências disponíveis

– **Conjunto de gráficos**: Conforme mostrado na figura 3.35, trata-se da tela onde é disponibilizada a listagem completa de conjuntos de gráficos criados para monitorar o tráfego de algumas interfaces de equipamentos de rede, de acordo com o que o usuário deste conjunto em especial desejar monitorar. Ao clicar em um destes conjuntos, os gráficos monitorados pelo mesmo serão exibidos na tela.

  A figura 3.39 apresenta a listagem de funcionalidades disponíveis no sistema, relacionadas à gerência dos conjuntos de gráficos criados;

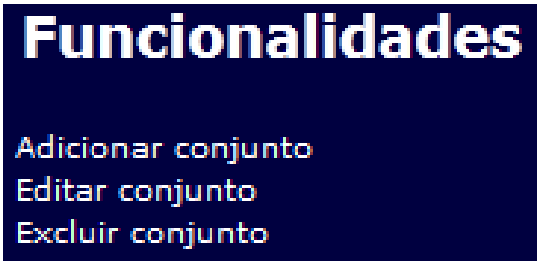


Figura 3.39: Funcionalidades disponíveis para conjuntos de gráficos

  * **Criar conjunto**: Tela para a inclusão de conjunto de gráficos. Nesta tela, são cadastrados o nome do conjunto; e a relação de gráficos que o mesmo irá monitorar. Para a escolha dos gráficos de monitoramento de tráfego nas interfaces, são carregadas as interfaces existentes em todos os equipamentos cadastrados no sistema. Esta carga das interfaces é realizada

através do uso do aplicativo *snmpwalk* para cada equipamento cadastrado.
A figura 3.40 apresenta a tela de inclusão de um novo conjunto de gráficos.

Figura 3.40: Incluir conjunto de gráficos

Nesta tela, o campo Nome é de preenchimento obrigatório, bem como é obrigatório selecionar pelo menos uma interface de rede para inclusão no conjunto. Caso estes requisitos não sejam satisfeitos, será exibida a mensagem ilustrada pela figura 3.41;

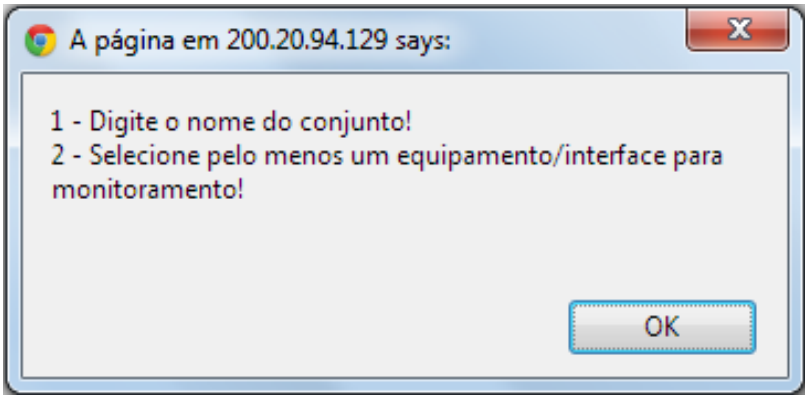


Figura 3.41: Campos obrigatórios na inclusão de um conjunto de gráficos

  - **Editar conjunto**: Idem ao comando de incluir conjunto, com a diferença de vir com os dados já cadastrados carregados nos campos, estando sujeitos a alterações;
  - **Excluir conjunto**: Remove um conjunto de gráficos do sistema.

- **Dados pessoais de usuário**: Listagem de usuários cadastrados no sistema.

  A figura 3.42 apresenta a listagem de usuários cadastrados no sistema.

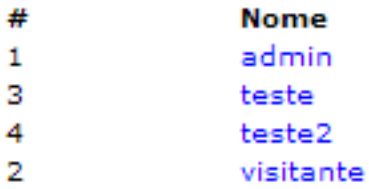


Figura 3.42: Listagem de usuários cadastrados

  - **Detalhe do usuário**: Tela onde são exibidos os dados de cada usuário. São eles:
    - **Nome do usuário**: Nome utilizado para o *login* no sistema;
    - **Nível de acesso**: Nível que determina que tipo de operações o usuário pode realizar. Caso seja nível "**admin**", o usuário tem livre acesso a todas as funcionalidades do sistema; caso seja nível "**visitante**", este terá acesso somente para leitura de informações, sem poder realizar nenhuma modificação ao sistema.

    A figura 3.43 apresenta os dados de um dado usuário cadastrado no sistema;
  - **Criar usuário**: Tela para a inclusão de um usuário no sistema. Nesta tela, são cadastrados o *login*, o nível de acesso e a senha do usuário para se *log*ar no sistema. É solicitado

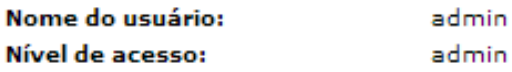


Figura 3.43: Dados de usuário cadastrado

ainda que a senha seja digitada uma segunda vez, para sua confirmação.

A figura 3.44 apresenta a tela de inclusão de um novo usuário.

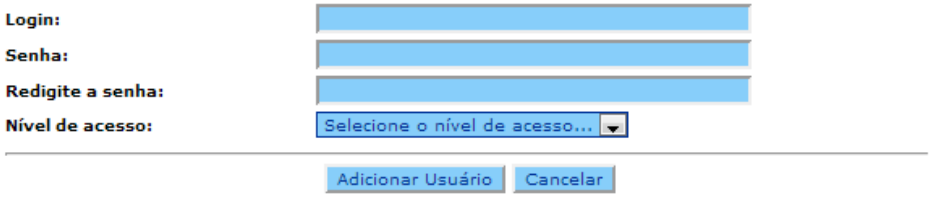


Figura 3.44: Incluir usuário

Nesta tela, todos os campos são de preenchimento obrigatório, e o campo de redigitação de senha deve conter o mesmo texto que o campo de senha. Caso estes requisitos não sejam satisfeitos, será apresentada a mensagem ilustrada pela figura 3.45;

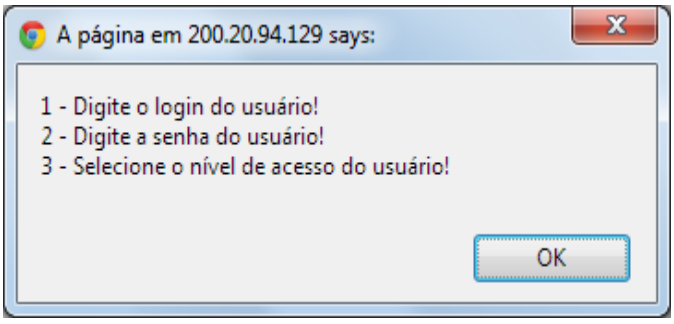


Figura 3.45: Campos obrigatórios na inclusão de um usuário

* **Editar usuário**: Idem ao comando de criar usuário, com a diferença de vir com os dados já cadastrados carregados nos campos, estando sujeitos a alterações.

A figura 3.46 apresenta a listagem de funcionalidades disponíveis no sistema, relacionadas à gerência dos usuários cadastrados.

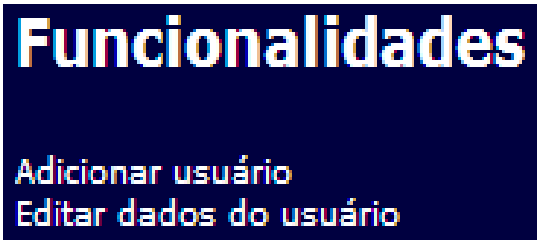


Figura 3.46: Funcionalidades disponíveis para usuários cadastrados

- **Logout**: Retorna à tela inicial de *login* no sistema.

3.4.2 Etapa 2

3.4.2.1 Simulação com uso do aplicativo Packet Tracer

Para a realização da Etapa 2, inicialmente foram feitas algumas simulações de rede no aplicativo Packet Tracer, da Cisco, a fim de verificar o comportamento do protocolo Spanning Tree, quando do momento de se definir a Árvore de Custo Mínimo da rede simulada. Dentro desta proposta, foram realizados os seguintes passos:

- Criação de uma rede em anel, com 5 *switches* Cisco Catalyst 2950T-24 ligando dois deles a um PC diferente cada, sem realizar nenhuma modificação em relação a seus endereços; ou suas larguras de banda, conectados uns aos outros conforme apresentado na figura 3.47.

  Conforme poderá ser observado na figura 3.47, os custos associados a cada conexão entre *switches* estão com o valor padrão 19. Este valor está associado à largura de banda da conexão estabelecida entre os *switches*, neste caso 100 Mbps. Os custos associados às larguras de banda seguem o critério exposto na tabela 3.2 a seguir, onde, quanto maior a largura de banda, menor o custo para se efetuar envio/recebimento de dados;

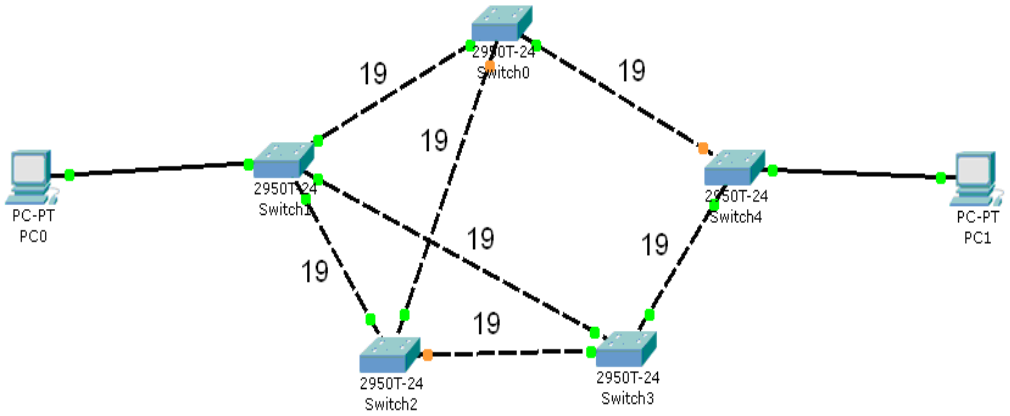


Figura 3.47: Configuração inicial da rede simulada via Packet Tracer

Tabela 3.1: Custos associados às larguras de banda - Packet Tracer

Largura de banda	Custo
10 Mbps	100
100 Mbps	19
1 Gbps	4
10 Gbps	2

- Uma vez criada esta rede inicial, configurar os 2 PC's aos endereços IP, às mascaras de rede, e aos *gateways* padrão indicados na tabela 3.2;
- Configurar os *switches* da rede, com os endereços IP indicados na tabela 3.3;
- Modificar a prioridade do *switch* Switch0, colocando-a com o valor de 4096;
- Modificar a prioridade do *switch* Switch0, colocando-a com o valor de 36864;

Tabela 3.2: Endereços IP da simulação com Packet Tracer

PC-PT	PC0	PC1
IP Configuration	Static	Static
IP Address	192.168.0.10	192.168.1.10
Subnet Mask	255.255.255.0	255.255.255.0
Default Gateway	192.168.0.1	192.168.1.1

Tabela 3.3: Endereços IP dos *switches* da simulação com Packet Tracer

2950T-24	Switch0	Switch1
IP Address	192.168.2.1	192.168.0.1
Subnet Mask	255.255.255.0	255.255.255.0
Switch2	Switch3	Switch4
192.168.4.1	192.168.3.1	192.168.1.1
255.255.255.0	255.255.255.0	255.255.255.0

- Configurar o *switch* Switch0 como raiz primária da rede;
- Modificar a prioridade do *switch* Switch0, colocando-a com o valor de 32768;
- Desabilitar todas as interfaces de rede do *switch* Switch1;
- Habilitar novamente todas as interfaces de rede do *switch* Switch1;
- Configurar o *switch* Switch1 como raiz primária da rede, o *switch* Switch4 como raiz secundária, e desabilitando em seguida o *switch* Switch1;
- Habilitar novamente o *switch* Switch1;
- Modificar a prioridade do *switch* Switch4, colocando-a com o valor de 32768;
- Desabilitar a interface de rede FastEthernet0/4 do *switch* Switch1;
- Habilitar novamente a interface de rede FastEthernet0/4 do *switch* Switch1;
- Configurar a velocidade da interface de rede FastEthernet0/4 do *switch* Switch1 para 10 Mbps;
- Configurar a velocidade da interface de rede FastEthernet0/3 do *switch* Switch3 para 10 Mbps.

3.4.2.2 Análise de ferramentas para implementação

Em um momento posterior à simulação realizada no aplicativo Packet Tracer, iniciou-se um processo de análise de ferramentas que poderiam ser utilizadas, a fim de se criar uma solução WEB de monitoramento de equipamentos de rede. Para isto, foi formulado um questionário, enviado para cada uma das gerências regionais do projeto Redecomep, com o conteúdo a seguir:

Bom dia,

Sou aluno de Mestrado Profissional, com ênfase em Instrumentação Científica, no Centro Brasileiro de Pesquisas Físicas, sob orientação do professor Nilton Alves Jr., o qual atua na gerência da RedeRio, e também na implantação da Redecomep no estado do Rio de Janeiro.

Minha dissertação está relacionada às atividades de gerência e monitoração de redes, com ênfase na implementação de um módulo de monitoramento baseado no protocolo Spanning Tree. Tendo todas estas questões colocadas, gostaria de saber se é possível contar com sua colaboração em responder ao questionário a seguir:

- **A rede implementada por vocês possui *switches* conectados com alguma redundância (dois ou mais caminhos possíveis entre pontos de rede)?**
- **O protocolo Spanning Tree está implementado na rede?**
- **Caso o protocolo Spanning Tree não esteja implementado, existe a possibilidade de uma implantação futura do mesmo?**
- **Qual versão do protocolo Spanning Tree está sendo utilizada?**
- **As portas e/ou *switches* que se sabe inicialmente que não possuirão loops estão configuradas como "Fast Switching"?**
- **Existe alguma solução de monitoramento de redes, baseada no protocolo Spanning Tree?**

- **Caso exista tal solução, quais as tecnologias utilizadas para a implementação da mesma?**

- **É possível, através desta solução, se monitorar o comportamento de rede com o protocolo Spanning Tree, através de um aplicativo Web (site, por exemplo)?**

- **Caso não exista tal solução, existe a possibilidade de implementação de uma solução baseada em página Web, para o monitoramento do protocolo Spanning Tree?**

Desde já grato pela compreensão.
Atenciosamente,

Conforme pode ser observado no corpo do questionário enviado, foi questionado se as redes implementadas:

- Possuem, de alguma forma, em algum ponto, a possibilidade de ter dois ou mais caminhos possíveis entre dois *switches* diferentes (redundâncias na rede);

- Se existe alguma implementação do protocolo Spanning Tree, para que seja viável realizar um monitoramento WEB dos equipamentos de rede, através deste protocolo;

- No caso de não haver tal implementação, se é possível realizá-la de alguma forma;

- No caso de haver tal implementação, qual versão do protocolo utilizada na mesma;

- Se existe a configuração "Fast Switching" implementada em alguma porta sem redundâncias, ou seja, se esta está configurada para agilizar o processo de cálculo da Árvore de Custo Mínimo, em razão desta ausência de redundâncias;

- Se, a partir da existência do protocolo Spanning Tree implementado, foi criada alguma ferramenta de monitoramento, baseada neste protocolo. Se sim, se esta solução é utilizável com tecnologia WEB;

- Em caso negativo, se é possível a criação de uma solução WEB que contemple o monitoramento de redes com o protocolo Spanning Tree.

Após o envio deste e-mail, foram enviadas 5 respostas, com os seguintes resultados:

- A primeira resposta recebida, enviada pela equipe do órgão RNP no estado do Paraná, informou que "A resposta é não para quase tudo.";

- Na segunda resposta, enviada pela equipe do órgão RNP no estado da Bahia, inicialmente comentou-se que "Aqui usamos o protocolo EAPS, desenvolvido pela ExtremeNetworks. Como são soluções não-compatíveis (protocolos EAPS e Spanning Tree), não temos como ter ambos na mesma infraestrutura, além de que o EAPS nos atende perfeitamente. Nossa visão é ter o EAPS dentro das VMANs (QinQ) da Redecomep - Remessa para usar junto aos usuários, pois muitos se conectam em diversos pontos e possuem infraestrutura própria conectando os mesmos prédios que a Remessa, então pode criar um loop inter-domínio.".

  Foi colocado também que a "idéia em responder é lhe passar a experiência da UFBA no uso de STP".

  Após isto, o representante do órgão RNP/BA forneceu as seguintes informações, para as perguntas formuladas no questionário:

  1. Sim, nos *switches* do *datacenter*. As redes são em anel.
  2. Sim, mas não o STP padrão, e sim o Multiple STP.
  3. Sem resposta cabível, em função da pergunta anterior ter apresentado resposta positiva.
  4. Multiple STP. Por quê? Para fazermos balanceamento de tráfego, ou seja, mandar umas vlans por um "lado" e outras por outro "lado".

5. Fast Start não? Sim, estão configuradas, mas porque o ambiente de datacenter é mais controlado.
6. Infelizmente, só o Syslog para avisar quando houver alteração topológica.
7. Idem à resposta anterior.
8. Possível é, mas não temos nada implantado nem conheço solução. Eu tive idéia no passado de desenvolver um *software* para fazer isso, para uso nas unidades acadêmicas, mas não deu certo. A heterogeneidade dos nossos *switches* pesou contra a solução.
9. No momento, não.

- Na terceira resposta, enviada pela equipe do órgão RNP em nível nacional, inicialmente comentou-se que "Em nosso projeto de redes metropolitanas, os *switches* são sempre conectados com topologia em anel, provendo desta forma redundância física aos mesmos.

  Porém, a configuração lógica das redes fica a critério de cada um dos consórcios. Os consórcios são locais e independentes entre si, e cada um tem a prerrogativa de configurar a topologia lógica da melhor forma que lhes convier. Assim sendo, é complicado eu lhe dar uma visão específica do que acontece em cada rede.

  O que posso lhe dizer é que os equipamentos que são adquiridos pelo projeto são dos fabricantes Extreme e Cisco, e ambos obviamente suportam protocolo Spanning Tree, sendo que os Cisco ainda têm algumas features proprietárias. Contudo, eu não saberia lhe dizer quais destas funções são ou não implementadas nas redes.

  Para esta visão mais detalhada relacionada aos seus questionamentos, seria necessário que conversasse com o pessoal técnico de cada uma das redes.";

- Na quarta resposta, enviada pela equipe do órgão RNP no estado da Paraíba, foram fornecidas as seguintes informações, para as perguntas formuladas no questionário:

1. Sim.
2. Não. Usamos o protocolo EAPS (Ethernet Authomatic Protection Switching / RFC 3619) implementado em equipamentos Extreme Networks (família Summit X450), que apresenta um tempo de convengència muito curto (da ordem de 60 ms).
3. Não.
4. Não é usada.
5. Não.
6. Monitoramos nossa rede via SNMP.
7. Não usamos.
8. Ver resposta anterior.
9. Ver resposta anterior.

- Na quinta resposta, enviada pela equipe do órgão RNP no estado do Rio Grande do Sul, foram fornecidas as seguintes informações, para as perguntas formuladas no questionário:

  1. Sim.
  2. Sim.
  3. Sem resposta cabível, em função da resposta anterior.
  4. 802.1w.
  5. Sim.
  6. Não.
  7. Sem resposta cabível, em função da resposta anterior.
  8. Ver resposta anterior.
  9. Ver resposta anterior.

Com base nas respostas enviadas, observou-se a inexistência de quaisquer soluções WEB para monitoramento de equipamentos de rede, com base no

protocolo Spanning Tree. Observou-se também que, quando algum monitoramento baseado neste protocolo é realizado, geralmente utiliza-se o protocolo SNMP, ou o aplicativo Syslog, levando-se assim a se considerar estas duas hipóteses para a criação de uma solução WEB.

A hipótese do aplicativo Syslog acabou descartada, em função deste aplicativo registrar, em seus arquivos de *log*, informações sobre tudo o que é processado no equipamento monitorado por ele. Isto leva a uma grande dificuldade em se filtrar a informação realmente necessária, em termos de protocolo Spanning Tree, devido a grande carga de informação gravada pelo aplicativo Syslog.

Mais adiante, ainda nesta seção, será discutida a viabilidade da utilização do protocolo SNMP na monitoração do protocolo Spanning Tree em equipamentos de rede.

Além deste questionário, também considerou-se a possibilidade de se utilizar o aplicativo Nagios, para se efetuar o monitoramento de rede, utilizando o protocolo Spanning Tree. No entanto, conforme explicitado na seção 1.3, este monitoramento só é possível quando os equipamentos estão conectados diretamente ao *servidor* que suporta a aplicação, o que não é o caso da situação aqui exposta.

3.4.2.3 Solução utilizada

Ao se fazer a análise do aplicativo Nagios, observou-se que o mesmo possui um *plugin* que permitiria o monitoramento de dados dos equipamentos de rede, através do protocolo Spanning Tree, utilizando-se de estatísticas obtidas por intermédio do protocolo SNMP, que possui um base MIB específica para obtenção de tais valores. Esta MIB chama-se **dot1dStp** e possui o identificador OID **1.3.6.1.2.1.17.2**.

Mesmo com o insucesso na tentativa de se utilizar o aplicativo Nagios para o que este trabalho se propõe, a situação descrita no parágrafo anterior permitiu a observação de que é possível utilizar o protocolo SNMP monitorar os equipamentos de rede do *backbone* da RedeRio, tendo por base o protocolo Spanning Tree.

A partir deste conhecimento, buscou-se no *site* da empresa Cisco, mais especificamente na página de suporte de utilização do protocolo SNMP [58], a base de dados MIB responsável pelo monitoramento de dados de um dado equipamento de rede, com o uso do protocolo Spanning Tree, chegando-se assim à base MIB **dot1dStp**.

Dentre as estatísticas existentes nesta base MIB, as utilizadas na solução desenvolvida são:

- **1.3.6.1.2.1.17.2.1 - dot1dStpProtocolSpecification**: Indica a versão do protocolo Spanning Tree que está sendo utilizada pelo equipamento monitorado. Pode assumir os seguintes valores:
  1. 1: *unknown*
  2. 2: *decLb100*
  3. 3: *ieee8021d*
- **1.3.6.1.2.1.17.2.2 - dot1dStpPriority**: Indica o valor de prioridade do equipamento monitorado;
- **1.3.6.1.2.1.17.2.5 - dot1dStpDesignatedRoot**: Indica o endereço físico (MAC Address) do equipamento eleito como raiz da árvore gerada pelo protocolo Spanning Tree;
- **1.3.6.1.2.1.17.2.6 - dot1dStpRootCost**: Indica o custo do caminho entre o equipamento raiz e o equipamento monitorado no momento;
- **1.3.6.1.2.1.17.2.7 - dot1dStpRootPort**: Indica o número da porta do equipamento corrente que oferece o caminho de menor custo até o equipamento raiz;
- **1.3.6.1.2.1.17.2.15 - dot1dStpPortTable**: Tabela de dados do protocolo Spanning Tree. Para cada porta do equipamento monitorado, haverá uma tabela seguindo este modelo. As estatísticas desta tabela contempladas na solução implementada são:
  - **1.3.6.1.2.1.17.2.15.1.1 - dot1dStpPort**: O número da porta monitorada no momento;

- **1.3.6.1.2.1.17.2.15.1.2 - dot1dStpPortPriority**: Indica o valor de prioridade da porta monitorada;
- **1.3.6.1.2.1.17.2.15.1.3 - dot1dStpPortState**: Indica o estado da porta, em termos de protocolo Spanning Tree. Pode ser:
  - 1: *disabled*
  - 2: *blocking*
  - 3: *listening*
  - 4: *learning*
  - 5: *forwarding*
  - 6: *broken*
- **1.3.6.1.2.1.17.2.15.1.4 - dot1dStpPortEnable**: Indica se a porta está ou não habilitada para transmissão de dados via protocolo Spanning Tree. Pode ser:
  - 1: *enabled*
  - 2: *disabled*
- **1.3.6.1.2.1.17.2.15.1.5 - dot1dStpPortPathCost**: Indica o custo que esta porta acrescenta ao caminho até o nó raiz da Árvore de Custo Mínimo;
- **1.3.6.1.2.1.17.2.15.1.7 - dot1dStpPortDesignatedCost**: Indica o custo do caminho de rede conectado à porta;
- **1.3.6.1.2.1.17.2.15.1.8 - dot1dStpPortDesignatedBridge**: Indica o endereço físico (MAC Address) do equipamento de rede que está conectado ao equipamento monitorado, através da porta corrente;
- **1.3.6.1.2.1.17.2.15.1.9 - dot1dStpPortDesignatedPort**: Indica o número de conexão da porta que se conecta à porta do equipamento monitorado, estabelecendo a conexão deste com outro equipamento de rede.

Tendo por base esta MIB, tornou-se possível o desenvolvimento da solução de monitoramento proposta nesta dissertação. A criação dela contemplou duas abordagens principais, descritas a seguir:

- A primeira abordagem consistiu em exibir, a partir de um conjunto de equipamentos de rede selecionados pelo usuário, os dados que os mesmos oferecem, em termos de protocolo Spanning Tree. Em outras palavras, será apresentado um resultado textual, categorizando os dados por equipamento selecionado, e para cada porta (interface de rede) de cada equipamento, também serão exibidos seus respectivos dados de monitoramento Spanning Tree.

  A figura 3.48 apresenta a forma como o usuário pode determinar quais equipamentos de rede terão seus dados de protocolo Spanning Tree apresentados textualmente no sistema de monitoramento.

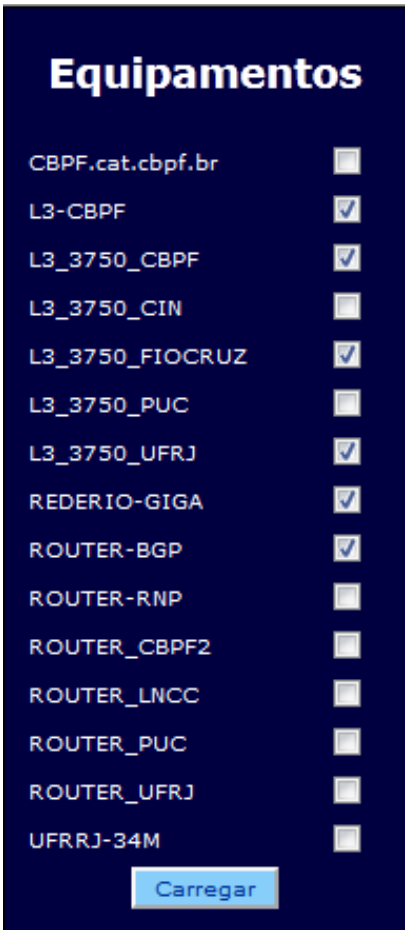


Figura 3.48: Seleção de equipamentos para monitorar, via Spanning Tree

Cade observar que, além dos dados obtidos via protocolo SNMP sobre

monitoramento Spanning Tree, será exibido também o endereço físico de cada equipamento, obtido pelo identificador OID **1.3.6.1.2.1.17.1.1**. Em caso de insucesso na obtenção das estatísticas desejadas, o sistema retornará uma mensagem de erro, atestando a impossibilidade de se obter tais dados no momento;

- A segunda abordagem, por sua vez, consistiu em montar um mapa de rede, que é uma imagem criada via linguagem PHP, utilizando-se da biblioteca gráfica GD, biblioteca esta atuando em conjunto com os dados obtidos do protocolo Spanning Tree via protocolo SNMP, a fim de exibir graficamente como os equipamentos de rede estão conectados uns aos outros, indicando as portas utilizadas nas conexões, os custos de conexão, os estados atuais destas portas e os endereços IP dos equipamentos.

  Dentro disto, deve-se observar a necessidade de calcular, dentro desta rotina de geração de imagens, como os desenhos serão feitos, em função da quantidade de equipamentos conectados a um dado equipamento qualquer.

  A figura 3.49 apresenta as funcionalidades do sistema relacionadas à monitoração via protocolo Spanning Tree (textual e gráfica).

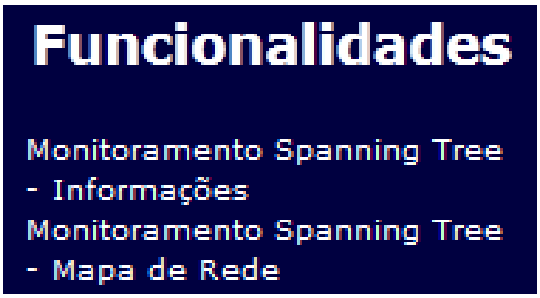


Figura 3.49: Funcionalidades para monitoramento Spanning Tree

3.4.2.4 Laboratório

Além das abordagens de implementação de monitoramento do protocolo Spanning Tree listadas ao longo desta sub-seção 3.4.2, também foi implemen-

tada uma simulação com 3 *switches* Cisco ME 3400E conectados entre si, de modo a formar uma rede em anel.

Em função da atual topologia de rede existente no *backbone* da RedeRIO, onde não há nenhuma redundância, levando a existir somente um caminho possível entre dois equipamentos quaisquer, conectados a esta rede, concluiu-se que era necessário testar a solução WEB em uma rede com possibilidade de haver mais de um caminho possível entre dois equipamentos, para garantir que a solução funcionaria adequadamente neste caso também.

Basicamente, esta etapa consiste na realização do mesmo experimento realizado no aplicativo Packet Tracer, mas agora utilizando equipamentos reais de rede.

Conforme mencionado no começo deste item, foram utilizados 3 *switches* Cisco ME 3400E, conectados entre si, de modo a formar uma rede com topologia em anel. As conexões entre eles foram feitas de modo a seguir o modelo apresentado na figura 3.50 a seguir:

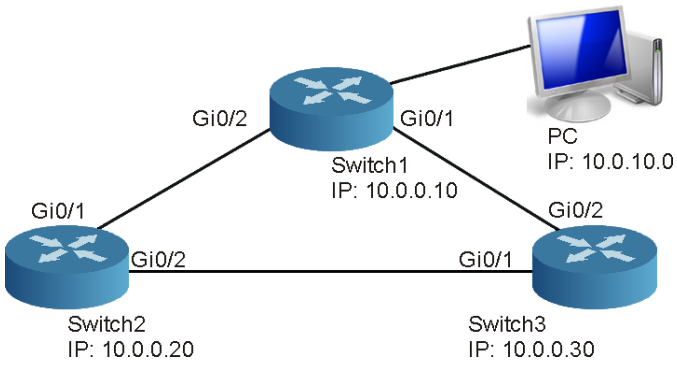


Figura 3.50: Modelo de rede utilizado na simulação em laboratório

A figura 3.51 mostra os *switches* Cisco ME 3400E propriamente ditos, conectados em rede.

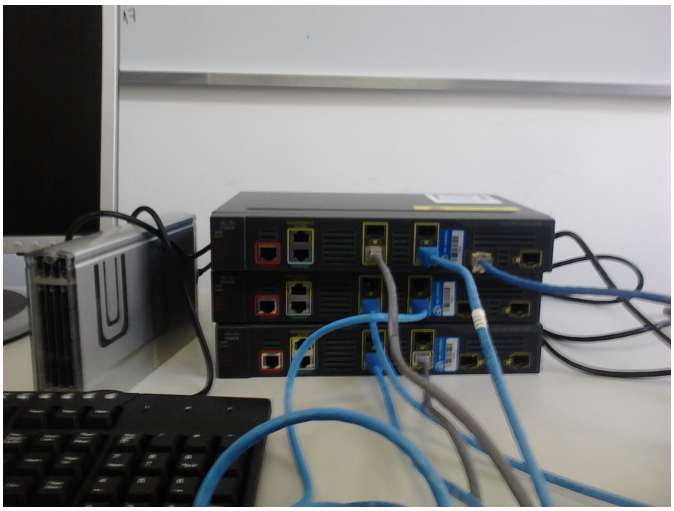


Figura 3.51: Switches Cisco ME 3400E

Capítulo 4

Resultados

Neste capítulo, serão abordados os resultados práticos obtidos ao longo do desenvolvimento do sistema descrito nesta dissertação.

4.1 Etapa 1

O ponto principal da etapa 1, que foi a implementação do protocolo SNMP para monitoramento de equipamentos de rede, se deu sob duas abordagens distintas.

- Primeiro, executou-se a simulação do aplicativo *snmpwalk*, via *prompt*, retornando todas as estatísticas de um dado equipamento de rede, passado como parâmetro através de seu endereço IP;

- Na outra abordagem, através de um aplicativo de gerência e monitoramento de redes, que utiliza o aplicativo *snmpwalk* para levantar todas as interfaces existentes de cada equipamento, bem como os dados técnicos e status das interfaces do equipamento monitorado.

4.1.1 Implementação via *prompt*

Para ilustrar esta abordagem, serão apresentadas duas formas de execução do aplicativo *snmpwalk*:

- Uma, a partir de um dado endereço IP, retornando todos os elementos de rede conectados ao *host* associado a este endereço;

- E outra, mais específica, onde, a partir de um dado *host*, busca-se um atributo específico de um dado elemento conectado a este *host*.

Por medida de simplicidade, esta execução será rodada a partir de um *host* de um computador pessoal, possuidor de poucos elementos de rede conectados.

A seguir, alguns resultados obtidos na primeira execução descrita nesta seção:

- A variável *sysDescr* (.1.3.6.1.2.1.1.1) retornou a descrição do equipamento gerenciado, incluindo o sistema operacional e a data de início da operação do mesmo;

- A variável *sysName* (.1.3.6.1.2.1.1.5) retornou o nome admininstrativo dado ao sistema operacional do equipamento gerenciado;

- A variável *sysORLastChange* (.1.3.6.1.2.1.1.8) retornou o tempo transcorrido desde a última modificação ocorrida em alguma instância do equipamento;

- A variável *sysORID.1* (.1.3.6.1.2.1.1.9.1.2.1) retornou um dos identificadores de funcionalidades que o equipamento em questão é capaz de realizar. Neste caso, o identificador relativo à base de dados MIB;

- A variável *sysORDescr.1* (.1.3.6.1.2.1.1.9.1.3.1) retornou uma das descrições de funcionalidades que o equipamento em questão é capaz de realizar. Neste caso, uma referência à MIB.

Uma forma possível de utilizar o aplicativo *snmpwalk*, conforme descrito na primeira execução tem a seguinte forma:

snmpwalk -v 2c -c public localhost

Forma esta que apresentou o resultado abaixo, quando utilizada:

root@antonio-desktop:~# snmpwalk -v 2c -c public localhost
SNMPv2-MIB::sysDescr.0 = STRING: Linux antonio-desktop 2.6.31-14-generic #48-Ubuntu SMP Fri Oct 16 14:04:26 UTC 2009 i686
(...)
SNMPv2-MIB::sysName.0 = STRING: antonio-desktop
(...)
SNMPv2-MIB::sysORLastChange.0 = Timeticks: (2) 0:00:00.02
SNMPv2-MIB::sysORID.1 = OID: SNMP-FRAMEWORK-MIB::s

nmpFrameworkMIBCompliance
(...)
SNMPv2-MIB::sysORDescr.1 = STRING: The SNMP Management Architecture MIB.
(...)

Na segunda execução, são obtidas as interfaces conectadas a um dado equipamento - cujos endereço IP e *community* reais foram omitidos por questões de segurança - através do aplicativo *snmpwalk*.

É interessante observar nesta execução, que o aplicativo *snmpwalk*, ao listar as interfaces de um equipamento, consegue detectar tanto as interfaces reais, ou seja, as que possuem estrutura física, como as interfaces do tipo Serial, FastEthernet ou Ethernet; quanto as interfaces virtuais, que possuem somente estrutura lógica, como as interfaces do tipo Null, Loopback ou Multilink.

A execução do aplicativo neste caso tem a seguinte forma:

snmpwalk -v 2c -c public 10.0.94.244 .1.3.6.1.2.1.2.2.1.2

Que apresentou o resultado abaixo:

root@antonio-desktop:~# snmpwalk -v 2c -c public 10.0.94.244 .1.3.6.1.2.1.2.2.1.2
IF-MIB::ifDescr.1 = STRING: Serial3/0
(...)
IF-MIB::ifDescr.3 = STRING: FastEthernet0/0
(...)
IF-MIB::ifDescr.12 = STRING: Ethernet2/0
(...)
IF-MIB::ifDescr.24 = STRING: Null0
IF-MIB::ifDescr.25 = STRING: Loopback0
(...)
IF-MIB::ifDescr.27 = STRING: Multilink100

4.1.2 Implementação via aplicativo de gerência WEB

Na implementação via aplicativo de gerência WEB, podemos ver, na prática, como a utilização do protocolo SNMP viabiliza as atividades de monitoração e gerência de rede, com o uso de uma interface WEB. Nisto, pode-se destacar a possibilidade de monitorar estatísticas como o tráfego de rede que passa pela interface; ou ainda verificar o status das interfaces do equipamento monitorado, as quantidades de pacotes recebidos/enviados, funcionamento da interface propriamente dita etc.

A seguir, alguns valores obtidos nesta implementação:

- A variável *ifSpeed* (.1.3.6.1.2.1.2.2.1.5.24) retornou a taxa estimada de velocidade de banda da interface 24 (FastEthernet0/24) do equipamento, em *bits* por segundo;
- A variável *ifPhysAddress* (.1.3.6.1.2.1.2.2.1.6.24) retornou o endereço físico (MAC Address) da interface 24 (FastEthernet0/24) do equipamento;
- A variável *ifAdminStatus* (.1.3.6.1.2.1.2.2.1.7.24) retornou o status operacional esperado para a interface 24 (FastEthernet0/24) do equipamento, neste caso, o status está com o valor 1 (*up*), ou seja, espera-se que a interface esteja operando normalmente;
- A variável *ifLastChange* (.1.3.6.1.2.1.2.2.1.9.24) retornou o intervalo de tempo em que a interface 24 (FastEthernet0/24) do equipamento está ininterruptamente operacional, em milhares de segundos;
- A variável *ifInOctets* (.1.3.6.1.2.1.2.2.1.10.24) retornou o total de *bytes* (octetos) recebidos pela interface 24 (FastEthernet0/24) do equipamento.

Como resultado desta execução, a figura 4.1 apresenta a utilização do aplicativo *snmpwalk* para obter a listagem de interfaces de um dado equipamento, através de busca pela estatística *ifDescr* (.1.3.6.1.2.1.2.2.1.2); e o

elemento MIB *locIfDescr*, ou *ifDescr2* (.1.3.6.1.4.1.9.2.2.1.1.28), que retorna o nome da instituição, cadastrado diretamente a cada interface do equipamento.

Outro resultado pode ser visualizado na figura 4.2, onde o aplicativo *snmpwalk* é empregado para obter as estatísticas, status, etc. de uma das interfaces propriamente ditas.

Interfaces:

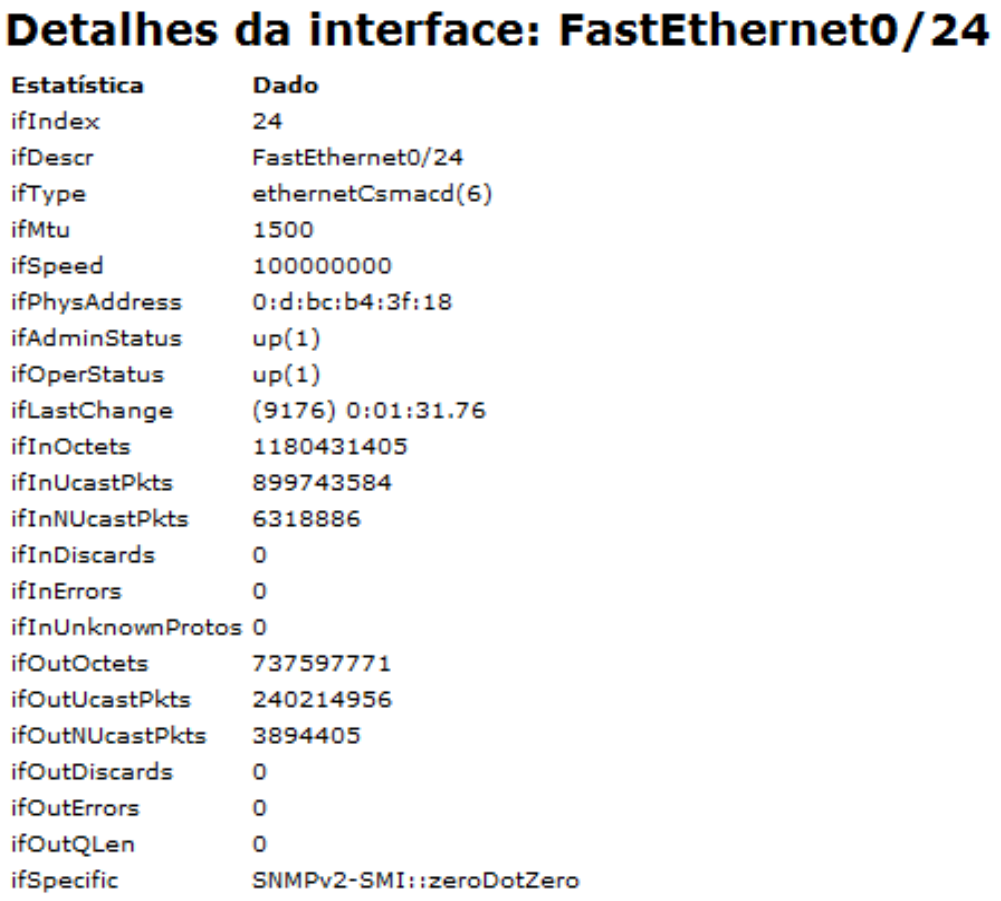
#	Nome	ifDescr2	Instituição
1	FastEthernet0/1		
18	FastEthernet0/18		
20	FastEthernet0/20		
21	FastEthernet0/21		
23	FastEthernet0/23		
24	FastEthernet0/24	UP-Link 7600	L3
25	GigabitEthernet0/1		
26	GigabitEthernet0/2		
27	Null0		
28	Vlan1		
29	Vlan94	VLAN 94	

Figura 4.1: Lista de interfaces obtidas via aplicativo *snmpwalk*

Cabe observar que, em ambas as abordagens onde o aplicativo *snmpwalk* foi utilizado; seja via linha de comando, ou via aplicação WEB; o aplicativo sempre retornará os mesmos resultados, uma vez que as requisições sejam iguais, variando apenas a forma como os mesmos estarão dispostos na interface de saída.

4.1.3 Implementação de gráficos

Outro resultado da etapa 1 que deve ser mencionado é o que diz respeito à geração de gráficos de monitoramento. Conforme apresentado nas seções 3.1 e 3.4.1, serão apresentados gráficos de monitoramento para cada interface de rede, de cada equipamento monitorado; bem como serão gerados também gráficos de monitoramento de CPU e temperatura de cada equipamento, desde que tal dispositivo esteja configurado para exibir um, ou ambos os monitoramentos especificados.



Detalhes da interface: FastEthernet0/24

Estatística	Dado
ifIndex	24
ifDescr	FastEthernet0/24
ifType	ethernetCsmacd(6)
ifMtu	1500
ifSpeed	100000000
ifPhysAddress	0:d:bc:b4:3f:18
ifAdminStatus	up(1)
ifOperStatus	up(1)
ifLastChange	(9176) 0:01:31.76
ifInOctets	1180431405
ifInUcastPkts	899743584
ifInNUcastPkts	6318886
ifInDiscards	0
ifInErrors	0
ifInUnknownProtos	0
ifOutOctets	737597771
ifOutUcastPkts	240214956
ifOutNUcastPkts	3894405
ifOutDiscards	0
ifOutErrors	0
ifOutQLen	0
ifSpecific	SNMPv2-SMI::zeroDotZero

Figura 4.2: Dados de interface obtidos via aplicativo *snmpwalk*

A exibição de tais dados será feita em quatro níveis de periodicidade: diário, semanal, mensal e anual.

Conforme mencionado na seção 3.3.2, o aplicativo MRTG é a ferramenta responsável por fazer a aquisição de dados dos equipamentos de rede, bem como gerar os gráficos que consolidam estes dados, de forma a se poder avaliar o comportamento de cada equipamento.

Antes disto, cogitou-se a possibilidade de utilizar uma solução baseada na linguagem JavaScript para gerar estes gráficos, hipótese esta abandonada, em razão da lentidão observada no instante em que a página é gerada, uma vez que todos os pontos de cada gráfico são desenhados em tempo de execução. Com a solução baseada no aplicativo MRTG, estes gráficos

são gerados separadamente das páginas do sistema propriamente ditas, que apenas carregam as imagens previamente confeccionadas.

A figura 4.3 apresenta um exemplo de gràfico de monitoramento de tráfego. Neste exemplo, podemos ver o gráfico mensal (mês de Outubro de 2012) da interface GigabitEthernet4/17 do equipamento REDERIO-GIGA, interface esta associada à instituiçao CBPF.

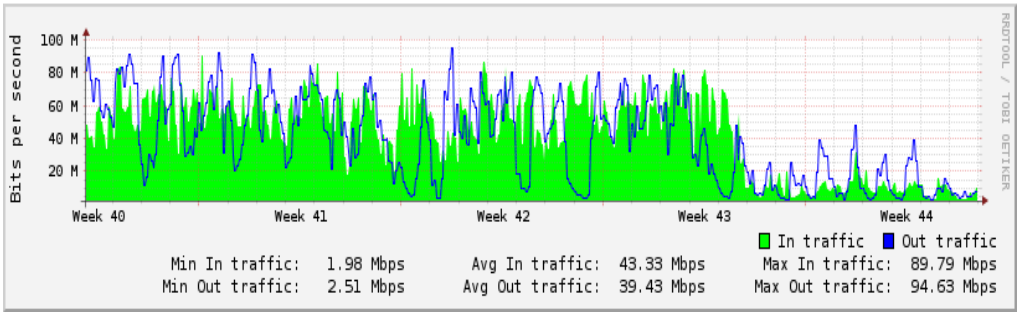


Figura 4.3: Exemplo de gráfico de monitoramento de tráfego

4.2 Etapa 2

4.2.1 Simulação com uso do aplicativo Packet Tracer

Através do conjunto de passos listado na sub-seção 3.4.2, observaram-se os seguintes resultados:

- Para a criação da rede, conforme descrito nos três primeiros passos, serão realizados os comandos IOS abaixo:

  1. A configuração dos PC's é realizada através da interface disponibilizada pelo aplicativo Packet Tracer, conforme pode ser visto na figura 4.4;
  2. Acesso ao *switch* Switch0 (2950T-24 Switch0):
     Switch>enable: Habilitar a edição de configurações do *switch* Switch0.
     Switch#config terminal: Acesso à edição de configurações do

Figura 4.4: Tela de configuração de um PC via Packet Tracer

switch Switch0.

Switch(config)#interface vlan 1: Acesso à edição da interface vlan 1.

Switch(config-if)#ip address 192.168.2.1 255.255.255.0: Configura o endereço IP do *switch* (192.168.2.1), e sua máscara de rede (255.255.255.0).

Switch(config-if)#no shutdown: Ativa a interface do *switch*.

Switch(config-if)#exit: Sai da seção de configurações da interface.

Switch(config)#int range Fa 0/1-24: Seleciona todas as interfaces FastEthernet, de Fa0/1 até Fa0/24.

Switch(config-if-range)#no shutdown: Ativa as interfaces do *switch*.

Switch(config-if-range)#exit: Sai da seção de configurações das interfaces.

3. O procedimento anterior deve ser repetido para todos os *switches* da rede, com os endereços IP e máscaras de rede listados na tabela 3.2.

Tabela 4.1: Endereços físicos e prioridades da simulação

	Endereço físico	Prioridade
Switch0	0060.2F50.B1BC	32769
Switch1	0001.975E.13B2	32769
Switch2	0001.C9D6.BB01	32769
Switch3	0001.C7B1.870E	32769
Switch4	0060.5CA1.22EC	32769

- Ao final destes passos, os *switches* deverão apresentar as prioridades e endereços físicos (MAC Address) apresentados na tabela 4.1 a seguir;

- Para verificar as configurações do protocolo Spanning Tree para cada *switch*, devem ser executados os seguintes comandos:

  Switch>enable: Habilitar a edição de configurações do *switch* Switch0.
  Switch#show spanning-tree: Verificar as configurações do protocolo Spanning Tree para o *switch* selecionado.

  O resultado do comando **show spanning-tree** acima apresenta inicialmente o nome da sub-rede VLAN vinculada ao *switch*. Em seguida, são apresentados os dados do equipamento eleito pelo protocolo Spanning Tree como raiz da Árvore de Custo Mínimo, seguido pelos dados do equipamento corrente, com o detalhamento de estados das interfaces de rede que estão sendo utilizadas pelo protocolo.

  Para o *switch* Switch0, a execução do comando **show spanning-tree** apresentou o resultado a seguir:

VLAN0001
Spanning tree enabled protocol ieee
Root ID Priority 32769
Address 0001.975E.13B2
Cost 19
Port 1(FastEthernet0/1)
Hello Time 2 sec Max Age 20 sec Forward D

elay 15 sec

Bridge ID Priority 32769 (priority 32768 sys-id-ext 1)
Address 0060.2F50.B1BC
Hello Time 2 sec Max Age 20 sec Forward D
elay 15 sec
Aging Time 20

Interface	**Role Sts Cost**	**Prio.Nbr Type**
Fa0/1	**Root FWD 19**	**128.1 P2p**
Fa0/2	**Desg FWD 19**	**128.2 P2p**
Fa0/3	**Altn BLK 19**	**128.3 P2p**

Conforme os resultados do comando **show spanning-tree**, e mostrado na imagem 4.5, o *switch* Switch1 foi eleito como o nó raiz da Árvore de Custo Mínimo, pelo protocolo Spanning Tree, árvore esta destacada em vermelho.

Também pode-se observar que todas as conexões escolhidas para a árvore apresentam as interfaces dos dois *switches* na cor verde, indicando que os estados delas estão congurados como *Forwarding*, ou seja, disponíveis para envio/recebimento de dados.

Por outro lado, as interfaces na cor laranja (FastEthernet0/3, do *switch* Switch0; FastEthernet0/2, do *switch* Switch2; e FastEthernet0/2, do *switch* Switch4) estão configuradas com estado *Blocking*, indisponíveis para envio/recebimento de dados;

- Uma ferramenta útil disponibilizada pelo aplicativo Packet Tracer é o comando **?**, que, quando utilizado após uma solicitação qualquer, pode disponibilizar a relação de funcionalidades que podem ser utilizadas em conjunto com o primeiro comando, jutamente com uma explicação sobre estas. Por exemplo, ao se executar **show spanning-tree ?**, será

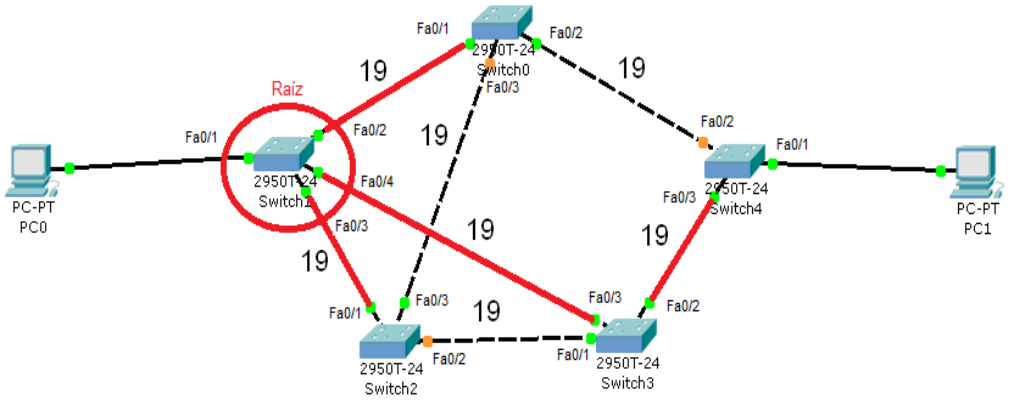


Figura 4.5: Árvore de Custo Mínimo inicial da simulação Packet Tracer

obtido o resultado a seguir:

Switch#show spanning-tree ?

active	**Report on active interfaces only**
detail	**Detailed information**
inconsistentports	**Show inconsistent ports**
interface	**Spanning Tree interface status and confi**
guration	
summary	**Summary of port states**
vlan	**VLAN Switch Spanning Trees**
<cr>	

- Para realizar o próximo passo, que consiste em atribuir o valor 4096 à prioridade do *switch* Switch0, deve se efetuar o procedimento a seguir (salientando que o valor de prioridade deve ser um múltiplo de 4096, com valor máximo de 61440):

  1. **Switch>enable**: Habilitar a edição de configurações do *switch* Switch0;

  2. **Switch#config terminal**: Acesso à edição de configurações do

switch Switch0;

3. **Switch(config)#spanning-tree vlan 1 priority 4096**: Acesso à edição da interface vlan 1, atribuindo ao *switch* prioridade de valor 4096.

- Cabe observar que o valor da prioridade atribuído ao *switch* será sempre igual ao valor atribuído mais 1. Ao final deste passo, a Árvore de Custo Mínimo da rede será redesenhada conforme exposto na imagem 4.6 a seguir, onde a redução do valor da prioridade do *switch* Switch0 faz com que este equipamento passe a ser a raíz da árvore:

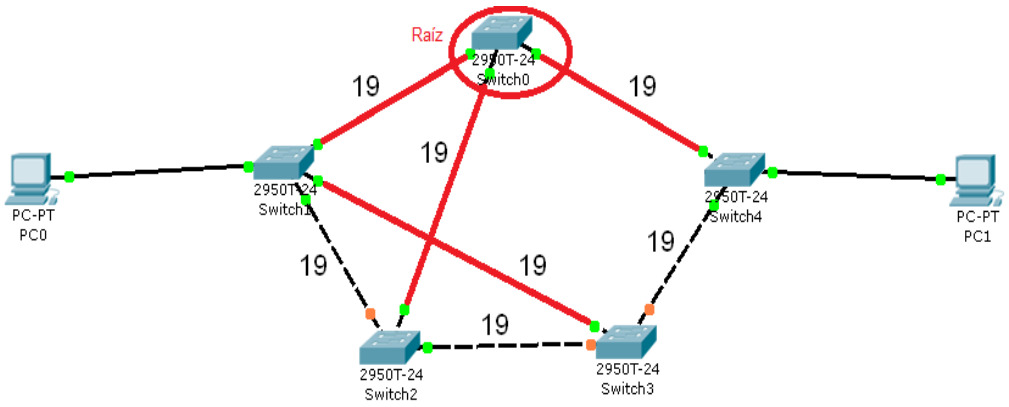


Figura 4.6: Árvore de Custo Mínimo da simulação Packet Tracer

- A mudança de raiz exposta acima ilustra como o protocolo Spanning Tree determina qual equipamento de rede será a raíz da Árvore de Custo Mínimo. O primeiro critério para tomada de decisão consiste em verificar qual dos equipamentos de rede está configurado com o menor valor de prioridade.

  Se um equipamento possuir um valor de prioridade menor que todos os outros, este equipamento será eleito como raíz da Árvore de Custo Mínimo; caso dois ou mais equipamentos possuam o menor valor de prioridade, o segundo critério será eleger como raiz aquele que possuir

o menor valor de endereço físico (MAC Address), o que remete à tabela 4.1, onde todos os *switches* possuem o mesmo valor de prioridade, o que leva à escolha do *switch* Switch1, que possui menor valor de endereço físico, como raiz, o que pode ser verificado na figura 4.5.

Contudo, após a mudança do valor de prioridade do *switch* Switch1, este dispositivo passou a ter menor prioridade que todos os outros equipamentos integrantes da rede, passando então a ser escolhido como raiz, conforme exposto na figura 4.6;

- Para realizar o próximo passo, que consiste em atribuir o valor 36864 à prioridade do *switch* Switch0, deve se efetuar o procedimento a seguir:

  1. **Switch>enable**: Habilitar a edição de configurações do *switch* Switch0;

  2. **Switch#config terminal**: Acesso à edição de configurações do *switch* Switch0;

  3. **Switch(config)#spanning-tree vlan 1 priority 36864**: Acesso à edição da interface vlan 1, atribuindo ao *switch* prioridade de valor 36864.

- Ao final deste passo, a Árvore de Custo Mínimo da rede será redesenhada conforme exposto na imagem 4.5, onde o aumento do valor da prioridade do *switch* Switch0 faz com que todos os outros equipamentos possuam prioridades menores que a deste equipamento. Como os demais equipamentos possuem o mesmo valor de prioridade, será escolhido como raiz o equipamento com o menor endereço físico, neste caso, o *switch* Switch1;

- Para realizar o próximo passo, que consiste em fazer do *switch* Switch0 a raiz primária da Árvore de Custo Mínimo, deve se efetuar o procedimento a seguir:

1. **Switch>enable**: Habilitar a edição de configurações do *switch* Switch0;

2. **Switch#config terminal**: Acesso à edição de configurações do *switch* Switch0;

3. **Switch(config)#spanning-tree vlan 1 root primary**: Configura o *switch* como raiz primária.

- Com isto, a prioridade do *switch* Switch0 será automaticamente alterada para o valor 28673. Assim, a Árvore de Custo Mínimo da rede será redesenhada conforme exposto na imagem 4.6, onde a alteração realizada no *switch* Switch0 faz com que este equipamento possua o menor valor de prioridade dentro desta rede, fazendo dele a raiz da Árvore de Custo Mínimo;

- Para realizar o próximo passo, que consiste em atribuir o valor 32768 à prioridade do *switch* Switch0, deve se efetuar o procedimento a seguir:

  1. **Switch>enable**: Habilitar a edição de configurações do *switch* Switch0;

  2. **Switch#config terminal**: Acesso à edição de configurações do *switch* Switch0;

  3. **Switch(config)#spanning-tree vlan 1 priority 32768**: Acesso à edição da interface vlan 1, atribuindo ao *switch* prioridade de valor 32768.

- Com isto, a Árvore de Custo Mínimo da rede será redesenhada conforme exposto na imagem 4.5, onde a mudança realizada no *switch* Switch0 faz com que todos os equipamentos possuam valores de prioridade iguais

entre si. Assim, será escolhido como raiz o equipamento com o menor endereço físico, neste caso, o *switch* Switch1;

- Para realizar o próximo passo, que consiste em desabilitar o *switch* Switch1, deve se efetuar o procedimento a seguir:

  1. **Switch>enable**: Habilitar a edição de configurações do *switch* Switch1;

  2. **Switch#config terminal**: Acesso à edição de configurações do *switch* Switch1;

  3. **Switch(config)#int range fa0/1-24**: Configura as interfaces de rede, da FastEthernet0/1 até a FastEthernet0/24, do *switch* para poder modificar todas elas simultaneamente;

  4. **Switch(config-if-range)#shutdown**: Desliga as interfaces de rede selecionadas anteriormente, do *switch* Switch1.

- Com isto, a Árvore de Custo Mínimo da rede será redesenhada conforme exposto na imagem 4.7, onde a exclusão do *switch* Switch1 faz com que seja necessário que um outro equipamento seja escolhido como raiz da Árvore de Custo Mínimo.

  Como todos os demais equipamentos possuem valores de prioridade iguais entre si, será escolhido como raiz o equipamento com o menor endereço físico, neste caso, o *switch* Switch2. Como o *switch* Switch1 foi desabilitado, todas as suas interfaces, bem como as interfaces de outros equipamentos que estejam ligadas ao *switch* Switch1, passarão a aparecer na cor vermelha, indicando que as mesmas não estão funcionando no momento;

- Para realizar o próximo passo, que consiste em habilitar o *switch* Switch1, deve se efetuar o procedimento a seguir:

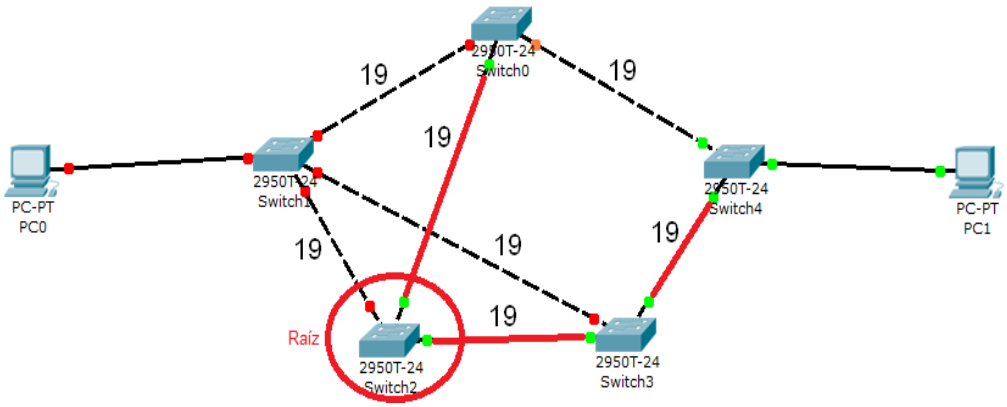


Figura 4.7: Árvore de Custo Mínimo da simulação Packet Tracer

1. **Switch>enable**: Habilitar a edição de configurações do *switch* Switch1;

2. **Switch#config terminal**: Acesso à edição de configurações do *switch* Switch1;

3. **Switch(config)#int range fa0/1-24**: Configura as interfaces de rede, da FastEthernet0/1 até a FastEthernet0/24, do *switch* para poder modificar todas elas simultaneamente;

4. **Switch(config-if-range)#no shutdown**: Liga as interfaces de rede selecionadas anteriormente, do *switch* Switch1.

- Com isto, a Árvore de Custo Mínimo da rede será redesenhada conforme exposto na imagem 4.5, onde todos os equipamentos possuam valores de prioridade iguais entre si. Assim, será escolhido como raiz o equipamento com o menor endereço físico, neste caso, o *switch* Switch1;

- Para realizar o próximo passo, que consiste em fazer do *switch* Switch1 a raiz primária, o *switch* Switch4 a raiz secundária da Árvore de Custo Mínimo, e em seguida, desligar o *switch* Switch1, deve se efetuar o

procedimento a seguir:

No *switch* Switch1:

1. **Switch>enable**: Habilitar a edição de configurações do *switch* Switch1;

2. **Switch#config terminal**: Acesso à edição de configurações do *switch* Switch1;

3. **Switch(config)#spanning-tree vlan 1 root primary**: Configura o *switch* como raiz primária da Árvore de Custo Mínimo.

Com isto, a prioridade do *switch* Switch1 será automaticamente alterada para o valor 24577.

No *switch* Switch4:

1. **Switch>enable**: Habilitar a edição de configurações do *switch* Switch4;

2. **Switch#config terminal**: Acesso à edição de configurações do *switch* Switch4;

3. **Switch(config)#spanning-tree vlan 1 root secondary**: Configura o *switch* como raiz secundária da Árvore de Custo Mínimo.

Com isto, a prioridade do *switch* Switch4 será automaticamente alterada para o valor 28673. Caso a raiz secundária seja definida sem definição de raiz primária, esta raiz secundária será tratada como raiz primária.

Após configurar o *switch* Switch4 da Árvore de Custo Mínimo, desligar novamente o *switch* Switch1, conforme explicado anteriormente, nesta seção.

- Com isto, a Árvore de Custo Mínimo da rede será redesenhada conforme exposto na imagem 4.8 a seguir, onde a exclusão do *switch* Switch1 faz com que seja necessário que um outro equipamento seja escolhido como raiz da Árvore de Custo Mínimo.

  Como o *switch* Switch4 havia sido configurado como raiz secundária, qualquer circunstância que fizesse com que a raiz primária da rede (Switch1) deixasse de ser a raiz levaria o *switch* Switch4 a ser a nova raiz. A partir do momento em que o *switch* Switch1 foi desabilitado, todas as suas interfaces, bem como as interfaces de outros equipamentos que estejam ligadas ao *switch* Switch1, passarão a aparecer na cor vermelha, indicando que as mesmas não estão funcionando no momento;

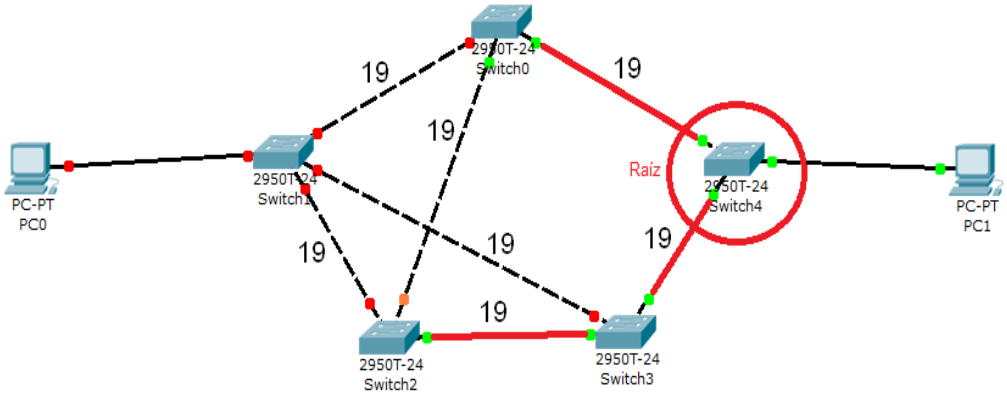


Figura 4.8: Árvore de Custo Mínimo da simulação Packet Tracer

- Para realizar o próximo passo, que consiste em religar novamente o *switch* Switch1, deve se efetuar o procedimento a seguir:

  1. **Switch>enable**: Habilitar a edição de configurações do *switch* Switch1;

2. **Switch#config terminal**: Acesso à edição de configurações do *switch* Switch1;

3. **Switch(config)#int range fa0/1-24**: Configura as interfaces de rede, da FastEthernet0/1 até a FastEthernet0/24, do *switch* para poder modificar todas elas simultaneamente;

4. **Switch(config-if-range)#no shutdown**: Liga as interfaces de rede selecionadas anteriormente, do *switch* Switch1.

- Com isto, a Árvore de Custo Mínimo da rede será redesenhada conforme exposto na imagem 4.5, onde o *switch* Switch1, por ter sido configurado como raiz primária, será escolhido como raiz;

- Para realizar o próximo passo, que consiste em configurar a prioridade do *switch* Switch4 com o valor 32768, e, em seguida, desabilitar a interface de rede FastEthernet0/4 do *switch* Switch1, deve se efetuar o procedimento a seguir:

  No *switch* Switch4:

1. **Switch>enable**: Habilitar a edição de configurações do *switch* Switch4;

2. **Switch#config terminal**: Acesso à edição de configurações do *switch* Switch4;

3. **Switch(config)#spanning-tree vlan 1 priority 32768**: Acesso à edição da interface vlan 1, atribuindo ao *switch* prioridade de

valor 32768.

No *switch* Switch1:

1. **Switch>enable**: Habilitar a edição de configurações do *switch* Switch1;

2. **Switch#configure terminal**: Acesso à edição de configurações de Switch1;

3. **Switch(config)#interface fa0/4**: Configura a interface de rede FastEthernet0/4, do *switch* para se poder modificá-la;

4. **Switch(config-if)#shutdown**: Desliga a interface de rede selecionada anteriormente, do *switch* Switch1.

- Com isto, a Árvore de Custo Mínimo da rede será redesenhada conforme exposto na imagem 4.9 a seguir, onde a exclusão da interface de rede FastEthernet0/4 do *switch* Switch1 faz com que seja necessário que o protocolo Spanning Tree recalcule novos caminhos ótimos para alcançar os *switches* Switch3 e Switch4, os quais eram alcançados através da interface de rede FastEthernet0/4 do *switch* Switch1;
- Para realizar o próximo passo, que consiste em habilitar novamente a interface de rede FastEthernet0/4 do *switch* Switch1, deve se efetuar o procedimento a seguir:

1. **Switch>enable**: Habilitar a edição de configurações do *switch* Switch1;

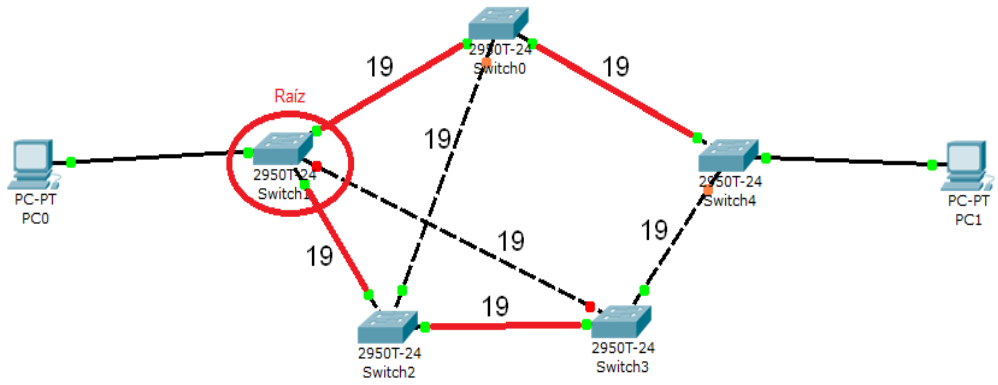


Figura 4.9: Árvore de Custo Mínimo da simulação Packet Tracer

2. **Switch#configure terminal**: Acesso à edição de configurações de Switch1;

3. **Switch(config)#interface fa0/4**: Configura a interface de rede FastEthernet0/4, do *switch* para se poder modificá-la;

4. **Switch(config-if)#no shutdown**: Liga a interface de rede selecionada anteriormente, do *switch* Switch1.

- Com isto, a Árvore de Custo Mínimo da rede será redesenhada conforme exposto na imagem 4.5;

- Para realizar o próximo passo, que consiste em configurar a velocidade da interface de rede FastEthernet0/4 do *switch* Switch1 para 10 Mbps, deve se efetuar o procedimento a seguir:

1. **Switch>enable**: Habilitar a edição de configurações do *switch* Switch1;

2. **Switch#configure terminal**: Acesso à edição de configurações de Switch1;

3. **Switch(config)#interface fa0/4**: Configura a interface de rede FastEthernet0/4, do *switch* para se poder modificá-la;

4. **Switch(config-if)#speed 10**: Configura a velocidade (largura de banda disponível) da interface de rede FastEthernet0/4, do *switch* Switch1, em 10Mbps.

- Para que a mudança de velocidade, e consequentemente de custo de conexão seja efetuada de fato, é necessário fazer o mesmo procedimento acima com a interface de rede que está conectada à interface FastEthernet0/4 do *switch* Switch1, neste caso, a interface FastEthernet0/3, do *switch* Switch3. Uma vez concluído este procedimento, será obtida uma Árvore de Custo Mínimo conforme à da figura 4.10 a seguir:

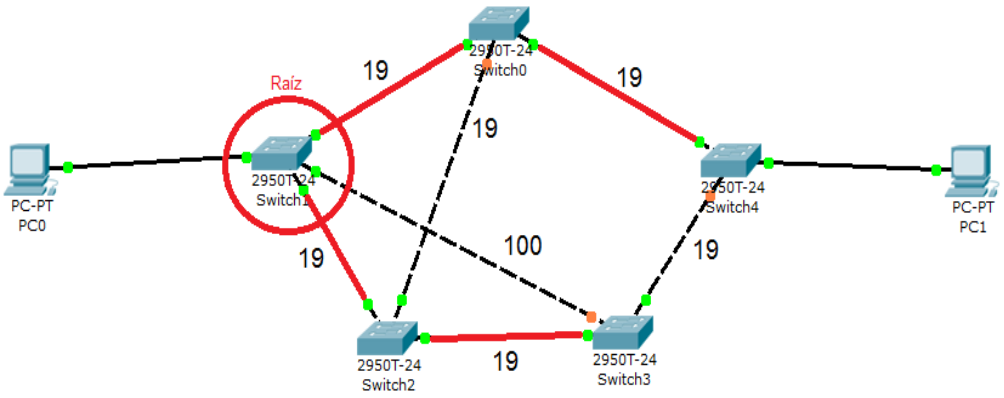


Figura 4.10: Árvore de Custo Mínimo da simulação Packet Tracer

- Conforme mostrado na figura 4.10, em comparação à figura 4.5, a mudança na largura de banda das interfaces de rede FastEthernet0/4 (*switch* Switch1) e FastEthernet0/3 (*switch* Switch3) fez com que o custo de rede aumentasse de 10 (largura de banda 100Mbps, conforme

tabela 3.1) para 100 (largura de banda 10Mbps). Este aumento de custo fez com que o protocolo Spanning Tree recalculasse os caminhos ótimos de rede entre os *switches*, passando assim a entender que o melhor caminho entre entre os *switches* Switch1 e Switch3 tornou-se o caminho que inclui o *switch* Switch2, conforme exposto na figura 4.10, de custo total 38, em vez do caminho direto entre os *switches* Switch1 e Switch3, agora de custo total 100.

Conforme apresentado pela simulação descrita nesta seção, o protocolo Spanning Tree opera em *switches* (também em *bridges*). Estes dispositivos trocam informações entre si através de mensagens *Bridge Protocol Data Units* - BPDU. Tal como mostrado na sub-seção 2.2.6.4, estas mensagens contêm dados de configuração e incluem o número identificador de cada equipamento [66].

Para a operação do protocolo Spanning Tree, inicialmente, um dos equipamentos conectados à rede é eleito como raiz da rede. Este *switch*-raiz torna-se assim o foco da rede, sendo que as escolhas de porta-raiz, porta-designada e não-designada para cada equipamento e todas as atualizações da rede são vistas sempre tendo o *switch*-raiz como referência.

Para elegê-lo, serão utilizados os seguintes critérios [66]:

- Menor valor de prioridade (o valor padrão é de 32768);
- Menor valor de endereço físico (MAC Address) (por exemplo, 0000.aaaa.bbbb.cccc.dddd.eeee representa um valor de endereço menor que 0000.aaab.bbbb.cccc.dddd.eeee).

As portas (interfaces de rede) podem possuir os seguintes status [66]:

- **Porta-raiz (Root port)**: Porta de um *switch* que representa o caminho com menor custo possível entre o *switch* em questão e o equipamento raiz. Sempre se econtra no estado *forwarding*;
- **Porta designada (Designated port)**: Porta de um *switch* que mantêm conectividade com outros equipamentos de rede, podendo enviar

ou receber mensagens. Quando há redundância de conexões, a porta com maior largura de banda (menor custo) é escolhida. Se as larguras de banda forem iguais, a porta com menor número identificador será escolhida (por exemplo, porta *e1* escolhida, em vez da porta *e8*). Sempre se econtra no estado *forwarding*;

- **Porta não-designada (Non-designated port)**: Porta que está em estado *blocking*. Quando uma porta designada perde a conectividade, assume este status, até que alguma eventual mudança na rede faça com que seu staus seja modificado.

Agora, serão vistos os estados em que as portas dos equipamentos de rede podem se encontrar.

- ***Blocking***: Não encaminha mensagens. Recebe e analisa mensagens BPDU. Ao se ligar um *switch*, este inicia seu funcionamento, estando no estado *blocking*;

- ***Listening***: Não encaminha mensagens, mas recebe e analisa mensagens BPDU, para se certificar de que não haverão *loops* na rede;

- ***Learning***: Não encaminha mensagens, recebe e analisa mensagens BPDU e registra os endereços físicos dos dispositivos diretamente conectados entre si;

- ***Forwarding***: Envia e recebe mensagens. Uma porta neste estado é tida como possuidora do menor custo entre o equipamento onde esta se encontra, e o *switch*-raiz.

Os estados mais usuais em que uma porta pode se encontrar são *Blocking* e *Forwarding*.

4.2.2 Solução utilizada

Conforme observado na seção 3.4.2.3, o protocolo SNMP possui uma base MIB criada para obter valores referentes ao protocolo Spanning Tree

operacional em um dado equipamento. Esta MIB chama-se **dot1dStp** e possui o identificador OID **1.3.6.1.2.1.17.2**.

Graças ao exposto no parágrafo anterior, constatou-se a viabilidade de realizar monitoramento dos equipamentos de rede do *backbone* da RedeRio via Spanning Tree, com o auxílio do protocolo SNMP.

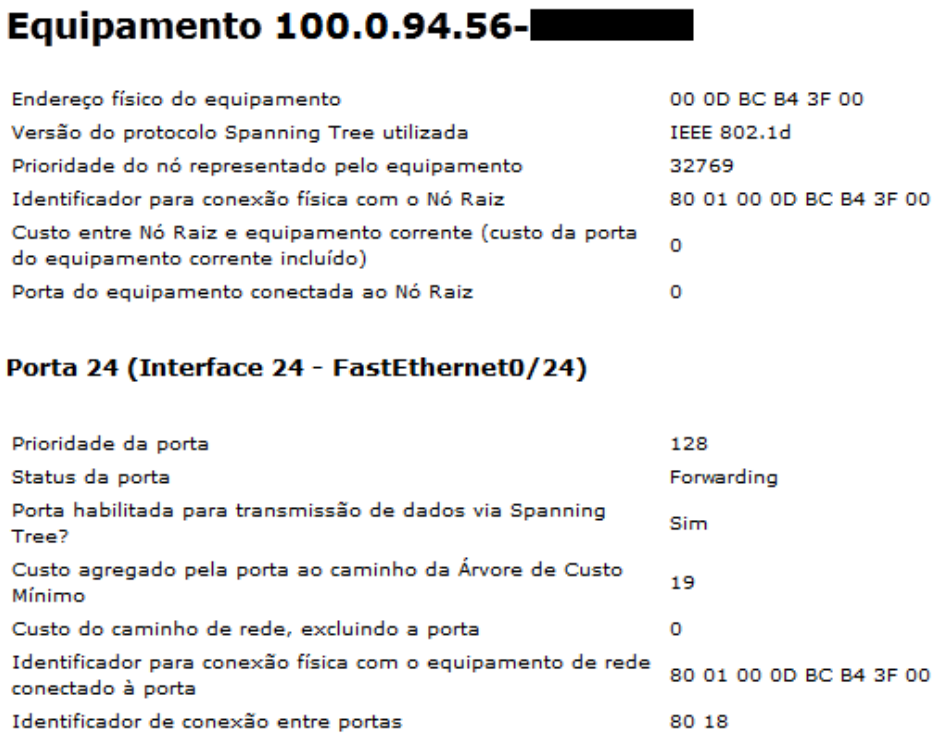
Em termos de resultados obtidos por parte do monitoramento Spanning Tree, a solução de monitoramento WEB descrita nesta dissertação deve ser analisada sob duas abordagens distintas, descritas a seguir, conforme especificadas na seção 3.4.2.3:

- Na primeira abordagem, onde, a partir de um conjunto de equipamentos de rede selecionados pelo usuário, o protocolo SNMP apresenta os dados que estes equipamentos oferecem, em termos de protocolo Spanning Tree, de forma textual, categorizando os dados por equipamento selecionado, e por cada porta (interface de rede) de cada equipamento.

  A figura 4.11 a seguir apresenta um exemplo de como é realizada a aquisição textual dos dados de monitoramento Spanning Tree de um dado equipamento de rede.

  Conforme pode ser observado na figura 4.11, são monitoradas, a nível de equipamento, as seguintes estatísticas:

  - Endereço físico;
  - Versão do protocolo Spanning Tree em operação no dispositivo;
  - Valor de prioridade atribuído ao equipamento, para fins de cálculo de Árvore de Custo Mínimo;
  - Identificador de conexão física da raiz com quem o equipamento monitorado está ligado;
  - Custo de caminho de rede, a partir da raiz até o equipamento monitorado;
  - Porta do equipamento que está se conectando ao nó raiz, ou ao caminho que leva à raiz da rede.



Equipamento 100.0.94.56-[redacted]

Endereço físico do equipamento	00 0D BC B4 3F 00
Versão do protocolo Spanning Tree utilizada	IEEE 802.1d
Prioridade do nó representado pelo equipamento	32769
Identificador para conexão física com o Nó Raiz	80 01 00 0D BC B4 3F 00
Custo entre Nó Raiz e equipamento corrente (custo da porta do equipamento corrente incluído)	0
Porta do equipamento conectada ao Nó Raiz	0

Porta 24 (Interface 24 - FastEthernet0/24)

Prioridade da porta	128
Status da porta	Forwarding
Porta habilitada para transmissão de dados via Spanning Tree?	Sim
Custo agregado pela porta ao caminho da Árvore de Custo Mínimo	19
Custo do caminho de rede, excluindo a porta	0
Identificador para conexão física com o equipamento de rede conectado à porta	80 01 00 0D BC B4 3F 00
Identificador de conexão entre portas	80 18

Figura 4.11: Aquisição de dados no monitoramento Spanning Tree

E, a nível de porta (interface de rede), são monitoradas as seguintes estatísticas:

- Valor de prioridade atribuído à porta, para fins de cálculo de Árvore de Custo Mínimo;
- Status operacional da porta, para envio/recebimento de mensagens;
- Verificação se porta está habilitada para envio/recebimento de mensagens, via Spanning Tree;
- Custo que a porta acrescenta à Árvore de Custo Mínimo;
- Custo agregado à Árvore de Custo Mínimo, excluindo a porta corrente;
- Identificador para conexão ao equipamento com quem a porta está ligada;
- Identificador de conexão em que a porta esteja conectada.

É possível, a partir dos dados textuais coletados, "desenhar" o mapa de rede do conjunto de equipamentos monitorados pelo sistema. Por exemplo, para determinar se um dado equipamento monitorado é a raiz da Árvore de Custo Mínimo, deve-se verificar se seu endereço físico possui a mesma sequência de identificação que o campo identificador para conexão física com o nó raiz, desprezando os quatro primeiros *bytes* (caracteres) deste segundo campo. Caso os valores apresentados sejam os mesmos, para o equipamento em questão, então este é o equipamento raiz da Árvore de Custo Mínimo.

O equipamento **100.0.94.56**, exemplificado na figura 4.11 é um exemplo de equipamento raiz de uma rede monitorada com o protocolo Spanning Tree, pois, como pode ser observado na figura, seus campos de "endereço físico" e de "indetificador para conexão física com o nó raiz" satisfazem o critério de igualdade apresentado no parágrafo anterior.

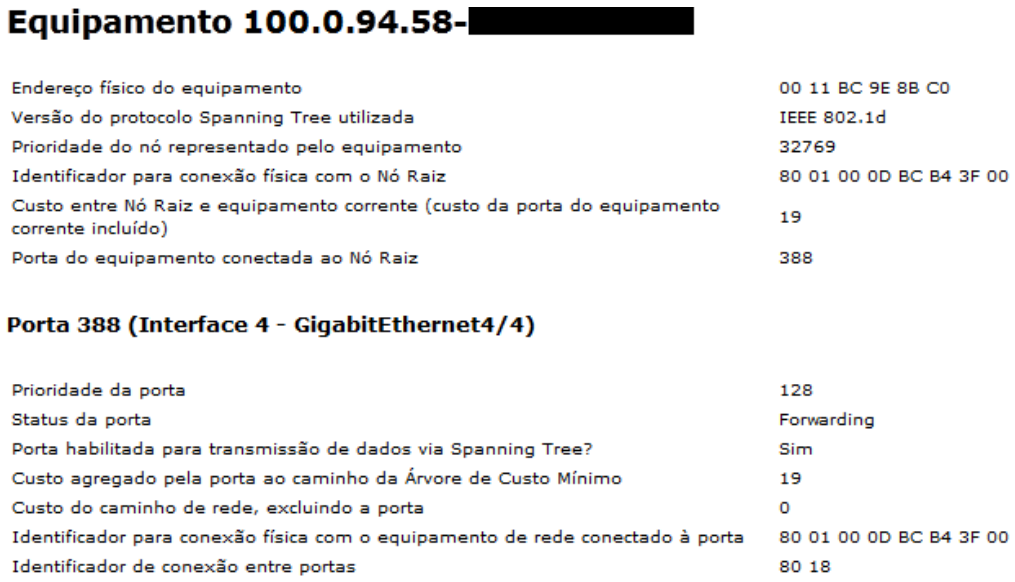
Para verificar quais portas (interfaces de rede) estão conectadas entre si, de modo a conectar dois equipamentos de rede um ao outro, deve-se verificar quais portas apresentam os mesmos valores para os campos "Identificador para conexão física com o equipamento de rede conectado à porta" e "Identificador de conexão entre portas".

Na figura 4.12 a seguir, é possível observar que os campos "Identificador para conexão física com o equipamento de rede conectado à porta" e "Identificador de conexão entre portas" mencionados no parágrafo anterior apresentam, para a porta **388** do equipamento **100.0.94.58**, os mesmos valores observados na porta **24** do equipamento **100.0.94.56**, conforme figura 4.11.

Com isto, podemos concluir que estes dois equipamentos estão conectados entre si, através destas portas (interfaces de rede);

- Na segunda abordagem, é criado um mapa de rede apresentado na forma de uma imagem, desenhada via linguagem PHP, atuando em conjunto com os dados obtidos do protocolo Spanning Tree via SNMP.

  Este mapa apresenta os equipamentos de rede conectados uns aos ou-



Equipamento 100.0.94.58-[redacted]

Endereço físico do equipamento	00 11 BC 9E 8B C0
Versão do protocolo Spanning Tree utilizada	IEEE 802.1d
Prioridade do nó representado pelo equipamento	32769
Identificador para conexão física com o Nó Raiz	80 01 00 0D BC B4 3F 00
Custo entre Nó Raiz e equipamento corrente (custo da porta do equipamento corrente incluído)	19
Porta do equipamento conectada ao Nó Raiz	388

Porta 388 (Interface 4 - GigabitEthernet4/4)

Prioridade da porta	128
Status da porta	Forwarding
Porta habilitada para transmissão de dados via Spanning Tree?	Sim
Custo agregado pela porta ao caminho da Árvore de Custo Mínimo	19
Custo do caminho de rede, excluindo a porta	0
Identificador para conexão física com o equipamento de rede conectado à porta	80 01 00 0D BC B4 3F 00
Identificador de conexão entre portas	80 18

Figura 4.12: Aquisição de dados no monitoramento Spanning Tree

tros, conforme explicitado na descrição textual apresentada no item anterior, iniciando a exibição pelo equipamento raiz, seguido pelo(s) equipamento(s) diretamente conectados a ele, citando as portas (interfaces de rede) utilizadas nas conexões, bem como os status destas portas e o(s) custo(s) desta(s) conexão(ões). Este processo de exibição de equipamentos se repetirá, até que toda a Árvore de Custo Mínimo seja exibida.

É possível também que mais de uma Árvore de Custo Mínimo seja apresentada na imagem. Isto ocorrerá se o conjunto de equipamentos monitorados estiver distribuído em mais de uma rede conexa. Desta forma, o processo descrito no parágrafo anterior será repetido tantas vezes quantas forem necessárias, de modo a exibir todos os equipamentos monitorados.

Na figura 4.13, é possível observar que, conforme observado na descrição textual apresentada anteriormente, o equipamento **100.0.94.58** está conectado, através da porta **388**, à porta **24** do equipamento **100.0.94.58**, com custo 19 (largura de banda 100 Mbps, vide tabela 3.2). Por outro lado, o equipamento **100.0.94.58** está conectado,

através da porta **428**, à porta **14** do equipamento **100.0.94.201**, com custo 4 (largura de banda 1 Gbps, vide tabela 3.2).

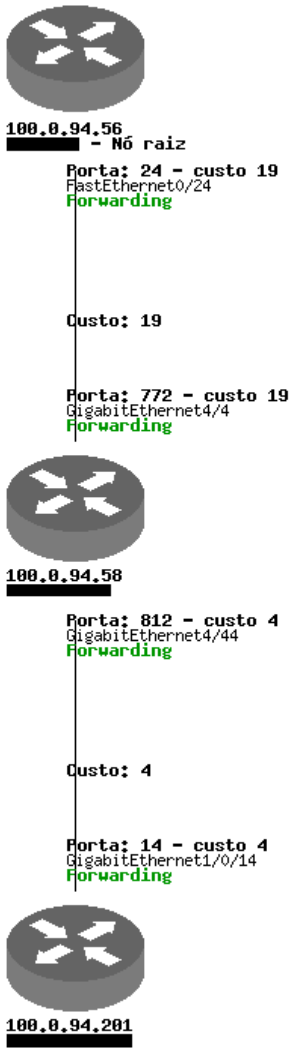


Figura 4.13: Mapa de rede do monitoramento Spanning Tree (SNMP)

4.2.3 Laboratório

Após a montagem da rede em anel, descrita na etapa de laboratório, na sub-seção 3.4.2, buscou-se iniciar o processo de monitoramento WEB dos *switches*, utilizando o protocolo Spanning Tree. Contudo, com a configuração

inicial dos *switches* Cisco ME 3400E, nenhum resultado relativo à base de dados MIB **dot1dStp (1.3.6.1.2.1.17.2)** foi obtido pelo sistema de monitoramento WEB.

No entanto, quando a versão do protocolo Spanning Tree operacional nos *switches* foi modificada de Rapid Spanning Tree (RSTP), para Multiple Spanning Tree (MSTP), com o comando **spanning-tree mode MSTP**, o sistema de monitoramento WEB passou a obter parte das estatísticas necessárias para o monitoramento. Contudo, o valor de custo de cada conexão (**1.3.6.1.2.1.17.2.6 - dot1dStpRootCost**) continuou sem ser obtido (aparecendo com valor zero), assim como os dados relativos a cada uma das portas (interfaces de rede) (**1.3.6.1.2.1.17.2.15 - dot1dStpPortTable**).

Para efeito de comparação, a versão operacional do protocolo Spanning Tree para os equipamentos de rede que estão operando atualmente no *backbone* RedeRio é a 802.1d, onde todas as estatísticas da base de dados MIB **dot1dStp** necessárias para o monitoramento WEB são obtidas sem restrição. Os equipamentos de rede operacionais no *backbone* RedeRio, por sua vez, são roteadores Cisco C7600.

Ao utilizar o comando **show spanning-tree** para os *switches* Cisco ME 3400E, observou-se também que os campos relacionados aos custos de conexão do equipamento corrente para com o *switch*-raiz da Árvore de Custo Mínimo também aparcem com valor zero. A seguir, seguem os resultados da execução mencionada neste parágrafo:

- **Switch1 (10.0.0.10)**

  switch1#show spanning-tree

  MST0
  Spanning tree enabled protocol mstp
  Root ID Priority 32768
  Address 5c50.1569.4d80
  Cost 0
  Port 1 (GigabitEthernet0/1)
  Hello Time 2 sec Max Age 20 sec Forward D

elay 15 sec

Bridge ID Priority 32768 (priority 32768 sys-id-ext 0)
Address 5c50.1569.8e00
Hello Time 2 sec Max Age 20 sec Forward D
elay 15 sec

Interface	Role	Sts	Cost	Prio.Nbr	Type
Gi0/1	Root	FWD	20000	128.1	P2p
Gi0/2	Altn	BLK	20000	128.2	P2p

- **Switch2 (10.0.0.20)**

switch2#show spanning-tree

MST0
Spanning tree enabled protocol mstp
Root ID Priority 32768
Address 5c50.1569.4d80
Cost 0
Port 2 (GigabitEthernet0/2)
Hello Time 2 sec Max Age 20 sec Forward D
elay 15 sec

Bridge ID Priority 32768 (priority 32768 sys-id-ext 0)
Address 5c50.1569.5e00
Hello Time 2 sec Max Age 20 sec Forward D
elay 15 sec

Interface	Role	Sts	Cost	Prio.Nbr	Type
Gi0/1	Desg	FWD	20000	128.1	P2p

Gi0/2 Root BLK 20000 128.2 P2p

- **Switch3 (10.0.0.30)**

switch3#show spanning-tree

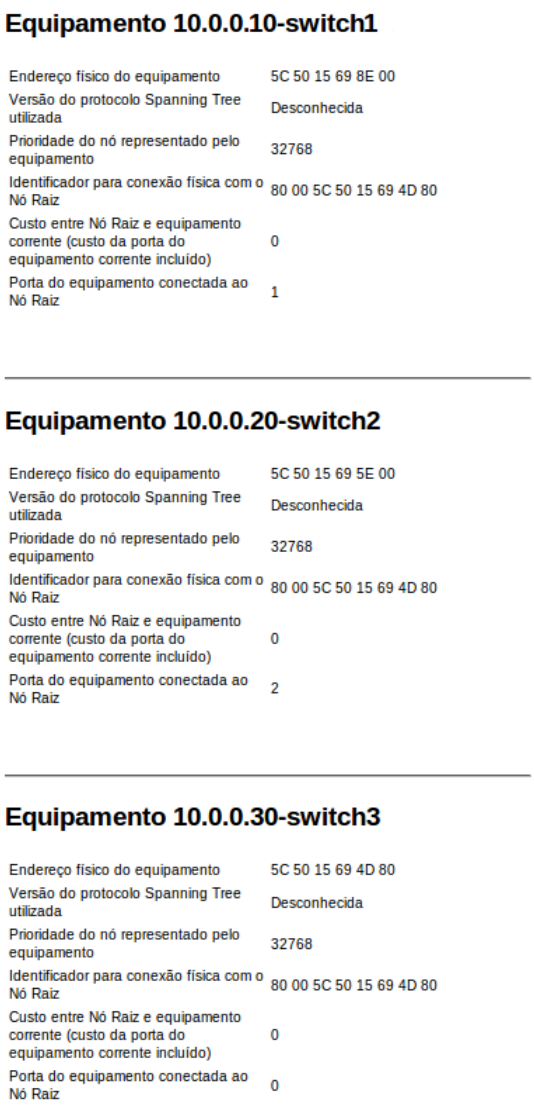
MST0
Spanning tree enabled protocol mstp
Root ID Priority 32768
Address 5c50.1569.4d80
This bridge is the root
Hello Time 2 sec Max Age 20 sec Forward D
elay 15 sec

Bridge ID Priority 32768 (priority 32768 sys-id-ext 0)
Address 5c50.1569.4d80
Hello Time 2 sec Max Age 20 sec Forward D
elay 15 sec

Interface	**Role**	**Sts**	**Cost**	**Prio.Nbr**	**Type**
Gi0/1	**Desg**	**FWD**	**20000**	**128.1**	**P2p**
Gi0/2	**Desg**	**FWD**	**20000**	**128.2**	**P2p**

Conforme mostrado nos resultados da execução do comando **show spanni ng-tree** para os *switches* Cisco ME 3400E, o *switch* Switch3 foi eleito como raiz da Árvore de Custo Mínimo, por possuir o menor valor de endereço físico (MAC Address), uma vez que todos os *switches* possuem o mesmo valor de prioridade. Para evitar redundâncias na rede, a interface GigabitEthernet0/2 do *switch* Switch1 teve seu estado modificado para *blocking*, de modo a impedir o envio/recebimento de mensagens através desta interface.

A figura 4.14 apresenta a relação textual dos dados de monitoramento Spanning Tree dos *switches* Cisco ME 3400E, seguindo o modelo de apresentação desenvolvido para a solução WEB descrita nesta dissertação.



Equipamento 10.0.0.10-switch1

Endereço físico do equipamento	5C 50 15 69 8E 00
Versão do protocolo Spanning Tree utilizada	Desconhecida
Prioridade do nó representado pelo equipamento	32768
Identificador para conexão física com o Nó Raiz	80 00 5C 50 15 69 4D 80
Custo entre Nó Raiz e equipamento corrente (custo da porta do equipamento corrente incluído)	0
Porta do equipamento conectada ao Nó Raiz	1

Equipamento 10.0.0.20-switch2

Endereço físico do equipamento	5C 50 15 69 5E 00
Versão do protocolo Spanning Tree utilizada	Desconhecida
Prioridade do nó representado pelo equipamento	32768
Identificador para conexão física com o Nó Raiz	80 00 5C 50 15 69 4D 80
Custo entre Nó Raiz e equipamento corrente (custo da porta do equipamento corrente incluído)	0
Porta do equipamento conectada ao Nó Raiz	2

Equipamento 10.0.0.30-switch3

Endereço físico do equipamento	5C 50 15 69 4D 80
Versão do protocolo Spanning Tree utilizada	Desconhecida
Prioridade do nó representado pelo equipamento	32768
Identificador para conexão física com o Nó Raiz	80 00 5C 50 15 69 4D 80
Custo entre Nó Raiz e equipamento corrente (custo da porta do equipamento corrente incluído)	0
Porta do equipamento conectada ao Nó Raiz	0

Figura 4.14: Aquisição de dados no monitoramento Spanning Tree

Como forma de buscar uma solução para estes problemas, viabilizando assim o monitoramento WEB do protocolo Spanning Tree para *switches* Cisco ME 3400E, buscou-se suporte técnico junto à empresa Cisco, fabricante dos mesmos. Conforme sugestão obtida, buscou-se também a possibilidade

de obter este suporte junto a empresas que mantém parceria com a Cisco.

Neste momento, foram enviados e-mails para diversas empresas apontadas no *site* oficial da Cisco como suas parceiras.

Desta forma, obteve-se contato com a equipe de suporte da empresa PromonLogicalis, a fim de buscar uma solução para os problemas de monitoramento relatado nesta sub-seção. Ao final deste contato, constatou-se a necessidade de solicitar abertura de um pedido de suporte junto à Cisco, a fim de compreender o porquê dos problemas aqui descritos, bem como obter uma solução para os mesmos. No momento, estão sendo realizados os trâmites burocráticos pertinentes à abertura do pedido.

Como alternativa, foi criada solução de monitoramento baseada no protocolo Telnet, para obtenção dos dados Spanning Tree dos *switches* Cisco ME 3400E. Esta solução obtém os dados, e os salva em arquivos de texto.

Estes arquivos são atualizados periodicamente, para detectar eventuais mudanças em seus dados, mudanças estas que podem alterar o desenho do mapa de rede. Uma vez obtidos os dados, eles são tratados em uma rotina escrita em PHP, responsável por desenhar o mapa de rede.

A figura 4.15 apresenta um exemplo de como o mapa de rede será gerado, a partir da utilização do protocolo Telnet para aquisição de dados. Neste exemplo, as linhas de conexão em verde indicam conexões onde a transmissão de dados está liberada pelo protocolo Spanning Tree (estado *forwarding* nas duas portas); as linhas de conexão em laranja indicam conexões onde a transmissão de dados está bloqueada (estado *blocking* em pelo menos uma das portas); e as linhas em vermelho indicam ausência de conexão entre os equipamentos no presente momento.

A implementação da solução Telnet apontou discrepâncias em relação à primeira solução, baseada no protocolo SNMP, em especial à determinação do equipamento raiz de uma dada rede. Isto ocorreu em função do protocolo SNMP estar obtendo valores incorretos para estas estatísticas, como por exemplo, o endereço físico dos equipamentos. Por causa disto, o cálculo de convergência para a geração da Árvore de Custo Mínimo apresentou resultados diferentes entre uma e outra abordagem, onde, enquanto na figura 4.13, o equipamento 10.0.94.58 não é apontado como raiz da árvore, na figura

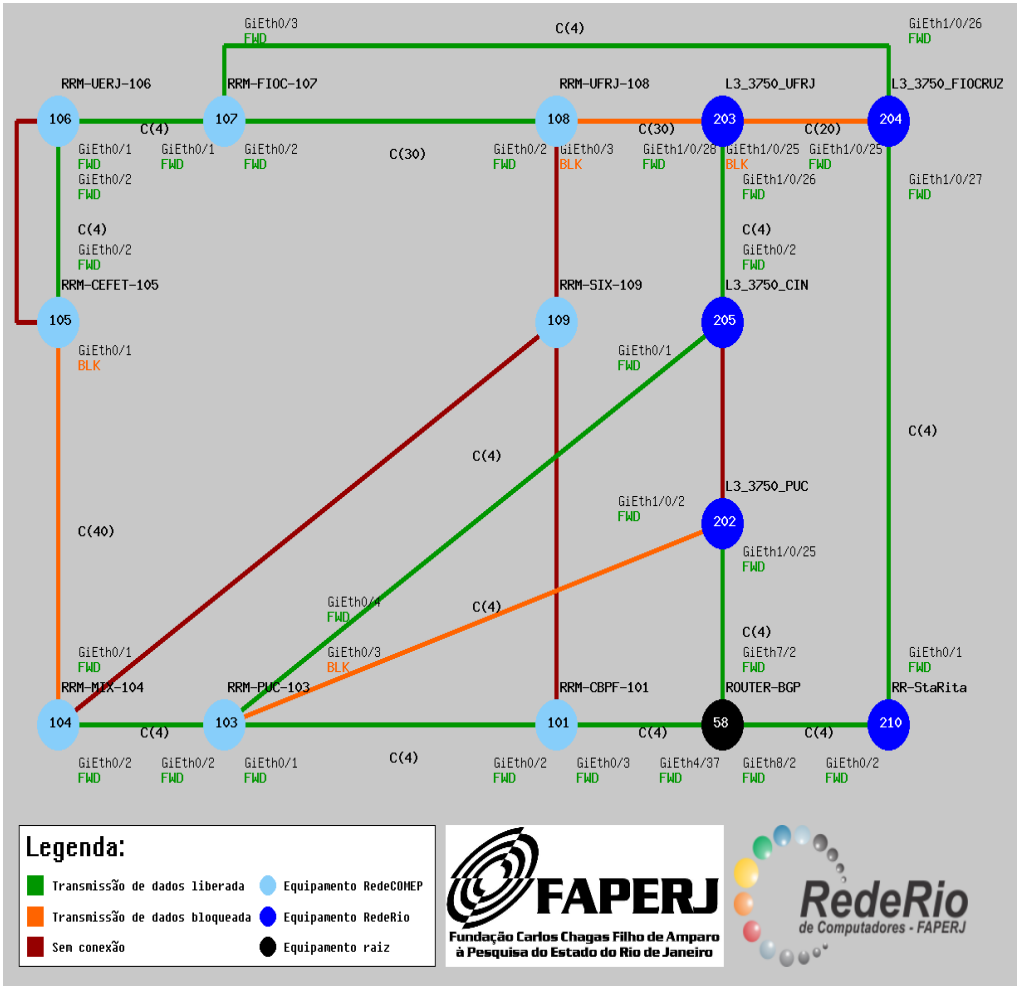


Figura 4.15: Mapa de rede do monitoramento Spanning Tree (Telnet)

4.15 este equipamento passa a se encontrar em tal condição.

Capítulo 5

Conclusões

Esta dissertação descreveu a implementação da gerência de dados para o projeto *Redes Comunitárias de Educação e Pesquisa* - Redecomep, coordenado pela *Rede Nacional de Pesquisas* - RNP, em parceria com a agência *Fundação de Amparo à Pesquisa do Estado do Rio de Janeiro* - FAPERJ, e financiado pela agência *Financiadora de Estudos e Projetos* - FINEP. Além disto, esta dissertação descreveu também a implementação dos módulos de gerência e monitoramento do tráfego de rede, através dos protocolos *Simple Network Management Protocol* - SNMP; e *Spanning Tree Protocol* - STP.

Inicialmente, foram apresentadas as motivações para a realização deste trabalho, incluindo-se neste contexto alguns exemplos de sistemas de monitoramento já implementados. Em seguida, foi feita uma introdução sobre alguns conceitos básicos de redes, seguida por outra introdução sobre os protocolos envolvidos neste projeto: o protocolo de gerência de equipamentos de rede SNMP, e o protocolo de roteamento de rede Spanning Tree, utilizado na segunda etapa do projeto.

Após isto, foi feita uma introdução ao que seria realizado em cada etapa. No capítulo referente à metodologia, foram abordados o ambiente e as ferramentas utilizadas neste sistema (plataforma LAMP - Linux, Apache, MySQL e PHP; mais os aplicativos SNMP, PHP-SNMP, MRTG e Packet Tracer), e a forma como cada uma das etapas foi implementada. Após a apresentação da metodologia seguiram os resultados obtidos, e por fim as conclusões de tudo o que foi realizado neste projeto.

Diante dos aspectos apresentados nos parágrafos anteriores, e ao longo desta dissertação, mostra-se então a importância do protocolo SNMP para o desenvolvimento de sistemas de gerência e monitoração de equipamentos diversos de redes de computadores. Esta importância pode ser verificada em aspectos como:

- A segurança oferecida pelo protocolo, devido à obrigatoriedade da utilização de senhas (*communities*), quando da busca por uma determinada estatística de interesse, ou para modificar valores de variáveis internas responsáveis por estatísticas do equipamento;

- O fato que dificilmente as mensagens do protocolo SNMP ocasionam

congestionamento na rede, em função da quantidade reduzida de aplicativos disponíveis para este protocolo;

- A possibilidade de utilizar o protocolo SNMP para obter dados de qualquer dispositivo de *hardware*, desde *switches* e roteadores; até periféricos, como impressoras. Cabe lembrar que o protocolo predecessor do SNMP (SGMP) permitia somente monitoramento de roteadores.

Por outro lado, pode-se destacar também algumas desvantagens do protocolo SNMP, tais como:

- A implementação do protocolo SNMP pode se tornar um problema em equipamentos com muitos itens existentes em sua base MIB, no sentido de que tais itens podem se tornar de difícil obtenção, caso não se conheça a estrutura da base MIB em questão;

- Equipamentos com muitas interfaces e/ou estatísticas podem acabar retornando uma série de estatísticas desnecessárias para o usuário, dificultando ao usuário o acesso ao que lhe é verdadeiramente necessário;

Como solução para as desvantagens apresentadas, foi proposta a seguinte prática:

- Sempre que for necessário obter uma estatística qualquer via protocolo SNMP, restringir ao máximo os critérios de busca para a mesma, através do identificador OID vinculado à estatística em questão.

Também pôde-se observar, com base nos aspectos apresentados ao longo desta dissertação, a importância do protocolo Spanning Tree para a otimização da atividade de transmissão de dados dentro de uma rede, o que torna interessante desenvolver uma ferramenta capaz de monitorar o comportamento de rede, pela visão proporcionada por este protocolo. Esta importância pode ser verificada em aspectos como:

- Impedir a ocorrência de *loops* de rede, que consistem na existência de pacotes de dados trafegando infinitamente dentro de uma rede, configurando cada interface de um *switch* em estado *forwarding* ou *blocking* [61];

- Possibilitar a criação do sistema de monitoramento de rede descrito nesta dissertação.

Cabe ressaltar também algumas dificuldades encontradas na realização deste trabalho como um todo, bem como visualizar aspectos do mesmo que podem ser implementados e/ou melhorados em trabalhos futuros. Dentro deste contexto, podem-se destacar:

- Problemas com alto consumo de CPU da máquina virtual - em função da rotina escrita na linguagem PHP para geração dos gráficos de monitoramento (rrd_graph.php) - e consequentemente, do servidor onde a máquina está instalada. Como forma de correção, agendou-se, no arquivo **crontab** do sistema operacional Ubuntu, uma rotina **killall php** para, periodicamente, encerrar todos os processos do tipo "php" que estejam rodando no momento em questão;

- Problemas com o monitoramento Spanning Tree nos *switches* Cisco ME 3400E, conforme mencionados na sub-seção 4.2.3. Sem solução no presente momento, ficando assim como meta para trabalhos futuros. Como alternativa, foi desenvolvida uma solução de monitoramento baseada no protocolo Telnet;

- Mal-funcionamento do aplicativo PHPMyAdmin, após a atualização do sistema operacional Ubuntu para a versão 10.04. Sem solução no presente momento, ficando assim como meta para trabalhos futuros. Como contingência para este problema, sempre que é preciso acessar o banco de dados do sistema, tem sido utilizado o aplicativo MySQL através de linhas de comando no aplicativo Terminal do sistema operacional Ubuntu, conforme descrito na sub-seção 3.3.1;

- Obtenção incorreta, por parte do protocolo SNMP, dos valores relativos às estatísticas Spanning Tree dos equipamentos de rede onde este monitoramento está funcionando (equipamentos com versão 802.1d do protocolo Spanning Tree, por exemplo). Sem solução no presente momento, ficando assim como meta para trabalhos futuros.

Bibliografia

[1] Robert Agranoff; Michael McGuire. "Big Questions in Public Network Management Research". *Journal of Public Administration Research & Theory, Jul 2001, Vol. 11 Issue 3, p295.*, 2001.

[2] Minh Hyunh; Prasant Mohapatra. "Metropolitan Ethernet Network: A move from LAN to MAN". *Computer Networks, Dec 2007, Vol. 51 Issue 17, p4867-4894.*, 2007.

[3] Arnold Picot; Christian Wernick. "The role of government in broadband access". *Journal Telecommunications Policy, Nov 2007, Vol. 31 Issue 10-11, p660-674.*, 2007.

[4] Sanjoy Dasgupta; Umesh Vazirani; Christos H. Papadimitriou. *Algorithms*. MCGRAW-HILL, 2009. ISBN 9788577260324.

[5] RNP. `http://www.redecomep.rnp.br`, Acessado em Setembro de 2013.

[6] Centro Brasileiro de Pesquisas Físicas. "Plano Diretor do CBPF - 2011-2015". *Centro Brasileiro de Pesquisas Físicas*, 2011.

[7] Tobias Oetiker. What is MRTG; `http://oss.oetiker.ch/mrtg/doc/mrtg.en.html`, Acessado em Novembro de 2011.

[8] Ethan Galstad. Nagios Official Page; `http://www.nagios.org`, Acessado em Julho de 2012.

[9] Ethan Galstad. Nagios Exchange - STP; `exchange.nagios.org/directory/Plugins/Network-Protocols/STP`, Acessado em Julho de 2012.

[10] The Cacti Group Inc. Cacti Official Page; `http://www.cacti.net/what_is_cacti.php`, Acessado em Julho de 2012.

[11] CERN. CERN Gridmap Official Page; `http://grid-monitoring.cern.ch/myegi/gridmap`, Acessado em Julho de 2012.

[12] E. Lanciotti; J. Andreeva; M. Boehm; A. Casajus; J. Casey; B. Gaidioz; C. Grigoras; L. Kokoszkiewicz; R. Rocha; P. Saiz; I. Sidorova; A. Tsaregorotsev. "Siteview GridMap: A new monitoring tool from the site point of view". *CERN IT Department*, 2009.

[13] Rodrigo C. Krüger; Rodrigo L. Michel. "Ferramentas Para Gerenciamento, Monitoramento de Redes e Assistência Remota". *Pós Graduação em Redes e Segurança de Sistemas - Pontifícia Universidade Católica do Paraná*, 2010.

[14] Daniel S. Spozito. "Monitoramento de Hosts em Redes TCP/IP sobre Base Georre-ferenciada com Métricas de Qualidade". *Programa de Pós-graduação em Engenharia Elétrica - Universidade Estadual Paulista*, 2011.

[15] James F. Kurose; Keith W. Ross. *Computer Networking: a Top-Down Approach Featuring the Internet*. Addison-Wesley, 2006. ISBN 8588639181.

[16] Larry L. Peterson; Bruce S. Davie. *Computer Networks: A System Approach*. Morgan Kaufman Publishers, 2007. ISBN 0123705487.

[17] Tatiana L. Ferraz; Marcelo P. Albuquerque. "Introdução ao PING e Traceroute". *Nota Técnica CBPF-NT-010/02*, 2002.

[18] Jonathan Postel. "Internet Protocol - RFC791". *Network Working Group*, 1981.

[19] Jonathan Postel. "Assigned Numbers - RFC790". *Network Working Group*, 1981.

[20] S. Bradner; A. Mankin. "The Recomendation for the IP Next Generation Protocol - RFC1752". *Network Working Group*, 1995.

[21] S. Deering; R. Hinden. "Internet Protocol, Version 6 (IPv6) Specification - RFC2460". *Network Working Group*, 1998.

[22] Jonathan Postel. "Assigned Numbers - RFC792". *Network Working Group*, 1981.

[23] Volnys Borges Bernal. "Tecnologia de Redes; Protocolo ICMP "Internet Control Message Protocol"". *Laboratório de Sistemas Integrados da Universidade de São Paulo*, 2000.

[24] Information Sciences Institute of University of Southern California. "Transmission Control Protocol - RFC793". *Defense Advanced Research Projects Agency*, 1981.

[25] Marcelo Gonçalves Rubinstein. "Redes de Computadores". *Programa de Pós-Graduação em Engenharia Eletrônica da Universidade do Estado do Rio de Janeiro*, 2011.

[26] Jonathan Postel. "User Datagram Protocol - RFC768". *Network Working Group*, 1980.

[27] Jonathan Postel; J. Reynolds. "Telnet Protocol Specification - RFC854". *Network Working Group*, 1983.

[28] Telnet.org. `http://www.telnet.org`, Acessado em Fevereiro de 2013.

[29] Jonathan Postel; J. Reynolds. "Telnet Option Specifications - RFC855". *Network Working Group*, 1983.

[30] Jonathan Postel; J. Reynolds. "Telnet Binary Transmission - RFC856". *Network Working Group*, 1983.

[31] Jonathan Postel; J. Reynolds. "Telnet Echo Option - RFC857". *Network Working Group*, 1983.

[32] Jonathan Postel; J. Reynolds. "Telnet Supress Go Ahead Option - RFC858". *Network Working Group*, 1983.

[33] Jonathan Postel; J. Reynolds. "Telnet Status Option - RFC859". *Network Working Group*, 1983.

[34] Jonathan Postel; J. Reynolds. "Telnet Timing Mark Option - RFC860". *Network Working Group*, 1983.

[35] Jonathan Postel; J. Reynolds. "Telnet Extended Options - List Option - RFC861". *Network Working Group*, 1983.

[36] Beethovem Z. Dias; Nilton A. Jr. "Protocolo de Gerenciamento SNMP". *Nota Técnica CBPF-NT-006/01*, 2001.

[37] J. Case; M. Fedor; M. Schoffstal; J. Davin. "A Simple Network Management Protocol (SNMP) - RFC1157". *Network Working Group*, 1990.

[38] Douglas R. Mauro; Kevin J. Schmidt. *SNMP Essencial*. Editora Campus, 2001. ISBN 8535208828.

[39] J. Davin; J. Case; M. Fedor; M. Schoffstall. "A Simple Gateway Monitoring Protocol - RFC1028". *Network Working Group*, 1987.

[40] S. Bradner. "The Internet Standards Process – Revision 3 - RFC2026". *Network Working Group*, 1996.

[41] J. Case; K. McCloghrie; M. Rose; S. Waldbusser. "Strucuture of Management Information for Version 2 of the Simple Network Management Protocol (SNMPv2) - RFC1902". *Network Working Group*, 1996.

[42] J. Case; K. McCloghrie; M. Rose; S. Waldbusser. "Textual Coventions for Version 2 of the Simple Network Management Protocol (SNMPv2) - RFC1903". *Network Working Group*, 1996.

[43] J. Case; K. McCloghrie; M. Rose; S. Waldbusser. "Conformance Statements for Version 2 of the Simple Network Management Protocol (SNMPv2) - RFC1904". *Network Working Group*, 1996.

[44] J. Case; K. McCloghrie; M. Rose; S. Waldbusser. "Protocol Operations for Version 2 of the Simple Network Management Protocol (SNMPv2) - RFC1905". *Network Working Group*, 1996.

[45] J. Case; K. McCloghrie; M. Rose; S. Waldbusser. "Transport Mappings for Version 2 of the Simple Network Management Protocol (SNMPv2) - RFC1906". *Network Working Group*, 1996.

[46] J. Case; K. McCloghrie; M. Rose; S. Waldbusser. "Management Information Base for Version 2 of the Simple Network Management Protocol (SNMPv2) - RFC1907". *Network Working Group*, 1996.

[47] J. Case; K. McCloghrie; M. Rose; S. Waldbusser. "Coexistence between Version 1 and Version 2 of the Internet-standard Network Management Framework - RFC1908". *Network Working Group*, 1996.

[48] K. McCloghrie. "An Administrative Infrastructure for SNMPv2 - RFC1909". *Network Working Group*, 1996.

[49] J. Case; R. Mundy; D. Partain; B. Stewart. "Introduction to Version 3 of the Internet-standard Network Management Framework - RFC2570". *Network Working Group*, 1999.

[50] D. Harrington; R. Presuhn; B. Wijnen. "An Architecture for Describing SNMP Management Frameworks - RFC2571". *Network Working Group*, 1999.

[51] J. Case; D. Harrington; R. Presuhn; B. Wijnen. "Message Processing and Dispatching for the Simple Network Management Protocol (SNMP) - RFC2572". *Network Working Group*, 1999.

[52] D. Levi; P. Meyer; B. Stewart. "SNMP Applications - RFC2573". *Network Working Group*, 1999.

[53] U. Blumenthal; B. Wijnen. "User-based Security Model (USM) for the Simple Network Management Protocol (SNMPv3) - RFC2574". *Network Working Group*, 1999.

[54] B. Wijnen; R. Presuhn; K. McCloghrie. "View-based Access Control Model (VACM) for the Simple Network Management Protocol (SNMP) - RFC2575". *Network Working Group*, 1999.

[55] K. McCloghrie; M. Rose. "Management Information Base for Network Management of TCP/IP-based Internets: MIB II - RFC1213". *Network Working Group*, 1991.

[56] K. McCloghrie; M. Rose. "Management Information Base for Network Management of TCP/IP-based Internets - RFC1066". *Network Working Group*, 1988.

[57] Grupo de Redes da Universidade Federal do Rio Grande do Sul. Especificação do protocolo SNMP; `http://penta.ufrgs.br/gr952/trab1/snmp_especificacao.html/`, Dezembro de 1995; Acessado em Dezembro de 2012.

[58] Cisco Systems Inc. SNMP Object Navigator; `tools.cisco.com/Support/SNMP/do/BrowseOID.do?local=en`, Maio de 2011; Acessado em Dezembro de 2012.

[59] K. McCloghrie; M. Rose. "Structure and Identification of Management Information for TCP/IP-based Internets - RFC1155". *Network Working Group*, 1990.

[60] SourceForge.net. Tutorial para utilização do protocolo SNMP; `http://www.net-snmp.org/`, Março de 2007; Acessado em Dezembro de 2012.

[61] Andrew S. Tanenbaum. *Modern Operating Systems*. Makron Books, 2007. ISBN 0130313580.

[62] Richard T. Griffiths. History of the Internet, Internet for Historians; `http://www.let.leidenuniv.nl/history/ivh/frame_theorie.html`, Outubro de 2002; Acessado em Dezembro de 2012.

[63] Jayme Luiz Szwarcfiter. *Grafos e algoritmos computacionais*. Campus, 1988. ISBN 8570013418.

[64] Joseph B. Kruskal. "On the Shortest Spanning Subtree of a Graph and the Traveling Salesman Problem". *Proceedings of the American Mathematical Society*, Fevereiro de 1956.

[65] Robert C. Prim. "Shortest connection networks and some generalizations". *Bell System Technical Journal*, 1957.

[66] Marco Aurélio Filippetti. *CCNA 4.1 - Guia Completo de Estudo*. Visual Books, 2008. ISBN 8575022385.

[67] Cisco Systems Inc. Spanning Tree Protocol - Cisco Systems; `http://www.cisco.com/en/US/tech/tk389/tk621/tsd_technology_support_protocol_home.html`, Acessado em Novembro de 2012.

[68] Kennedy Clark; Kevin Hamilton. *Cisco LAN Switching*. Cisco Press, 1999. ISBN 1578700949.

[69] Cisco. Understanding rapid spanning tree protocol (802.1w); `http://www.cisco.com/en/US/tech/tk389/tk621/technologies_white_paper09186a0080094cfa.shtml`, Outubro de 2006; Acessado em Novembro de 2012.

[70] Augusto Campos. O que é Linux; `http://br-linux.org/faq-linux/`, Março de 2006; Acessado em Setembro de 2011.

[71] Rafael Proença. O que é Ubuntu?; `http://www.ubuntu-br.org/ubuntu`, 2007; Acessado em Setembro de 2011.

[72] André Milani. *MySQL - Guia do Programador*. Novatec, 2007. ISBN 8575221035.

[73] R. Fielding; J. Gettys; J. Mogul; H. Frystyk; L. Masinter; P. Leach; T. Berners-Lee. "hypertext Transfer Protocol – HTTP/1.1 - RFC2616". *Network Working Group*, 1999.

[74] Apache Software Foundation. Apache HTTP Server - About Apache; http://httpd.apache.org/ABOUT_APACHE.html, Acessado em Outubro de 2011.

[75] The PHP Group. PHP: Hypertext Preprocessor; http://www.php.net, Acessado em Novembro de 2011.

[76] SourceForge.net. PHPMyAdmin Home Page; http://www.phpmyadmin.net, Acessado em Setembro de 2012.

[77] Flávio Régis de Arruda. Simulador de Escalonamento Distribuído de Processos Baseado no Algoritmo Round-Robin; http://www.ime.usp.br/~kon/MAC5755/trabalhos/software/FlavioArruda/pagina.html, Acessado em Setembro de 2011.

[78] Tobias Oetiker. About RRDtool; http://oss.oetiker.ch/rrdtool/doc/rrdtool.en.html, Acessado em Novembro de 2011.

[79] Prajakta S. Kalekar. "Time series Forecasting using Holt-Winters Exponential Smoothing". *Kanwal Rekhi School of Information Technology*, 2004.

[80] The PHP Group. PHP: GD - Manual; http://www.php.net/manual/pt_BR/book.image.php, Acessado em Novembro de 2012.

[81] Cisco Networking Academy. "Packet Tracer 5.0 Overview". *Cisco Systems, Inc.*, 2008.

[82] David Eppstein. "Spanning Trees And Spanners". *Handbook of Computational Geometry:*, 1999.

[83] Lisiane Hartmann. Protocolo de Roteamento EGP; http://penta.ufrgs.br/rc952/Lisiane/lis220.html, Setembro de 1995; Acessado em 3 de Dezembro de 2011.

Apêndice A

Modelo Entidade-Relacionamento do sistema

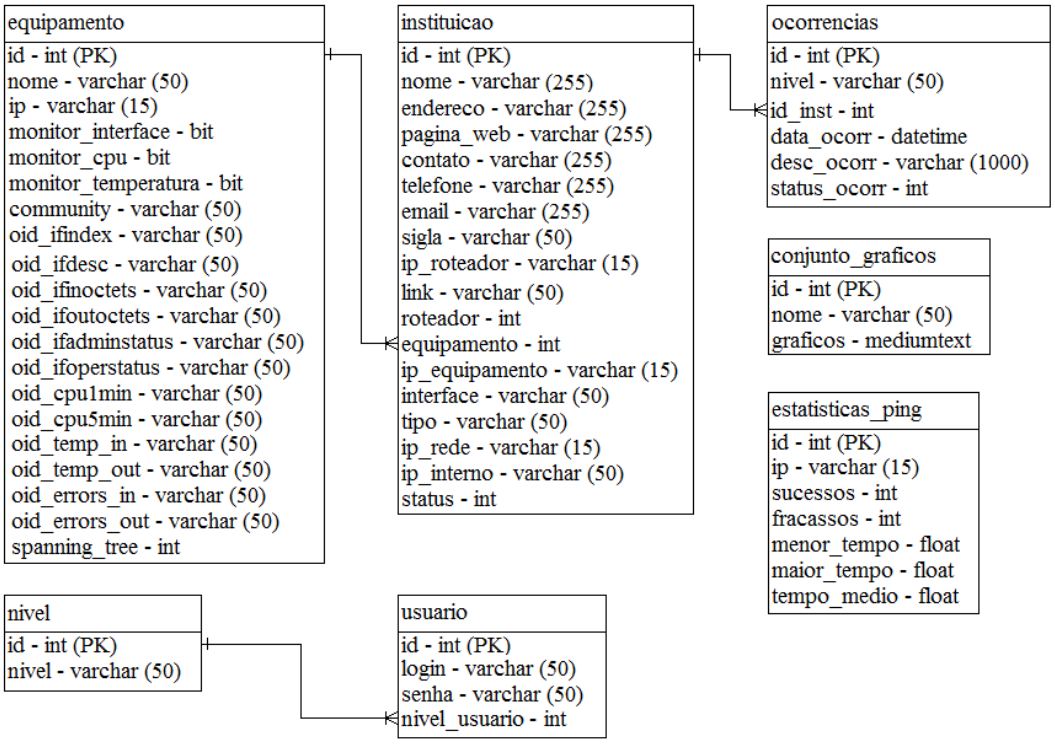


Figura A.1: Modelo Entidade-Relacionamento

Apêndice B

Exemplos de nós existentes na sub-árvore MIB II

A seguir, seguem alguns exemplos de estatísticas que podem ser obtidas via monitoramento SNMP, por meio de acessos à base MIB do equipamento monitorado.

- **Nó System (.1.3.6.1.2.1.1)**: Faz a listagem de objetos existentes no sistema operacional do equipamento gerenciado;
  - **sysDescr (.1.3.6.1.2.1.1.1)**: Descrição textual do equipamento. Pode incluir o nome e a versão do hardware do equipamento e seu sistema operacional. Variável do tipo String;
  - **sysUpTime (.1.3.6.1.2.1.1.3)**: Tempo transcorrido (em milhares de segundos) desde a última vez em que o equipamento foi reinicializado. Variável do tipo TimeTicks;
  - **sysContact (.1.3.6.1.2.1.1.4)**: Descrição da identificação do gerente da máquina gerenciada e informações de contato. Variável do tipo String;
  - **sysORLastChange (.1.3.6.1.2.1.1.8)**: Tempo transcorrido (em milhares de segundos) desde a última modificação ocorrida em alguma instância do equipamento. Variável do tipo TimeTicks.

- **Nó Interfaces (.1.3.6.1.2.1.2)**: Lista as informações relativas às interfaces do equipamento;
  - **ifNumber (.1.3.6.1.2.1.2.1)**: Total de interfaces de rede existentes no equipamento (independente do estado atual delas). Variável do tipo Integer;
  - **ifAdminStatus (.1.3.6.1.2.1.2.2.1.7)**: Estado esperado para a interface. Variável do tipo Integer, tendo como possíveis valores *up(1)* ou *down(2)*;
  - **ifOperStatus (.1.3.6.1.2.1.2.2.1.8)**: Estado atual da interface. Variável do tipo Integer, tendo como possíveis valores *up(1)* ou *down(2)*;
  - **ifInOctets (.1.3.6.1.2.1.2.2.1.10)**: Número total de *bytes* recebidos pela interface. Variável do tipo Counter;
  - **ifOutOctets (.1.3.6.1.2.1.2.2.1.16)**: Número total de *bytes* enviados pela interface. Variável do tipo Counter.
- **Nó AT (.1.3.6.1.2.1.3)**: Responsável pela tradução de endereços das interfaces do equipamento, mantido somente por questões de compatibilidade com outras versões da base MIB, sendo que não existirá na próxima versão da mesma (MIB-III);
  - **atIfIndex (.1.3.6.1.2.1.3.1.1.1)**: Identificador da interface do equipamento. Variável do tipo Integer;
  - **atPhysAddress (.1.3.6.1.2.1.3.1.1.2)**: Endereço físico da interface do equipamento. Variável do tipo String;
  - **atNetAddress (.1.3.6.1.2.1.3.1.1.3)**: Endereço de rede (IP) da interface do equipamento. Variável do tipo NetworkAddress.
- **Nó IP (.1.3.6.1.2.1.4)**: Faz o mapeamento dos atributos do protocolo IP para o equipamento;
  - **ipForwarding (.1.3.6.1.2.1.4.1)**: Indica se o equipamento é um *gateway*. Variável do tipo Integer, podendo assumir os valores *1*

(*forwarding* - equipamento é um *gateway*); ou 2 (*notForwarding* - equipamento não é um *gateway*);

  - **ipDefaultTTL (.1.3.6.1.2.1.4.2)**: Indica o valor *default* da variável TTL, responsável pelo número máximo de saltos que um datagrama IP pode realizar de um ponto a outro, dentro de uma rede. Variável do tipo Integer;
  - **ipInReceives (.1.3.6.1.2.1.4.3)**: Total de datagramas recebidos pelas interfaces, independente de terem ou não sido recebidos com sucesso. Variável do tipo Counter32;
  - **ipInHdrErrors (.1.3.6.1.2.1.4.4)**: Total de datagramas recebidos e descartados com erros em seus respectivos cabeçalhos. Variável do tipo Counter32.

- **Nó ICMP (.1.3.6.1.2.1.5)**: Faz o mapeamento dos atributos do protocolo ICMP para o equipamento;

  - **icmpInMsgs (.1.3.6.1.2.1.5.1)**: Total de mensagens ICMP recebidas pelo equipamento, com ou sem erros no recebimento. Variável do tipo Counter32;
  - **icmpInErrors (.1.3.6.1.2.1.5.2)**: Total de mensagens ICMP recebidas pelo equipamento com algum erro especificado pelo protocolo ICMP. Variável do tipo Counter32;
  - **icmpInDestUnreachs (.1.3.6.1.2.1.5.3)**: Total de mensagens ICMP recebidas pelo equipamento com erro de destino não encontrado (*Destination Unreachable*). Variável do tipo Counter32;
  - **icmpOutMsgs (.1.3.6.1.2.1.5.14)**: Total de mensagens ICMP enviadas pelo equipamento, com ou sem erros no envio. Variável do tipo Counter32;
  - **icmpOutErrors (.1.3.6.1.2.1.5.15)**: Total de mensagens ICMP enviadas pelo equipamento com algum erro especificado pelo protocolo ICMP. Variável do tipo Counter32.

- **Nó TCP (.1.3.6.1.2.1.6)**: Faz o mapeamento dos atributos do protocolo TCP para o equipamento;
  - **tcpRtoMin(.1.3.6.2.1.6.2)**: Tempo mínimo permitido, em milisegundos, para a retransmissão de pacotes com erro de *timeout* para implementações TCP. Variável do tipo Integer;
  - **tcpMaxConn(.1.3.6.2.1.6.4)**: Número máximo de conexões TC P suportadas pelo equipamento. Variável do tipo Integer;
  - **tcpCurrentEstab (.1.3.6.2.1.6.9)**: Número de conexões TCP abertas no equipamento (estabelecidas ou à espera de fechamento). Variável do tipo Gauge;
  - **tcpInSegs (.1.3.6.2.1.6.11)**: Total de segmentos recebidos pelo equipamento, via protocolo TCP, com ou sem erro na transmissão. Variável do tipo Counter;
  - **tcpRetransSegs (.1.3.6.2.1.6.12)**: Número total de segmentos retransmitidos pelo equipamento, via protocolo TCP, após erro na transmissão anterior. Variável do tipo Counter.
- **Nó UDP (.1.3.6.1.2.1.7)**: Faz o mapeamento dos atributos do protocolo UDP para o equipamento;
  - **udpInDatagrams (.1.3.6.1.2.1.7.1)**: Total de datagramas UDP entregues aos usuários deste protocolo. Variável do tipo Counter;
  - **udpNoPorts (.1.3.6.1.2.1.7.2)**: Total de datagramas UDP recebidos, onde não existe aplicação na porta especificada por cada um deles. Variável do tipo Counter;
  - **udpInErrors (.1.3.6.1.2.1.7.3)**: Total de datagramas UDP recebidos com quaisquer erros, exceto erro de inexistência de aplicação na porta especificada pelo datagrama. Variável do tipo Counter;
  - **udpLocalAddress (.1.3.6.1.2.1.7.5.1.1)**: Endereço IP do usuário UDP do equipamento gerenciado. Variável do tipo IpAddress;

  - **udpLocalPort (.1.3.6.1.2.1.7.5.1.2)**: Número da porta do usuário UDP do equipamento gerenciado. Variável do tipo Integer.

- **Nó EGP (.1.3.6.1.2.1.8)**: Faz o mapeamento dos atributos do protocolo EGP, responsável por detectar equipamentos de rede e/ou redes que estejam conectados ao equipamento monitorado [83];

  - **egpInMsgs (.1.3.6.1.2.1.8.1)**: Total de mensagens EGP recebidas sem erro. Variável do tipo Counter;
  - **egpOutErrors (.1.3.6.1.2.1.8.4)**: Total de mensagens EGP que não puderam ser enviadas com sucesso. Variável do tipo Counter;
  - **egpNeighState (.1.3.6.1.2.1.8.5.1.1)**: Estado atual do sistema EGP em relação ao acesso a seu vizinho mais imediato. Variável do tipo Integer, podendo assumir os valores *1* (*idle* - inativo); *2* (*acquisition* - depois de solicitado o acesso); *3* (*down* - após confirmação de acesso); *4* (*up* - após o acesso, estabelecimento de conexão); ou *5* (*cease* - fim de utilização);
  - **egpNeighAddr (.1.3.6.1.2.1.8.5.1.2)**: Endereço IP do vizinho EGP do equipamento de rede monitorado. Variável do tipo IpAddress;
  - **egpAs (.1.3.6.1.2.1.8.6)**: Número identificador do Sistema Autônomo (*Autonomous System* - AS) da entidade EGP. Sistema Autônomo pode ser entendido como por todo conjunto de redes e/ou equipamentos de rede que sejam administrados por uma entidade (empresa, administrador de redes, etc.) específica [83]. Variável do tipo Integer.

- **Nó Transmission (.1.3.6.1.2.1.10)**: Configurações de transmissão do elemento de rede;

- **Nó SNMP (.1.3.6.1.2.1.11)**: Faz o mapeamento dos atributos do protocolo SNMP para o equipamento.

- **snmpInPkts (.1.3.6.1.2.1.11.1)**: Total de mensagens recebidas pelo equipamento monitorado pelo protocolo SNMP. Variável do tipo Counter;
- **snmpOutPkts (.1.3.6.1.2.1.11.2)**: Total de mensagens enviadas pelo equipamento monitorado pelo protocolo SNMP. Variável do tipo Counter;
- **snmpInTotalReqVars (.1.3.6.1.2.1.11.13)**: Total de objetos da base MIB resgatados pelo equipamento monitorado pelo protocolo SNMP. Variável do tipo Counter.

Apêndice C

Funcionamento de alguns aplicativos do protocolo SNMP

Neste apêndice, serão apresentados alguns dos aplicativos mais utilizados no protocolo SNMP, seu funcionamento e sua sintaxe básica.

Antes de iniciar a descrição propriamente dita sobre estes aplicativos, é pertinente esclarecer algumas nomenclaturas utilizadas na sintaxe básica das linhas de comando destes aplicativos.

O termo COMMON FLAGS representa os parâmetros de acesso mais utilizados pelos aplicativos do protocolo SNMP, como *version* (-v) e *community* (-c), parâmetros estes que normalmente são obrigatórios para o acesso a um equipamento de rede, mas havendo também casos em que os mesmos podem ser opcionais, caso não hajam restrições de acesso ao equipamento.

O termo OPTIONS representa os parâmetros diversos de configuração do aplicativo em questão, que podem ser utilizados para definir formas específicas de acesso a dadas informações do equipamento monitorado.

O termo HOST representa o equipamento acessado pelo aplicativo. Este termo pode vir em forma textual, ou na forma do endereço IP do equipamento.

O termo OID representa a estatística a ser acessada pelo aplicativo, quando for o caso. Pode aparecer como nome textual da mesma, ou como seu respectivo identificador OID.

Uma vez esclarecidas as nomenclaturas utilizadas, pode-se então descrever os aplicativos do protocolo SNMP propriamente ditos.

Os comandos são limitados, em termos de quantidade de comandos disponíveis para uso, e se baseiam no mecanismo de busca/alteração. Estão disponíveis operações de alteração e de obtenção dos valores de um objeto. Esta limitação no conjunto de comandos facilita a implementação do protocolo, tornando-o mais simples, estável e flexível, reduzindo assim o tráfego de mensagens de gerenciamento através da rede, reduzindo assim a possibilidade de tal tráfego gerar congestionamentos e perdas desnecessárias de pacotes na rede, permitindo ainda a introdução de outras funcionalidades ao protocolo[36].

Fazem parte do conjunto bàsico do comando SNMP [60]:

1. **snmptranslate**

   (a) **Descrição**: Este aplicativo traduz um ou mais identificadores de objetos SNMP de sua forma de string para o correspondente numérico da mesma, ou vice-versa;

   (b) **Sintaxe Básica**: *snmptranslate [OPTIONS] OID [OID] ...*

2. **snmpwalk**

   (a) **Descricão**: Retorna uma sub-árvore de valores de gerenciamento, com seus respectivos dados, utilizando requisições SNMP do tipo GETNEXT, onde a requisição obtém o próximo identificador OID, a partir do identificador que foi passado como argumento de entrada no aplicativo *snmpwalk*. O identificador OID especifica a parte da estatística que será procurada na requisição, retornando assim todas as variáveis da sub-árvore com raiz no identificador OID especificado para o usuário.

   Caso não seja passado nenhum identificador OID na linha de comando do aplicativo, será utilizado como valor *default* o SNMPv2-SMI::mib-2, junto com todos os valores de objetos de outros módulos MIB que estejam incluídos na sub-árvore gerada a partir do identificador *default*.

Ocorrendo erro na rede, durante o processamento da requisição, um pacote de erro será enviado, junto com uma mensagem discriminando o que está incorreto na requisição.

Se a construção da sub-árvore causar uma busca que exceda o fim da base de estatísticas da rede, a mensagem *"End of MIB"* será mostrada ao usuário;

(b) **Sintaxe básica**: *snmpwalk [COMMON FLAGS] [OPTIONS] HOST [OID]*

(c) **Application Flags**:

-Cc Desabilita a detecção, ativada por *default* no aplicativo *snmpwalk*, se os OID's retornados pelo comandos estão fora de ordem;

-ce {HOST} Dentro da sub-árvore gerada no *snmpwalk*, este comando pode extrair uma porção menor da mesma, algumas colunas específicas de uma tabela, ou ainda duas ou mais tabelas; indo da raiz da sub-árvore de HOST, até o OID especificado ao final do comando. Neste caso, a sintaxe do aplicativo *snmpwalk* passa a ser *snmpwalk [APPLICATION FLAGS — -ce [HOST]] [COMMON FLAGS] [OID]*

-Cp Ao final da busca, retorna o total de variáveis encontradas na execução do aplicativo *snmpwalk*;

-Ct Ao final da busca, retorna o tempo gasto na execução do aplicativo *snmpwalk*.

(d) **Common Flags**:

-c Comunidade (*community*) para a qual se pretende gerar a sub-árvore de busca. Ex.: -c public (*community* do tipo public);

-v Versão do protocolo SNMP que está rodando no ambiente de onde se faz a consulta. Ex.: -v 2c (versão 2 do protocolo SNMP).

3. **snmpdelta**:

(a) **Descricão**: Aplicativo utilizado para monitorar, ao longo do tempo, a troca de valores de uma determinada estatística do equipamento monitorado. Esta monitoração apresenta, a cada segundo, a diferença entre o valor atual da estatística e o valor que ela apresentava no segundo anterior.

Este aplicativo pode ser utilizado também para apresentar variações de mais de uma estatística simultaneamente. Para isto, deverão ser passados na linha de comando os identificadores OID das estatísticas de interesse;

(b) **Sintaxe básica**: *snmpdelta [COMMON FLAGS] [OPTIONS] HO ST OID [OID] ...*

4. **snmpget**:

(a) **Descrição**: Enquanto o aplicativo *snmpwalk* retorna uma subárvore com raiz no HOST, o aplicativo *snmpget* retorna a string de um único identificador OID, juntamente com a estatística associada ao mesmo. É possível colocar mais de um identificador como parâmetro de entrada no aplicativo, de modo que, cada linha da saída retornada corresponderá a um identificador OID passado na entrada (funcionalidade disponível a partir da versão 2 do protocolo SNMP, na versão 1 só é possível se passar um identificador OID por entrada);

(b) **Sintaxe básica**: *snmpget [COMMON FLAGS] HOST OID [OID] ...*

(c) **Application Flags**:

-On Utilizado para recuperar o valor numérico de um dado OID. O OID passado no aplicativo *snmpget*, neste caso, deverá ser obrigatoriamente no formato de string.

(d) **Common Flags**: As mesmas do aplicativo *snmpwalk*.

5. **snmpgetnext**:

(a) **Descrição**: Opera de forma similar ao aplicativo *snmpget*, no entanto retorna a string do identificador OID imediatamente posterior - na base MIB do equipamento monitorado - ao identificador passado como argumento na linha de comando do aplicativo, juntamente com a estatística associada ao mesmo;

(b) **Sintaxe básica**: *snmpgetnext [COMMON FLAGS] [OPTIONS] HOST OID [OID] ...*

6. **snmpset**:

(a) **Descrição**: Aplicativo utilizado para escrever um determinado valor em alguma estatística do equipamento gerenciado. A linha de comando do aplicativo deve conter o tipo de variável da estatística - tipo este que deve ser compatível com o tipo de dado definido previamente para esta estatística - e o valor a ser escrito;

(b) **Sintaxe básica**: *snmpset [COMMON FLAGS] [OPTIONS] HOS T OID TYPE VALUE [OID TYPE VALUE] ...*

7. **snmpdf**:

(a) **Descrição**: Aplicativo utilizado para obter a quantidade de espaço disponível em disco do equipamento monitorado, analisando o quanto de espaço está sendo ocupado pelos processos em operação no mesmo;

(b) **Sintaxe básica**: *snmpdf [COMMON FLAGS] [OPTIONS] HOST*

8. **snmpnetstat**:

(a) **Descrição**: Aplicativo utilizado para apresentar informações diversas de rede sobre o equipamento monitorado. Estas informações são organizadas no formato de tabela, e podem representar desde a relação de conexões estabelecidas por elementos integrantes do equipamento a outros pontos de rede; até estatísticas das interfaces do equipamento (por exemplo, pacotes recebidos e enviados);

(b) **Sintaxe básica**: *snmpnetstat [COMMON FLAGS] [OPTIONS] HOST*

9. **snmpstatus**:

(a) **Descrição**: Aplicativo utilizado para apresentar o status geral do equipamento monitorado. Este status mostra:

- O endereço IP do equipamento;
- Uma descrição geral do equipamento;
- A última vez em que o gerenciamento do equipamento via protocolo SNMP foi inicializado;
- O total de pacotes recebidos por todas as interfaces do equipamento;
- O total de pacotes enviados por todas as interfaces do equipamento;
- O total de datagramas IP recebidos pelas interfaces;
- O total de datagramas IP enviados pelas interfaces.

(b) **Sintaxe básica**: *snmpstatus [COMMON FLAGS] [OPTIONS] HOST*