UNIVERSIDADE REGIONAL DE BLUMENAU

CENTRO DE CIÊNCIAS EXTAS E NATURAIS

CURSO DE CIÊNCIAS DA COMPUTAÇÃO

(Bacharelado)

PROTÓTIPO DE UM SOFTWARE AGENTE SNMP PARA REDE WINDOWS

TRABALHO DE CONCLUSÃO DE CURSO SUBMETIDO À UNIVERSIDADE REGIONAL DE BLUMENAU PARA OBTENÇÃO DOS CRÉDITOS NA DISCIPLINA COM NOME EQUIVALENTE NO CURSO DE CIÊNCIAS DA COMPUTAÇÃO - BACHARELADO

LUCIANO WALTRICK GOETEN

BLUMENAU, JUNHO / 2001

PROTÓTIPO DE UM SOFTWARE AGENTE SNMP PARA REDE WINDOWS

LUCIANO WALTRICK GOETEN

ESTE TRABALHO DE CONCLUSÃO DE CURSO, FOI JULGADO ADEQUADO PARA OBTENÇÃO DOS CRÉDITOS NA DISCIPLINA DE TRABALHO DE CONCLUSÃO DE CURSO OBRIGATÓRIA PARA OBTENÇÃO DO TÍTULO DE:

BACHAREL EM CIÊNCIAS DA COMPUTAÇÃO

Prof. Sérgio Stringari - Orientador

Prof. José Roque Voltolini da Silva - Coordenador do TCC

BANCA EXAMINADORA

Prof. Sérgio Stringari

Prof. Francisco Adell Péricas

Prof. Miguel Alexandre Wisintainer

AGRADECIMENTOS

Agradeço em primeiro lugar, aos meus pais e, minha irmã Tatiane Caroline Waltrick Goeten, os quais, em muitas vezes deram-me forças e conselhos para que jamais desistisse dessa etapa que hoje está sendo concluída.

Aos meus amigos que sempre estiveram ao meu lado no decorrer desta caminhada, dando apoio nas horas mais difíceis e também nas horas fáceis.

Ao meu mestre e orientador Sérgio Stringari, que sempre nas horas de sufoco, jamais negou-se a ajudar-me na elaboração e fundamentação deste TCC, auxiliando-me com materiais e bibliografias para que este não ficasse comprometido.

Um agradecimento todo especial a meu amigo Saulo Della Giustina que algumas horas esteve comigo durante toda a caminhada da faculdade sempre dando incentivos para que jamais desistisse desse caminho por mim escolhido.

Outro agradecimento especial vai para o amigo Alexandre Frare, que muito auxiliou nas horas em que já não estava mais encontrando soluções para determinados problemas e sempre mostrou-se disposto a ajudar.

Jamais poderia deixar de agradecer ao amigão Sandro dos Santos Carvalho, o qual, disponibilizou seu computador durante dois meses direto para que os mesmos pudessem estar interligados em rede para efetuar os testes do protótipo do TCC proposto.

A todos aqueles que de alguma forma estiveram presentes durante esses anos de graduação.

E ao grande criador **Deus**, que sem ele, jamais estaria concluindo esta etapa.

RESUMO

Este Trabalho de Conclusão de Curso (TCC) apresenta um estudo sobre gerência de redes de computadores enfocando as considerações do protocolo SNMP (S*imple Network Management Protoco*l) em redes locais de computadores. Apresenta também o desenvolvimento de um protótipo de software agente SNMP para o sistema operacional Windows.

ABSTRACT

This TCC assigment presents a study about computers Network management, which focus the Simple Network Management Protocol (SNMP) in computer local network. It also presents the prototype software developing agent SNMP for the windows operational system.

SUMÁRIO

ÍNDICE DE FIGURAS

ÍNDICE DE TABELAS

LISTA DE SIGLAS E ABREVIATURAS

API - *Application Program Interface*

ASN.1 - *Abstract Syntax Notation. One*

CMIP - *Communication Management Information Protocol*

EGP - *Exterior Gateway Protocol*

FDDI - *Fiber Distributed Data Interface*

FTP - *File Transfer Protocol*

GUI - *Graphical User Interface*

IAB - *Internet Activitie Board*

IP - *Internet Protocol*

ISSO - *International Organization for Standartization*

LAN - *Local Area Network*

MAN - *Metropolitan Area Network*

Mbps - *Megabits por segundo*

MIB - *Management Information Base*

NMS - *Network Management Station*

OSI - *Open System Interconnection*

PDU - *Protocol Data Unit*

RFC - *Request for Comment*

RMON - *Remote Monitoring*

SMI - *Structure of Management Information*

SMTP - *Simple Mail Transfer Protocol*

SNMP - *Simple Network Management Protocol*

TCP - *Transmission Control Protocol*

UDP - *User Datagram Protocol*

WAN - *Wide Area Network*

1. INTRODUÇÃO

Através dos avanços das tecnologias de interconectividade e dos benefícios proporcionados pelas Redes de Computadores, cada vez mais computadores são interconectados nas organizações. Paralelamente, a diminuição dos custos dos equipamentos permite adquirir e agregar à rede cada vez mais equipamentos, de tipos diversos, tornando essas redes cada vez maiores e mais complexas. Com isso, as redes começaram a ser interconectadas muito rapidamente; redes locais conectadas a redes regionais, as quais, por sua vez, ligadas a backbones nacionais.

Hoje essas redes são extremamente importantes para o dia-a-dia de muitas empresas em todo o mundo, porque normalmente, junto com sua utilização, vem a eficácia e a competitividade. Essa importância vem crescendo de tal forma que as empresas têm se tornado altamente dependentes destas redes, sentindo imediatamente o impacto quando os seus recursos não estão disponíveis (Cohen, 2000).

Este novo ambiente originou alguns problemas administrativos. As tarefas de configuração, identificação de falhas e controle de dispositivos e recursos da rede passaram a consumir tempo e recursos das organizações.

Porém, quanto mais informações trafegam pela rede ou mais se faz uso da mesma em um determinado período de tempo, a rede pode "sofrer" com o desempenho. Quando poucos usuários utilizam a rede para trocar informações entre suas estações de trabalho, ou qualquer outra atividade através da rede, o tráfego de informações ainda poderá ser pequeno e não comprometer o desempenho da rede, mas quando vários usuários utilizam-se destes recursos poderá haver uma queda de desempenho significativa através da sobrecarga da rede (*overhead*), prejudicando os serviços e a operacionalidade da rede (Rekowsky,1999).

Cientes deste problema, a solução então passou a ser buscada na atividade de Gerenciamento de Rede. Esta atividade passou a evoluir de forma rápida e concisa, sendo hoje uma das especialidades da área de Redes de Computadores que mais cresce.

Ao longo do seu desenvolvimento muitas definições foram propostas para a Gerência de Redes. Resume-se a seguir algumas dessas definições (Cohen 2000).

1. Consiste no controle e administração de forma racional dos recursos de hardware e software em um ambiente distribuído, buscando melhor desempenho e eficiência do sistema.
2. Consiste no controle de uma rede de computadores e seus serviços.
3. Tem por objetivo maximizar o controle organizacional das redes de computadores, de maneira mais eficiente e confiável, ou seja, planejar, supervisionar, monitorar e controlar qualquer atividade da rede.
4. Consiste na detecção e correção de falhas, em um tempo mínimo, e no estabelecimento de procedimentos para a previsão de problemas futuros.

Para atender a esta necessidade de gerenciamento foram desenvolvidos protocolos de gerenciamento. A principal preocupação de um protocolo de gerenciamento é permitir às aplicações de gerência de rede realizar tarefas tais como: obter dados sobre o desempenho e tráfego da rede em tempo real, diagnosticar problemas de comunicação e reconfigurar a rede atendendo às mudanças das necessidades dos usuários e o ambiente (Chisane, 1999). Porém, vários obstáculos teriam que ser superados, entre eles a heterogeneidade dos equipamentos de rede (computadores, roteadores, hubs e dispositivos de meio), dos protocolos de comunicação e das tecnologias de rede.

Segundo (Stallings, 1996), as informações de gerenciamento permitem, dentre outras tarefas, produzir registros de auditoria para as conexões, desconexões, falhas e outros eventos significativos na rede. Esses registros por sua vez, permitem determinar futuras necessidades de adicionar equipamentos, identificar e isolar erros comuns de um cliente, além de estudar outras tendências de uso. Deve-se destacar que estas informações devem possibilitar uma atuação preventiva e, não meramente reativa com relação aos problemas.

OBJETIVO DO TRABALHO

O objetivo deste trabalho é desenvolver um protótipo de software agente SNMP o qual responderá a requisições da estação de gerenciamento, podendo ser o envio de informações de gerência ou ações sobre as variáveis do dispositivo da rede onde atua.

1.1 ÁREA

Este trabalho de conclusão de curso tem como área principal a área de *Redes de Computadores* e como sub-área, enfoca a *Gerência de Redes de Computadores*.

1.2 ORGANIZAÇÃO DO TRABALHO

Este Trabalho de Conclusão de Curso está organizado conforme a seguir.

O capítulo 1 introduz o contexto geral do trabalho dividido nos segmentos de Introdução, Objetivos e Área do TCC proposto.

O capítulo 2 enfoca as redes de computadores, considerações de gerenciamento de redes e arquitetura de redes relevantes para a fundamentação deste TCC.

O capítulo 3 fundamenta sobre considerações do protocolo SNMP, bem como suas extensões, a base de informações de gerenciamento, operações aplicadas as variáveis da MIB, serviços exigidos pelo protocolo e formato das mensagens.

No capítulo 4 é descrito a especificação do funcionamento do protótipo, linguagens livres utilizadas (fluxogramas), sistema operacional e linguagem de programação, bem como telas de funcionamento do protótipo.

O capítulo 5 relata as conclusões obtidas na fundamentação deste TCC e sugestões de continuidade de trabalhos futuros a serem realizados.

Para finalizar este TCC, são apresentadas, no capítulo 6 as referências bibliográficas utilizadas durante o período de desenvolvimento deste Trabalho de Conclusão de Curso.

2. REDES DE COMPUTADORES

A evolução dos equipamentos de informática não foi algo que aconteceu rapidamente. Os equipamentos estão ficando cada vez mais potentes e de auxílio quase indispensável para as empresas e instituições. Cada vez mais utiliza-se computadores para agilizar tarefas do dia a dia através de ferramentas sofisticadas. Para muitas empresas surgiu a necessidade de que os computadores estivessem ligados de alguma forma aos demais equipamentos existentes dentro da corporação para otimizar tempo e recursos como impressoras e unidades de *backup*, não sendo privilégios de alguns, e sim, estando disponível para todos os usuários.

Uma grande parte dos usuários de redes não entendem porque precisam dela e, perguntam-se por que não podem ter os equipamentos desejados em suas máquinas. A falta de treinamento destes usuários da rede, também pode levar ao uso inadequado da mesma, sobrecarregando-a e consequentemente deixando-a mais lenta. Outro aspecto importante é que na maioria dos casos, as redes não estão somente instaladas em ambientes de trabalho que tem a informática como seu principal foco de trabalho (Sztajnberg, 1996).

Segundo (Soares, 1995) e (Tanembaum, 1994), existem basicamente três tipos de tecnologias para redes de computadores:

a) Redes Locais (*Local Area Netwoks - LANs*)

As LANs emergiram para possibilitar o compartilhamento de recursos (hardware e software) e informações, bem como a troca de informações entre vários computadores, mantendo a independência entre as diversas estações de trabalho conectadas à rede e possibilitando a integração destas estações e seus recursos em ambientes de trabalho corporativos (Soares, 1995) e (Tanembaum,1994).

Segundo (Soares, 1995) e (Tanembaum, 1994), uma rede local é uma rede que possibilita a interconexão de equipamentos de comunicação de dados numa pequena região. As LANs ainda caracterizam-se por não utilizarem recursos de telecomunicação

como meio de transmissão entre seus nós, que correspondem as próprias estações dos usuários conectados.

Entre as características básicas das LANs segundo (Soares,1995) e (Tanembaum,1994) estão:

a) Abrangência geográfica limitada (distâncias menores que 25 Km);
b) Altas taxas de transmissão;
c) Baixas taxas de erro;
d) Possuem pequenos atrasos de transmissão;
e) Geralmente são redes privadas;
f) Facilidade de interconexão entre redes distintas.

b) Redes Metropolitanas (Metropolitan Area Networks - MANs)

Uma rede metropolitana possui características similares as LANs, mas difere-se por cobrir uma área física maior que as LANs. Uma MAN pode cobrir praticamente uma cidade inteira (Tanembaum, 1994) e (Soares, 1995).

Como a mais nova das três tecnologias de rede apresentadas, as MANs surgiram com o objetivo de viabilizar o compartilhamento de recursos de hardware e software entre redes (Soares,1995) e (Tanembaum,1994).

c) Redes Geograficamente Distribuídas (Wide Area Networks - WANs)

Segundo (Soares,1995) as WANs são as pioneiras entre as redes de computadores. São constituídas de uma diversidade de aplicações e usos, destacando-se as redes públicas, isto é, "o sistema de comunicação é mantido, gerenciado e de propriedade de grandes operadoras (públicas ou privadas) e seu acesso é público".

As WANs possuem um custo elevado de comunicação, por isso a taxa de transmissão é baixa, mesmo que hoje em dia alguns *links* cheguem a velocidade de *megabits* por segundo (Mbps). Links são conexões de software entre um *driver* (parte da

placa de rede que cuida da comunicação entre a placa de rede e o protocolo de transporte) da placa de rede e um protocolo de transporte de rede

2.1. GERENCIAMENTO DE REDES

Com a utilização de microcomputadores e com o aumento de seu uso em ambientes de trabalho observou-se a necessidade de compartilhar os diversos recursos disponíveis como impressoras, unidades de CD-ROM, unidades de disco rígido, entre outros, de tal forma que tornassem-se comunitários para os usuários do ambiente de trabalho, otimizando tempo e recursos.

De acordo com (Brisa, 1994), o gerenciamento de redes é o conjunto de funções que visa promover a produtividade da planta e dos recursos disponíveis a interagir, de forma organizada, as funções de operação, administração e manutenção de todos os elementos da rede e os serviços de telecomunicações.

A gerência está associada ao controle de atividades e ao monitoramento do uso de recursos da rede. As tarefas básicas de gerência em redes, são obter informações da rede, tratar estas informações possibilitando um diagnóstico e, encaminhar as soluções dos problemas (Sztajnberg, 1996).

O contínuo crescimento em número, diversidade dos componentes das redes de computadores e a busca constante por uma maior produtividade, tem tornado cada vez mais necessária a atividade de gerenciamento de redes, de forma a obter um melhor aproveitamento e otimização dos recursos disponíveis.

Com o crescimento e a modernização das tecnologias de comunicação, o gerenciamento de desempenho em redes torna-se cada vez mais difícil. Aplicativos exigem respostas das suas requisições em tempo menor, interagindo com módulos distintos distribuídos pelos computadores interligados à rede. As aplicações distribuídas como são denominados estes aplicativos, fazem com que exija-se maior confiabilidade e modernidade dos equipamentos. Para que, com estas exigências dos aplicativos, o

desempenho da rede não seja comprometido, é necessário que seja implementada uma gerência inteligente da rede (Soares, 1995).

Uma gerência inteligente deve ser capaz de criar e manter um histórico das informações recebidas e ser capaz de identificar tendências e desvios das diretivas base.

Segundo (Rekowsky, 1999) e (Meirelles, 1999), gerenciamento de redes é o processo de utilizar hardware e software através de pessoal treinado para monitorar o *status* dos componentes da rede e cabeamento relacionado, questionar usuários finais e implementar ou recomendar ações para amenizar quedas, e/ou aumentar o desempenho de comunicação, bem como conduzir tarefas administrativas associadas com a operação da rede.

Os sistemas de gerenciamento atuais são baseados na interação de vários componentes da rede. A seguir podem ser vistos os principais componentes (Tanembaum, 1994).

a) ***dispositivos gerenciados***: são dispositivos de hardware como os computadores, roteadores e serviços de terminais que estão conectados a rede;

b) ***agentes***: são programas que residem nos elementos da rede que devem ser gerenciados. Os agentes coletam e armazenam diversas informações de gerenciamento;

c) ***base de informação de gerenciamento (Management Information Base - MIB)***: é uma estrutura de dados que contém uma relação dos objetos que podem ser gerenciados. Os dados contidos nesta estrutura são obtidos pelos agentes e armazenados nesta estrutura;

d) ***gerentes***: são eles que monitoram os dados obtidos sobre os diversos dispositivos da rede e disponibilizam estes dados já interpretados para o administrador da rede;

e) ***protocolos de gerenciamento***: através destes protocolos é possível estabelecer a interação entre os programas gerentes e agentes;

f) ***interfaces de programas aplicativos (Application Program Interface - API)***: através das APIs é possível interagir com o sistema operacional utilizado;

g) ***interfaces gráficas com o usuário (Graphical User Interface - GUI)***: através destas interfaces que a aplicação disponibiliza os dados e as informações para o usuário.

Apesar dos conceitos serem distintos existem alguns aspectos comum entre eles. Os autores destes conceitos mencionam que o aumento da complexidade das redes exige que seu gerenciamento seja feito de maneira simples, através de ferramentas de gerenciamento modernas e de fácil interação com o usuário.

O gerenciamento de redes e a necessidade de hierarquias e organização dos conceitos e estruturas resultaram em dois modelos de gerenciamento de redes: o modelo OSI e o modelo internet, mais conhecido como SNMP.

2.2. MODELO OSI PARA GERENCIAMENTO DE REDE

O gerenciamento no modelo *Open System Interconnection* (Interconexão de Sistemas Abertos - OSI) da *International Organization for Standartization* (Organização Internacional para Padronização - ISO) baseia-se na teoria da orientação a objetos. Através de entidades lógicas, também conhecidos como objetos gerenciados, é possível representar estes recursos gerenciados. Ao desenvolver uma aplicação de gerenciamento são utilizados processos distribuídos que são denominados de gerentes ou monitores, que são responsáveis pelo gerenciamento, e os agentes, também denominados de clientes, que efetuam ações (Mafinski, 1999), (Rekowsky, 1999) e (Sztajnberg, 1996).

Nele define-se um modelo funcional onde cada área tem um conjunto de funções, que ao serem implementadas, serão utilizadas para gerenciar a rede, onde encontram-se cinco áreas funcionais, como mostrado abaixo (Mafinski,1999), (Rekowsky, 1999) e (Sztajnberg, 1996).

a) ***Gerência de configuração***: representa o estado da rede;

b) ***Gerência de desempenho***: representa vazão e taxa de erros;

c) ***Gerência de falhas***: comportamento anormal da rede ou de seus segmentos;

d) ***Gerência de contabilidade***: analisa o consumo de recursos na rede;

e) ***Gerência de segurança***: controle e permissões de acesso à rede.

Relacionado ao modelo de gerenciamento OSI, com relação a sua estrutura, subdivide-se em três tipos principais (Sztajnberg, 1996)

a) ***Gerenciamento de Sistemas***: executado na camada de aplicação, precisa de apoio das sete camadas do modelo OSI para poder realizar a gerência e pode gerenciar qualquer objeto associado a um sistema aberto;

b) ***Gerenciamento de camada***: ocorre sobre os objetos gerenciados relacionados as atividades de uma camada específica e não depende dos protocolos de gerenciamento de outras camadas;

c) ***Operações de camada***: utilizada no gerenciamento de uma única instância de comunicação em uma camada e utiliza-se do protocolo normal da camada para a troca de informações, não precisando de um protocolo especial para poder realizar estas trocas de informações.

Com relação a MIB, pode-se dizer que ela guarda as informações transferidas ou modificadas pelo uso de protocolos de gerenciamento OSI. Tais informações podem ser fornecidas por agentes administrativos locais ou remotos. As operações sobre objetos gerenciados são definidas de maneira abstrata e seu detalhamento está fora do escopo

OSI, uma vez que o agente e os objetos gerenciados situam-se no mesmo ambiente local (Brisa, 1993) e (Sztajnberg, 1996).

No modelo OSI, existem três tipos de hierarquias de gerenciamento utilizadas pelo Sistemas de Gerenciamento: Herança, Nomeação e Registro. Estes três tipos serão demonstrados a seguir.

2.2.1. HIERARQUIA DE HERANÇA

A hierarquia de herança também denominada hierarquia de classe, tem como objetivo facilitar a modelagem dos objetos, através do paradigma da utilização da orientação a objetos. Assim podem ser definidas classes, superclasses, subclasses. Trata-se de uma ferramenta para uma melhor definição das classes.

2.2.2. HIERARQUIA DE NOMEAÇÃO

A hierarquia de nomeação, também chamada hierarquia de *containment*, descreve a relação de "estar contido em" aplicado aos objetos. Um objeto está contido dentro de um e somente um objeto gerenciado.

Este relacionamento de *containment* pode ser usado para modelar hierarquias de partes do mundo real (módulos, submódulos e componentes eletrônicos) ou hierarquias organizacionais (diretório, arquivos, registros e campos) (Mafinski, 1999).

Um objeto gerenciado existe somente se o objeto que contém existir e, dependendo da definição, um objeto só pode ser removido se aqueles que lhe pertencerem forem removidos primeiro.

2.2.3. HIERARQUIA DE REGISTRO

A hierarquia de registro é usada para identificar de maneira universal os objetos, independentemente das hierarquias de heranças e nomeação. Esta hierarquia é

especificada segundo as regras estabelecidas pela notação ASN.1 (Abstract Syntax Notation. One) para árvore de registros usada na atribuição de identificadores a objetos. Cada nó desta árvore está associado a uma autoridade de registro que determina como são atribuídos os seus números. Desta maneira, cada objeto é identificado por uma seqüência de números que correspondem a um nó da árvore (Mafisnki, 1999).

Ainda como parte do modelo de gerenciamento OSI, há o protocolo de informação de gerência de comunicação (*Communication Management Information Protocol* - CMIP).

Este protocolo possui comandos que permitem a requisição de informações ou ações sobre os objetos por parte do usuário. Alguns comandos para leitura de variáveis existentes são *get* e *response* e para modificar o seu valor o comando é *put*. A modificação do valor de uma variável pode implicar no disparo de um novo processo, bem como no reinicio de um processo que já estava em andamento.

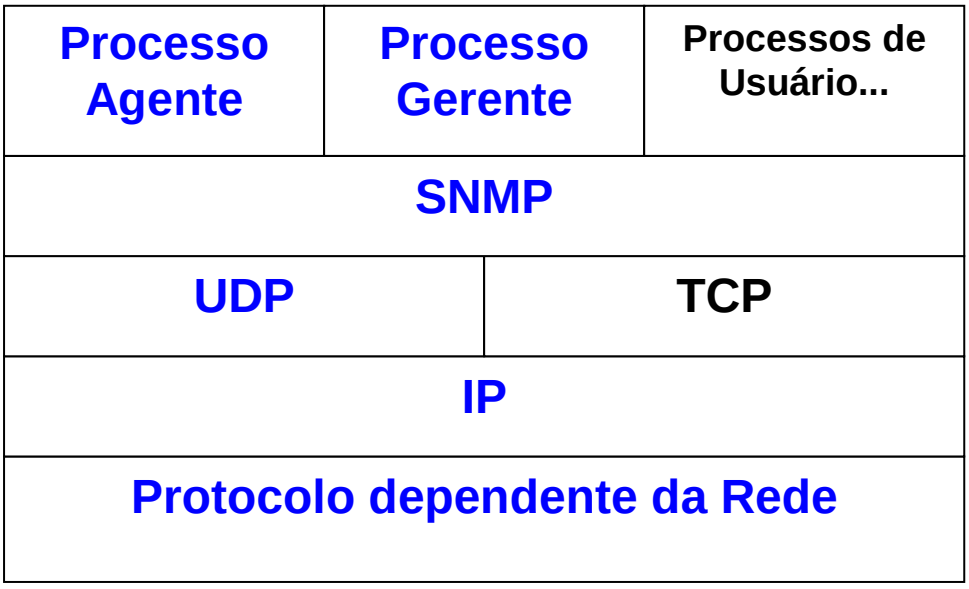
Para um melhor entendimento e detalhamento sobre o modelo de gerenciamento de redes OSI e sobre o CMIP, podem ser obtidas em (Brisa, 1993), (Brisa, 1994), (Soares, 1995), (Sztajnberg, 1996) e (Tanembaum, 1994).

3. MODELO DE GERENCIAMENTO SNMP

O SNMP foi desenvolvido nos anos 80 como resposta para os problemas de gerenciamento em ambiente TCP/IP Internet, envolvendo redes heterogêneas. Inicialmente foi concebido para ser apenas uma solução provisória até o desenvolvimento de um protocolo de gerenciamento mais completo, o CMIP (*Common Management Information Protocol*) . Desta forma, como não havia um protocolo melhor disponível, o SNMP passou a ser o protocolo mais utilizado.

O *Simple Network Management Protocol* (SNMP) é um protocolo da camada de aplicação, como mostrado na Figura 1, desenvolvido para facilitar a troca de informações de gerenciamento entre dispositivos de rede. Estas informações transportadas pelo SNMP (como pacotes por segundo e taxa de erro na rede), permitem aos administradores da rede gerenciar o seu desempenho de forma remota, encontrando e solucionando os problemas e, planejar seu crescimento.

Figura 1: Protocolo SNMP sobre as camadas do TCP/IP.



Atualmente existem três versões de SNMP: o SNMPv1 o SNMPv2 e o SNMPv3. O SNMPv3 corrige problemas de segurança encontrados nas primeiras versões do SNMP, além de adicionar novas funcionalidades, onde destacam-se a

capacidade de gerenciamento distribuído, via primitivas de comunicação gerente-gerente, e as formas de tratamento e transporte dos dados.

Hoje o SNMP é o protocolo mais implementado nos produtos comerciais de gerenciamento assim como em universidades e organizações de pesquisa. Isso graças ao fato de o SNMP ser um protocolo relativamente simples de ser utilizado. Esta simplicidade torna mais fácil para o usuário programar as variáveis que gostaria de gerenciar. Para o SNMP cada variável consiste das seguintes informações:

a) um título que identifica a variável.

b) a estrutura de dado da variável (inteiro, string,...)

c) o status da variável (read-only ou read-write)

d) o valor da variável.

Outra vantagem do SNMP é a sua vasta utilização atualmente. Esta popularidade deve-se ao fato de que até agora nenhum outro protocolo melhor apareceu para substituir a implementação "provisória" do SNMP. O resultado disso é que a maioria dos vendedores de hardware/software para redes desenvolvem seus produtos para suportar o SNMP como protocolo de gerenciamento padrão.

A grande expansibilidade é outro benefício do SNMP. A relativa facilidade de implementação torna também mais fácil a atualização deste protocolo, visando atender as novas necessidades dos usuários de redes. Porém, o SNMP não é um protocolo de gerenciamento perfeito. Ele também possui falhas (Cohen, 2000).

A primeira deficiência do SNMP padrão está nas brechas de segurança que podem permitir o acesso de intrusos às informações transportadas pela rede. Esses intrusos podem, portanto, acessando estas informações, potencialmente, retirar algumas máquinas da rede.

A solução para este problema, no entanto, é simples. A expansão do SNMP, a versão - SNMPv3, adiciona alguns mecanismos de segurança que auxiliam no combate

dos três maiores problemas de segurança: a privacidade dos dados (previne que intrusos tenham acesso às informações de gerenciamento transportadas pela rede), autenticação (previne que intrusos enviem dados falsos através da rede) e controle de acesso (restringe o acesso a determinadas variáveis para certos usuários, reduzindo a possibilidade de um usuário , acidentalmente, danificar a rede).

3.1. EXTENSÕES AO SNMP

Com sua facilidade de implementação, livrando os desenvolvedores das dificuldades de compatibilidade do modelo OSI, o protocolo SNMP progrediu rapidamente, tornando-se cada vez mais atraente para o gerenciamento de redes TCP/IP. Com isso, o SNMP ganhou um grande número de implementações por parte de diversos fabricantes, tornando-se o protocolo mais difundido para o gerenciamento de redes.

Atualmente, o SNMP básico é largamente utilizado. A maior parte dos fabricantes de computadores hosts, estações, roteadores e hubs já oferecem uma implementação deste protocolo.

Com a grande popularidade do SNMP, várias extensões têm sido propostas. Talvez a mais importante dessas iniciativas seja o desenvolvimento da capacidade de monitoramento remoto ao SNMP. A especificação *Remote Monitoring* (RMON) define capacidades adicionais à MIB convencional do SNMP, além de funções que exploram a MIB RMON. O RMON possibilita ao gerente da rede monitorar as subredes como um todo único. Fornecedores e usuários vêem no RMON uma extensão essencial ao SNMP. Mesmo sendo relativamente novo, o RMON já está sendo amplamente explorado, possuindo várias implementações (Rekowski, 1999).

Além da RMON, várias outras extensões à MIB do SNMP têm sido desenvolvidas. Algumas dessas extensões buscam compatibilizar o SNMP com outras padronizações, como token ring e FDDI. Outras extensões são específicas de determinados fornecedores.

3.2. ELEMENTOS DO SNMP

Segundo (Brisa, 1994) e (Tanembaum, 1994), classificam-se como elementos do protocolo SNMP agentes, gerentes e a base de informações de gerenciamento (MIB) que serão descritos a seguir.

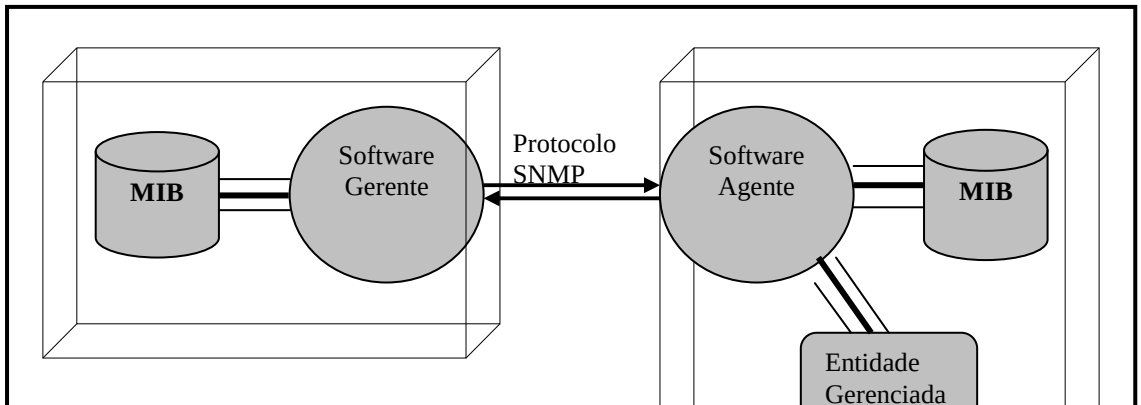
3.2.1. AGENTES

No modelo de gerenciamento SNMP, hosts, pontes, roteadores e hubs devem ser equipados com agentes SNMP para que possam ser gerenciados pela estação de gerenciamento (NMS) através do gerente SNMP. O agente responde a requisições da estação de gerenciamento, que pode ser o envio de informações de gerência ou ações sobre as variáveis do dispositivo onde está.

O funcionamento desta estrutura só é possível graças ao acesso direto à MIB (*Management Information Base*) que o agente possui, pois todas as informações de gerência encontram-se lá. Ao receber uma mensagem SNMP do gerente, o agente identifica que operação está sendo requisitada e qual(is) a(s) variável(is) relacionadas, e a partir daí executa a operação sobre a MIB ; em seguida, monta uma nova mensagem de resposta, que será enviada ao gerente.

A relação entre agente, gerente e MIB pode ser visualizada no esquema simplificado a seguir.

Figura 2: Esquema simplificado da relação Agente, Gerente e MIB



À primeira vista a comunicação do agente com o gerente pode parecer injusta, uma vez que o agente apenas responde ao que lhe é questionado. Mas há momentos em que o agente pode "falar" com o gerente sem que antes seja questionado. Isso ocorre quando o agente detecta, a partir da análise do contexto da MIB, alguma situação inesperada (Tanembaum, 1994). Neste momento, o agente gera uma mensagem especial, o Trap, e a envia ao gerente, relatando sobre a situação.

Para o tratamento destas exceções e o envio de Traps, é concedido ao agente um certo poder de decisão, cabendo a ele, a partir da análise do contexto da MIB, decidir se é ou não necessário enviar o Trap ao gerente. Esse poder de decisão é concedido ao agente para que em certas situações, como quando da inicialização do sistema, Traps desnecessários não sejam trafegados pela rede, o que, quando se tratando de dezenas de agentes, poderia interferir no desempenho global da rede.

Cabe ao agente um papel fundamental em todo o processo de gerenciamento da rede, acessando e disponibilizando informações de gerência contidas na MIB, além de indicar situações inesperadas de funcionamento do dispositivo ao qual gerencia através do envio de Traps ao gerente.

Segundo (Brisa, 1994) e (Meirelles, 1999), o código de um agente é constituído de pelo menos três funções básicas:

a) Um núcleo para implementação e tratamento dos protocolos de gerência;

b) Um módulo de interface com o elemento de rede para coleta das informações;

c) Um módulo para montagem e disponibilização da estrutura de informação definida para o elemento.

A operação Trap difere de todas as outras. Ela é utilizada por um agente SNMP para notificar de forma assíncrona a um gerente algum evento extraordinário que tenha ocorrido no objeto gerenciado.

Diversos questionamentos são feitos quanto a esta operação. Talvez o maior deles seja: Quais eventos devem realmente ser notificados ao gerente? Embora todos concordem que o gerente deva ser informado de alguns eventos significativos, muitos fornecedores de produtos que implementam o SNMP trazem Traps específicos, muitos deles desnecessários (Sztajnberg, 1996).

Outro importante questionamento é: Já que a operação Trap não gera um Response, como um agente pode ter certeza de que o gerente o recebeu?. Esta preocupação se perde quando a rede está em perfeito funcionamento. Mas e se a máquina estiver por exemplo, perdendo pacotes? Isso não é fácil de solucionar. Uma possibilidade seria o agente gerar tantos Traps quanto necessário até que o gerente seja informado do evento. Essa solução, no entanto, aumentaria o tráfego na rede, afetando o seu desempenho.

3.2.2. GERENTES

Servindo de interface entre as aplicações de gerência correntes no NMS e os agentes espalhados pelos dispositivos da rede está o gerente. Cabe ao gerente enviar comandos aos agentes, solicitando informações sobre variáveis de um objeto gerenciado, ou modificando o valor de determinada variável.

Os gerentes então processam estas informações colhidas pelos agentes, e as repassam à aplicação que as requisitou. A comunicação entre o gerente e as aplicações é possível através da utilização das APIs do gerente SNMP pelo sistema.

A API (*Application Program Interface*) é um conjunto de funções que intermediam a execução de comandos entre um programa e outro, de forma a simplificar a um programa o acesso a funções do outro programa, e que no caso do SNMP intermediam as execuções entre uma aplicação de gerência e o gerente SNMP (Tanembaum, 1994).

Quando uma solicitação da aplicação de gerência chega ao gerente, através da API, o gerente mapeia esta solicitação para um ou mais comandos SNMP que, através da troca de mensagens apropriadas, chegarão aos agentes correspondentes.

Este comando SNMP montado pelo gerente pode ser enviado a um outro gerente, que pode ou não repassá-lo a um agente. Só que a comunicação gerente-gerente só é possível na versão estendida do SNMP, o SNMPv3, e não faz parte do SNMP padrão (Meirelles, 1999).

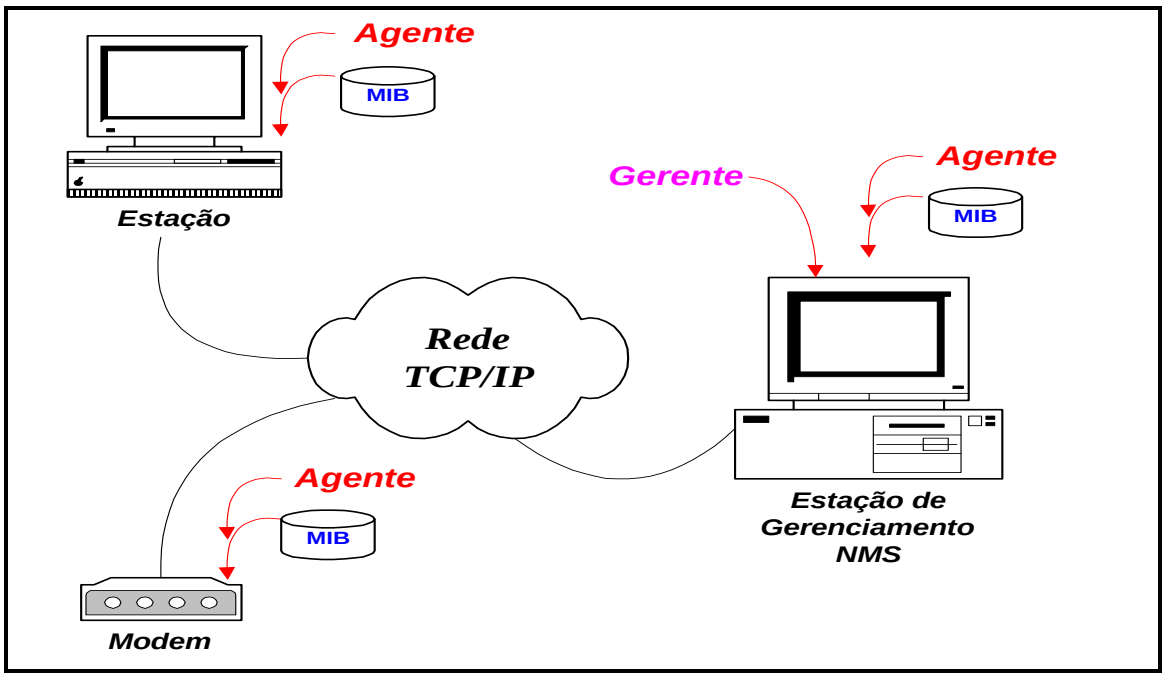
Cabe também ao gerente encaminhar à aplicação de gerência os Traps que porventura sejam enviados pelos agentes. Assim, o software de gerência terá conhecimento da presença de um novo equipamento na rede ou do mal funcionamento de algum dos dispositivos da rede.

Quando da inicialização do software de gerência, nada se sabe sobre a configuração ou funcionamento da rede. Essas informações vão sendo conhecidas através de Traps que são enviados pelos agentes; a partir daí o gerente realiza *polling* (Tanenbaum, 1994) a fim de manter a comunicação com os agentes, possibilitando ao software de gerência mapear, monitorar e controlar a rede.

3.2.3. MANAGEMENT INFORMATION BASE (MIB)

No modelo SNMP, os recursos de uma rede são representados como objetos. Cada objeto é, essencialmente, uma variável que representa um aspecto do dispositivo gerenciado. Todos os objetos (variáveis) são armazenados na *Management Information Base* (MIB), como mostrado na figura abaixo.

Figura 3: Elementos básicos do SNMP



Fonte: (Cohen, 2000)

3.2.4. ESTRUTURA DA INFORMAÇÃO DE GERENCIAMENTO (SMI)

A Estrutura da Informação de Gerenciamento (SMI - *Structure of Management Information*) é especificada na RFC 1155 e define a estrutura através da qual uma MIB pode ser definida e construída. A SMI define os tipos de dados que devem ser usados na MIB e especifica como os recursos na MIB são representados e nomeados.

A SMI evita tipo de dados complexos para simplificar a tarefa de implementação e aumentar a interoperabilidade. Inevitavelmente, as MIBs desenvolvidas por fornecedores irão conter tipo de dados particulares, a menos que restrições sejam impostas na definição de cada tipo de dado a fim de manter a interoperabilidade.

Para fornecer um caminho de padronização na representação das informações de gerenciamento, a SMI deve fornecer o seguinte:

a) Padronização técnica na definição estrutural de uma MIB;

b) Padronização técnica na definição de objetos individuais, incluindo a sintaxe e o valor de cada objeto;

c) Padronização técnica na codificação dos valores dos objetos.

3.2.4.1. TIPOS DE DADOS

A definição dos objetos (variáveis) numa MIB do SNMP padrão, contém os seguintes tipos de dados:

a) ***Integer***, ***OCTET STRING***, ***NULL***, ***OBJECT IDENTIFIER*** e ***sequence*** - são tipos universais, de uso geral.

b) ***ipAddress*** - Um endereço de 32 bits utilizando o formato IP.

c) ***Counter32*** - Um inteiro positivo que pode ser incrementado, mas nunca decrementado. Seu valor máximo é 2^{32-1} (4.294.967.295); quando atingir seu valor máximo é reiniciado em zero.

d) ***Gauge32*** - Um inteiro positivo que pode ser incrementado e decrementado. Seu valor máximo é o mesmo do *Counter32*; quando este valor máximo é alcançado ele não é reiniciado, pois pode ser decrementado.

e) ***TimeTicks*** - Este inteiro positivo conta, em milésimos de segundos, um determinado período.

f) ***Opaque*** - Este tipo permite suportar dados arbitrários. O dado é codificado como um OCTET STRING para transmissão.

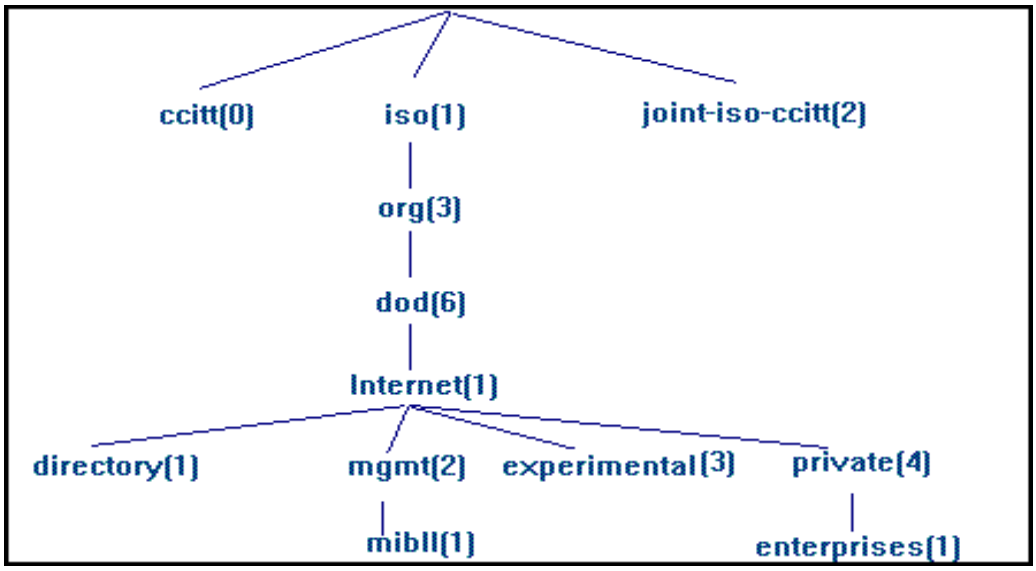
3.2.4.2. ESTRUTURA DA MIB

Todos os objetos gerenciados no modelo SNMP estão armazenados e dispostos numa estrutura hierárquica, mais conhecida como **árvore**. As folhas da árvore são os objetos gerenciados atuais, onde cada um representa algum recurso, atividade ou informação relacionada que deve ser gerenciada. A própria estrutura em árvore define o agrupamento de objetos em conjuntos relacionados. Esta estrutura pode ser visualizada na Figura 4.

Para cada tipo de objeto na MIB é associado um identificador de tipo ASN.1, o OBJECT IDENTIFIER. O identificador serve para nomear o objeto. Como o valor associado com o tipo OBJECT IDENTIFIER é hierárquico, a nomeação também serve para identificar a estrutura dos tipos de objeto.

O identificador é único para cada tipo de objeto. Seu valor consiste numa seqüência de inteiros. Iniciando pela raiz da árvore, cada parte inteira componente do valor do identificador representa um ramo da árvore. A partir da raiz, encontramos três ramos principais no primeiro nível da árvore: **iso**, **ccitt** e uma junção **iso-ccitt**. Sob o ramo iso uma sub-árvore é utilizada por outras organizações (org), uma das quais é o *U.S. Departament of Defense* (DoD).

Figura 4: Estrutura em árvore de uma MIB SNMP.



A especificação MIB define os grupos de variáveis necessárias à monitoração e controle de vários componentes da rede. Para o SNMP, nem todos os grupos de variáveis definidas pela especificação MIB são obrigatórios. Um dos mais importantes e obrigatório para o SNMP é o grupo Internet.

A SMI define quatro grupos principais dentro do grupo Internet:

a) ***directory***: é um grupo reservado, contém informações do diretório OSI (X.500);

b) ***mgmt***: é um grupo utilizado para objetos definidos em documentos da IAB (*Internet Activitie Board* - órgão que registra e normatiza as MIBs), e que compõem a MIB padrão SNMP. Atualmente, duas versões de MIB são disponíveis, MIB I e MIB II, onde a segunda é apenas uma extensão da primeira;

c) ***experimental***: é um grupo utilizado para identificar objetos de experiências com a Internet;

d) ***private***: é um grupo utilizado para identificar objetos definidos unilateralmente. Seu subgrupo *enterprises* serve para registrar as MIBs

estendidas. Estas MIBs estendidas são desenvolvidas por fornecedores, e contêm informações necessárias a outros usuários e fornecedores, a fim de permitir a interoperabilidade com o seu produto. Um ramo é criado na sub-árvore *enterprises* para cada fornecedor que registra um identificador de objeto *enterprise*.

Basicamente, são quatro os tipos de MIBs: MIB I, MIB II, MIB experimental e MIB privada, de acordo com (Meirelles, 1999) e (Sztajnberg, 1996).

As MIBs do tipo I e II fornecem informações gerais sobre o equipamento gerenciado, sem considerar as características específicas deste equipamento. A MIB II, é uma evolução da MIB I, na qual foram introduzidas novas informações além daqueles já contidas na MIB I, onde dentro destas MIBs pode-se obter informações como tipo e o *status* da interface (Ethernet, Token-Ring), número de pacotes transmitidos, pacotes com erros, informações sobre o protocolo de transmissão, entre outros.

As MIBs são definidas como grupos. Estes grupos, ou também denominados de categorias, são utilizados pelos identificadores para especificar itens através de um código para cada categoria, como mostrado nas tabelas I e II referentes as MIBs I e II respectivamente abaixo.

TABELA 1: Categorias de informações da MIB-I

Categoria da MIB	**Objetos contidos para**
System (sistema)	Informações básicas sobre o sistema operacional ou referentes ao *host*
Interfaces	Interfaces de rede
At (conv. end.)	Tradução de endereços
ip	Software do protocolo internet (IP)
Icmp	Software do protocolo estatístico para controle interno de mensagens
Tcp	Software do protocolo de controle de transmissão (TCP)
Udp	Software do protocolo de datagrama do usuário (UDP)
Egp	Software do protocolo de *gateway* externo (EGP)

TABELA 2: Categorias de informações da MIB-II

Categoria da MIB	Objetos contidos para
System (sistema)	Informações básicas sobre o sistema operacional ou referentes ao host
Interfaces	Interfaces de rede
At (conv. end.)	Tradução de endereços
Ip	Software do protocolo internet (IP)
Icmp	Software de protocolo estatístico para controle interno de mensagens
Tcp	Software de controle de transmissão (TCP)
Udp	Software do protocolo de datagrama do usuário (UDP)
Egp	Software do protocolo de *gateway* externo (EGP)
Transmission	Meio de transmissão
Snmp	Protocolo e aplicações SNMP

A construção das MIBs segue as especificações contidas nos documentos denominados *Structure of Management Information* (Informações de Gerenciamento de Estruturas - SMI) para as versões 1 e 2 do protocolo SNMP. Estas especificações impõem um grau de simplicidade elevado na descrição das MIBs para que a implementação destas, torne-se cada vez mais simples de ser realizada (Meirelles, 1999) e (Sztajnberg, 1996).

A lista definida de objetos gerenciáveis foi derivada daqueles elementos considerados essenciais. Esta implementação não é restrita, uma vez que a SMI proporciona mecanismos de extensão como uma nova versão de uma MIB e uma definição de um objeto privado ou que não seja padrão (Sztajnberg, 1996).

A seguir são listados alguns exemplos de grupos e seus objetos:

a) Grupo System

- sysDescr: completa descrição do sistema (versão, hardware, sistema operacional);
- sysObjectID: objeto para identificação do vendedor;
- sysUpTime: tempo desde sua última reinicialização;
- sysContact: nome da pessoa para contato;
- sysServices: serviços oferecidos pelo sistema.

b) Grupo IP

- ipForwarding: indica se esta entidade é um gateway IP;
- ipInHdrErrors: número de datagramas recebidos descartados devido a erros em seu cabeçalho;
- ipInAddrErrors: número de datagramas recebido descartados devido a erros em seu endereço IP;
- ipReasmOKs: número de datagramas IP remontados com sucesso;
- ipRouteMask: máscara de sub-rede para rota.

c) Grupo TCP

- tcpRtoAlgorithm: algoritmo para determinar o timeout para retransmissão de um octeto desconhecido;
- tcpMaxconn: limite do número de conexões TCP que a entidade pode sustentar;
- tcpInSegs: número de segmentos recebidos incluindo aqueles recebidos com erro;
- cpConnRemAddress: endereço remoto IP para determinada conexão TCP;
- tcpInErrs: número de segmentos descartados devido ao formato de erro;
- tcpOutRsts: número de reinicializações geradas.

d) Grupo UDP

- udpInDatagrams: número de datagramas UDP entregues ao usuário UDP;
- udpNoPorts: número de datagramas UDP recebidos para aqueles onde não existe aplicação para aquela porta de destino;
- udpInErrors: número de datagramas UDP recebidos que não podem ser entregues por diversas razões, menos a falta de uma aplicação para a porta de destino;
- udpOutDatagrams: número de datagramas UDP enviados por esta unidade.

e) Grupo Interfaces

- IfIndex: número de interface;
- IfDescr: descrição da interface;
- IfType: tipo da interface;
- IfMtu: tamanho do maior datagrama IP;
- IfAdminisStatus: status da interface;
- IfLastChange: hora em que a interface inicializou o status corrente.

A MIB da Internet não inclui informações de gerenciamento para aplicações como *TELNET* (Acesso a Terminal Remoto), Transferência de Arquivos (FTP - *File Transfer Protocol*) e Correio Eletrônico (*Simple Mail Transfer Protocol* - SMTP).

3.3. OPERAÇÕES SNMP APLICADAS ÀS VARIÁVEIS DA MIB

No modelo de gerenciamento SNMP, para que haja a comunicação entre agentes e gerentes são necessárias algumas operações que possibilitam o acesso às variáveis contidas na MIB, como listadas a seguir:

3.3.1. GET

O gerente SNMP envia o comando Get a um determinado agente toda vez que necessita recuperar uma informação de gerenciamento específica do objeto gerenciado pelo agente. Estas informações encontram-se na forma básica de variáveis, que por sua vez estão na *Management Information Base* (MIB) do elemento de rede gerenciado. Logo, o comando Get, na verdade, solicita ao agente a leitura de determinada(s) variável(is) da MIB em questão.

3.3.2. GETNEXT

O comando GetNext assemelha-se ao comando Get, no entanto, enquanto o comando Get solicita ao agente a leitura de determinada instância de uma variável, ao receber um comando GetNext, o agente deve ler a próxima instância disponível, na ordem da MIB, da(s) variável(is) associadas.

Esta operação é especialmente poderosa na recuperação de uma tabela de instâncias da MIB, cujo tamanho seja desconhecido. Para isto, é inicialmente especificado no comando GetNext o nome da tabela, o que retornará como resposta o primeiro item (neste caso, uma instância de variável) da tabela. Com o nome do primeiro item da tabela, o gerente pode enviar um outro GetNext, desta vez passando a resposta do GetNext anterior como parâmetro, a fim de recuperar o próximo item da tabela, e assim sucessivamente até recuperar todas as instâncias da variável na tabela. Para este caso, se o comando Get fosse utilizado ele falharia, pois o nome da tabela não corresponde a uma instância individual da variável.

3.3.3. SET

A operação Set requisita a um determinado agente a atribuição/alteração do valor de determinada(s) variável(is) de uma MIB. Alguns desenvolvedores acreditam que este comando não deve retornar um Response. Outros acham que a operação Set deve retornar alguma indicação de que a operação foi efetuada. Porém o mais correto seria que após cada operação Set sobre uma variável, uma operação Get fosse efetuada sobre a mesma variável a fim de assegurar que a operação Set foi efetuada.

3.3.4. TRAP

A operação TRAP difere de todas as outras. Ela é utilizada por um agente SNMP para notificar de forma assíncrona a um gerente algum evento extraordinário que tenha ocorrido no objeto gerenciado.

3.3.5. RESPONSES

Sempre que um agente recebe um comando Get, GetNext ou Set ele tenta executar a operação associada e, conseguindo ou não, constrói uma outra mensagem que é enviada ao emissor da requisição. Esta mensagem é a GetResponse. Das operações SNMP, apenas o Trap não gera um Response (Sztajnberg, 1996).

3.4. SNMP SOBRE A CAMADA DE TRANSPORTE

O protocolo SNMP foi desenvolvido para rodar sobre a camada de transporte, na camada de aplicação da pilha do protocolo TCP/IP. A maioria das implementações do SNMP utilizam o *User Datagram Protocol* (UDP), como protocolo de transporte. O UDP é um protocolo não-confiável, não garantindo a entrega, a ordem ou proteção contra duplicação das mensagens.

O SNMP utiliza o UDP pois foi desenvolvido para funcionar sobre um serviço de transporte sem conexão. A maior razão para isto é o desempenho. Utilizando um

serviço orientado à conexão, como o TCP, a execução de cada operação necessitaria de uma prévia conexão, gerando um *overhead* significativo, o que é inaceitável na atividade de gerência onde as informações devem ser trocadas entre as entidades da forma mais rápida possível.

Duas portas UDP são associadas as operações SNMP, as quais, logicamente funcionam como um meio de comunicação entre as estações. As operações Get, GetNext, GetBulk, Inform, e Set utilizam a porta 161. Já a operação Trap, por ser acionada em casos de exceção, tem reservada para si a porta 162 (Tanembaum, 1994).

Segmentos UDP são transmitidos em datagramas IP. O cabeçalho UDP inclui os campos de origem e destino, e um *checksum* opcional que protege o cabeçalho UDP e os dados de usuário (em caso do *checksum* ser violado, a PDU é descartada).

Como o UDP é um protocolo não-confiável, é possível que mensagens SNMP sejam perdidas. O SNMP por si só não fornece garantia sobre a entrega das mensagens. As ações a serem tomadas quando da perda de uma mensagem SNMP não são abordadas pelo padrão. No entanto, algumas considerações podem ser feitas sobre a perda de mensagens SNMP.

No caso de uma mensagem de Get, GetNext ou GetBulk, a estação de gerenciamento deve assumir que esta mensagem (ou o GetResponse correspondente) tenha sido perdida se não receber resposta num certo período de tempo. A estação de gerenciamento então pode repetir a mensagem uma ou mais vezes até que obtenha a resposta. Desde que um único *request-id* (operação que identifica a mensagem) seja utilizado para cada operação distinta, não haverá dificuldade em identificar mensagens duplicadas.

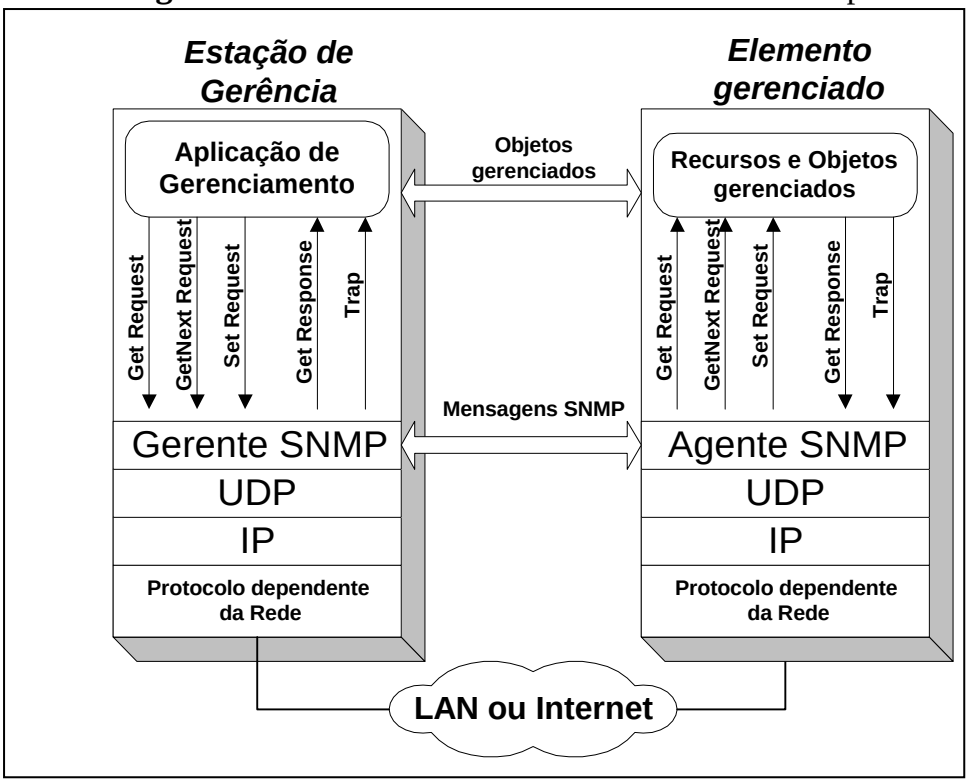
No caso de uma mensagem de Set, a recuperação deve provavelmente envolver o teste no objeto associado à operação através de uma Get sobre este mesmo objeto a fim de determinar se a operação Set foi ou não efetivada.

Uma vez que a operação Trap não gera uma mensagem de resposta, não é fácil identificar a perda de uma mensagem desse tipo.

No SNMP o gerente executa o gerenciamento através da utilização do protocolo SNMP, que é implementado no topo das camadas UDP, IP e do protocolo dependente do tipo de rede (Ethernet, FDDI, X.25, por exemplo).

A figura 5 ilustra o contexto do protocolo SNMP na pilha de protocolo TCP/IP, utilizando o UDP como protocolo de conexão.

Figura 5: Protocolo SNMP sobre a camada de transporte.



3.4.1. SERVIÇOS EXIGIDOS PELO SNMP

Rodando na camada de Aplicação da arquitetura TCP/IP, como mostrado na Figura 5, o SNMP encontra-se acima das camadas de Transporte e Rede. Para que o SNMP possa funcionar adequadamente, estas camadas (Transporte e Rede) devem fornecer ao menos cinco serviços que são de vital importância à movimentação das mensagens SNMP através da rede:

3.4.1.1 ROTEAMENTO

A camada de Rede é quem fornece as funções de roteamento, as quais aperfeiçoam toda a utilidade do gerenciamento da rede, dando a possibilidade de “re-rotear” os pacotes em torno de áreas da rede com falhas. Isso permite que o gerenciamento da rede continue operando mesmo após falha em alguma(s) máquina(s) na rede.

3.4.1.2. INDEPENDÊNCIA DO MEIO

Permite ao SNMP gerenciar diferentes tipos de elementos da rede, de forma independente. Caso o SNMP fosse construído sobre a camada de enlace, estaria preso a esta implementação de enlace específica, desse modo o SNMP não poderia gerenciar um outro dispositivo que implementasse outro protocolo de enlace, o que degradaria o gerenciamento da rede.

3.4.1.3. FRAGMENTAÇÃO E REMONTAGEM

Este serviço está relacionado com a independência do meio. A mensagem SNMP pode ser fragmentados em pacotes, transmitida pelo meio e remontada no destino. O protocolo IP permite que os pacotes SNMP trafeguem pelo meio com diferentes tamanhos. A Fragmentação e a Remontagem reduzem a robustez geral do gerenciamento da rede, pois se qualquer fragmento for perdido pelo caminho, toda a operação irá falhar.

3.4.1.4. CHECKSUM PONTO-A-PONTO

O Checksum ponto-a-ponto é um serviço de checagem de erros fornecido pela camada de Transporte que permite a uma entidade conferir a integridade dos dados recebidos através da rede, aumentando a confiabilidade da transmissão.

3.4.1.5. MULTIPLEXAÇÃO/DEMULTIPLEXAÇÃO

Os serviços de Multiplexação e Demultiplexação são fornecidos pelo protocolo de Transporte. Estes serviços facilitam em muito os possíveis relacionamentos de gerenciamento com o SNMP, permitindo que várias aplicações SNMP possam “utilizar” serviços da camada de transporte.

3.5. FORMATO DA MENSAGEM SNMP

No modelo de gerenciamento SNMP, as informações são trocadas entre os gerentes e agentes na forma de mensagens. Cada mensagem possui duas partes, um cabeçalho e uma *Protocol Data Unit* (PDU). O cabeçalho inclui um número de versão (*version*) que indica a versão do SNMP e um nome de comunidade (*community*). O nome de comunidade possui duas funções. Primeiro, define um dispositivo de acesso para um grupo de NMSs. Segundo, aqueles dispositivos cujo nome de comunidade é desconhecido são excluídos de operações SNMP; os gerentes também podem utilizar o nome de comunidade como uma forma de autenticação. O campo PDU pode conter qualquer um dos tipos de PDUs utilizados pelo SNMP (SNMP PDU). Esta estrutura pode ser visualizada na Figura 6 mostrada abaixo.

Figura 6: Formato da mensagem SNMPv1

.

As PDUs GetRequest PDU, GetNextRequest PDU e SetRequest PDU são definidas no SNMP com o mesmo formato da GetResponse PDU com os campos *error-status* e *error-index* com valor zero. Essas PDUs possuem os seguintes campos (Oda, 1998):

a) ***version*** - Indica a versão do protocolo SNMP utilizado.

b) ***community*** - O nome de comunidade atua como uma senha para autenticar a mensagem SNMP.

c) ***SNMP PDU*** - É a unidade de dados de protocolo (*Protocol Data Unit* - PDU) utilizada pelo SNMP; contém os dados referentes a operação desejada (Get, GetNext etc). Sua forma mais geral inclui os seguintes campos:

- ***PDU type*** - indica o tipo de PDU, neste caso pode ser uma GetRequest PDU, uma GetNextRequest PDU, uma SetRequest PDU ou uma GetResponse PDU.

- ***request-id*** - Usado para identificar o *request*; essa mesma identificação será utilizada na resposta a esta mensagem.

- ***error-status*** - É um sinalizador utilizado para indicar que uma situação inesperada ou erro ocorreu no processamento da mensagem; seus valores possíveis são:

- ***noError (0)*** - Indica que não houve qualquer tipo de erro no processamento da mensagem.
- ***tooBig (1)*** - Se o tamanho da mensagem GetResponse gerada exceder uma limitação local, então o campo *error-status* assume este valor. Neste caso, o valor do campo *error-index* é zero.

- ***noSuchName (2)*** - Indica que o valor de alguma variável da lista de variáveis (*variablebindings*) não está disponível na MIB em questão. Neste caso, o valor do campo *error-index* indica em qual variável da lista ocorreu o problema.

- ***badValue (3)*** - Significa que o valor para uma determinada variável da lista não está de acordo com a linguagem ASN.1; ou que o tipo, tamanho ou valor é inconsistente. Assim como no caso anterior, o valor do campo *error-index* indica em qual variável da lista ocorreu o problema.

- ***readOnly (4)*** - Indica que determinada variável da lista só pode ser lida e não alterada. Neste caso, esse tipo de código de erro só é retornado após uma operação de Set sobre alguma variável. O valor do campo *error-index* indica em qual variável da lista ocorreu o problema.

- ***genErr (5)*** - Se o valor de uma determinada variável da lista não puder ser identificado ou alterado por outras razões que não as já citadas,

então o campo *error-status* assume o valor *genErr* ou simplesmente 5. Neste caso, o valor do campo *error-index* indica em qual variável da lista ocorreu o problema.

d) ***error-index*** - Quando o campo *error-status* é diferente de zero, este campo fornece uma informação adicional indicando qual variável da lista de variáveis causou a exceção (ou erro).

e) ***variable-bindings*** - Uma lista de nomes de variáveis e seus respectivos valores (em alguns casos, como no GetRequest PDU, esses valores são nulos).

Por se tratar de um caso particular de mensagem, indicando uma situação inesperada, a Trap PDU possui uma estrutura diferente das demais PDUs utilizadas pelo SNMP.

3.5.1. GETREQUEST PDU

A GetRequest PDU é utilizada pela estação de gerência SNMP para executar a operação Get, requisitando ao agente SNMP a leitura de variáveis da MIB da máquina onde ele se encontra. A entidade emissora inclui os seguintes campos nesta PDU:

Figura 7: Formato da GetRequest PDU.

PDU type	request-id	0	0	Variable-bindings

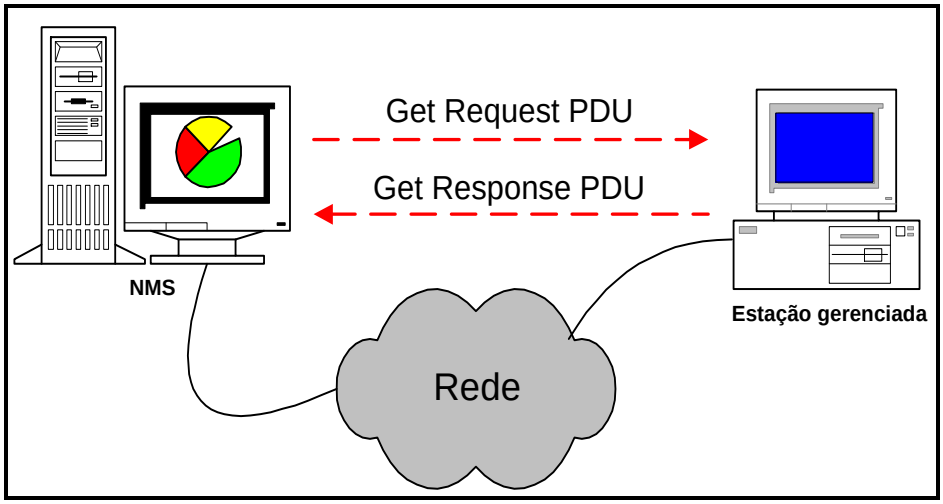
a) ***PDU type*** - Este campo indica que se trata de um GetRequest PDU.

b) ***request-id*** - A entidade emissora associa a este campo um número que identifica unicamente este *request*. O *request-id* possibilita à aplicação SNMP relacionar uma mensagem de response recebida a uma mensagem de *request* enviada, pois o response trará o mesmo valor do *request-id*. Além

disso, o *request-id* permite ao agente identificar PDUs duplicadas geradas graças a falhas no serviço de transporte.

c) ***variable-bindings*** - Uma lista de instâncias de objetos (variáveis) dos quais são requisitados os valores.

Figura 8: Passagem da GetRequest PDU.



Fonte: (Cohen, 2000)

A entidade SNMP receptora responde a uma GetRequest PDU com uma GetResponse PDU contendo o mesmo valor do *request-id*. A operação GetRequest é atômica: ou todas as variáveis requisitadas são recuperadas ou nenhuma é. Caso a entidade receptora da GetRequest PDU possa fornecer o valor de todas as variáveis listadas no campo *variablebindings*, então é montada uma GetResponse PDU contendo este campo acrescentado do respectivo valor de cada variável. Se o valor de apenas uma variável não puder ser informado, então nenhum dos valores das outras variáveis são retornados. As condições de erro que podem acontecer são (Oda, 1998) e (Cohen, 2000):

a) Para uma variável referenciada no campo *variable-bindings* não seja encontrada uma correspondente na MIB em questão; ou a variável referenciada pode ser um tipo agregado e desta forma não há uma valor agregado a esta instância. Neste caso, a entidade receptora retorna uma GetResponse PDU com o campo *error-status* indicando *noSuchName* e com o valor do campo error-index indicando que variável da lista de variáveis

(*variablebindings*) causou o problema, ou seja, se a terceira variável da lista de variáveis não estiver disponível para a operação Get, então o campo *error-index* da GetResponse PDU possui o valor 3.

a) A entidade receptora pode fornecer o valor de todas as variáveis da lista, mas o tamanho resultante da GetResponse PDU pode exceder a limitação local. Neste caso, a entidade receptora envia à entidade emissora uma GetResponse PDU com o campo *error-status* indicando *tooBig*.

b) A entidade receptora pode não fornecer o valor de uma determinada variável por alguma outra razão. Neste caso, a entidade receptora retorna uma GetResponse PDU com o campo *error-status* indicando *genErr* e com o valor do campo *error-index* indicando que variável da lista de variáveis causou o erro.

Se nenhum dos casos anteriores se aplicam, então a entidade receptora envia para a entidade da qual originou a GetRequest PDU uma GetResponse PDU com o campo *error-status* indicando *noError* e com o valor do campo *error-index* com valor zero.

3.5.2. GETNEXTREQUEST PDU

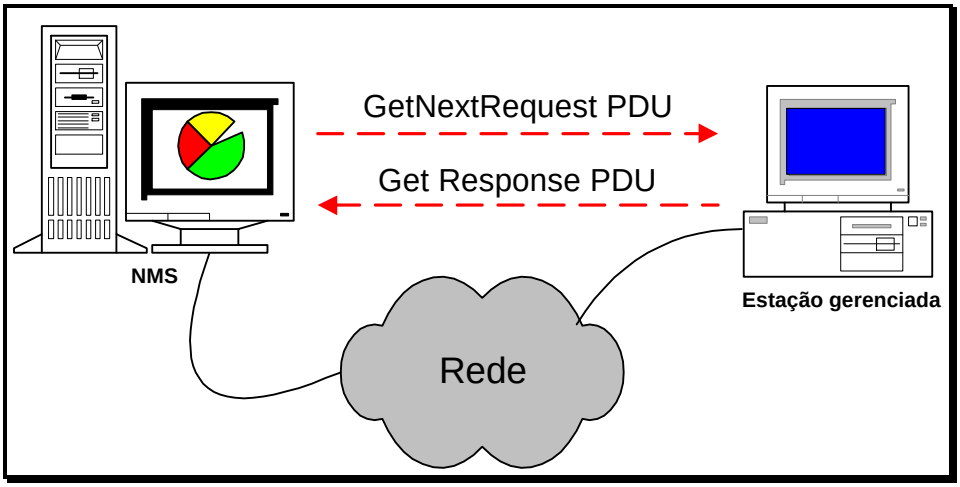
A GetNextRequest PDU é utilizada pela estação de gerência SNMP para executar a operação GetNext, requisitando ao agente SNMP a leitura da próxima variável na ordem lexicográfica da MIB da máquina onde ele se encontra. A GetNextRequest PDU é praticamente idêntica à GetRequest PDU, possuindo o mesmo mecanismo de troca e a mesma estrutura, como mostrado na Figura 8.

Figura 9: Formato da GetNextRequest PDU.

PDU type	request-id	0	0	Variable-bindings

A única diferença é o seguinte: numa GetRequest PDU cada variável descrita na lista de variáveis (*variablebindings*) deve ter seu valor retornado. Na GetNextRequest PDU, para cada variável, é retornado o valor da instância que é a próxima na ordem segundo a definição da MIB. Assim como a GetRequest PDU, a GetNextRequest PDU é atômica: ou todos os valores de todas as variáveis da lista (*variablebindings*) são retornados ou nenhum deles é.

Figura 10 Passagem da GetNextRequest PDU.



Fonte: (Cohen, 2000)

3.5.3. GETBULKREQUEST PDU

A GetBulk PDU é utilizada por um gerente SNMPv2 para executar a operação GetBulk, uma das melhores características adicionadas ao SNMP padrão, que permite que um gerente SNMPv2 resgate uma grande quantidade de informações de gerenciamento de uma determinada MIB. A GetBulkRequest PDU permite a um gerente SNMPv2 requisitar que a resposta (GetResponse PDU) seja tão grande quanto possível dentro da restrição de tamanho.

A GetBulkRequest PDU segue o mesmo princípio da GetNextRequest PDU, recuperando sempre a próxima instância, na ordenação da MIB, de cada variável da lista de variáveis (*variablebindings*). A diferença é que, com a GetBulkRequest PDU, é possível especificar quantas próximas instâncias na ordem devem ser retornadas na mesma GetResponse PDU.

Figura 11: Formato da GetBulkRequest PDU.

PDU type	Request-id	MaxRepeaters	NonRepeaters	Variable-bindings

Como mostrado na Figura 11, a GetBulkRequest PDU utiliza um formato diferente das demais PDUs, com os seguintes campos:

a) ***PDU type***, ***request-id*** e ***variable bindings*** - Estes campos possuem na GetBulkRequest PDU a mesma função que possuem nas demais PDUs (GetRequest PDU, GetNextRequest PDU, SetRequest PDU, Response PDU, InformRequest PDU e Trap PDU).

b) ***Nonrepeaters*** - Especifica o total das primeiras variáveis da lista de variáveis (*variablebindings*) das quais apenas a próxima instância na ordem deve ser retornada.

c) ***Max-repetitions*** - Especifica o número de sucessores, também na ordem, que devem ser retornados para o resto (*total_de_variáveis_da_lista* - *Nonrepeaters*) das variáveis da lista de variáveis.

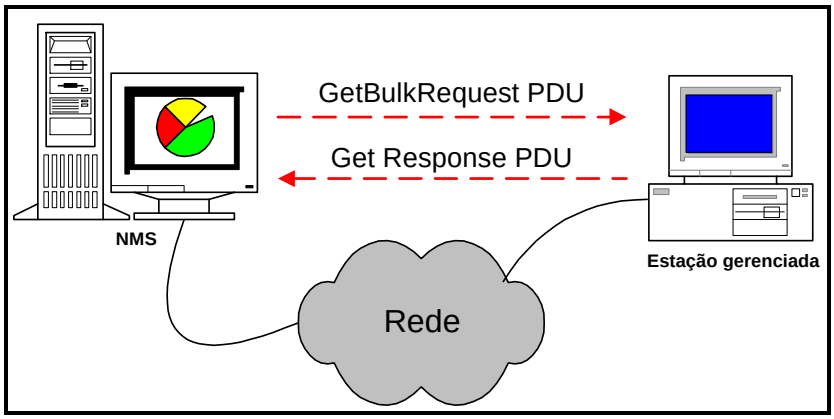
Após executar a operação GetBulk o agente deve construir uma GetResponse PDU, contendo as instâncias das variáveis requisitadas e seus respectivos valores. No entanto, pode acontecer de a GetResponse PDU não retornar todos os pares (instância, valor) requisitados na lista (podendo até retornar nenhum dos pares). Isso pode ocorrer por três razões:

a) Se o tamanho da mensagem que encapsula a GetResponse PDU exceder a limitação local de tamanho ou exceder o tamanho máximo de uma mensagem de *request* (definido pelo protocolo), então a GetResponse é gerada com um número menor de pares no campo *variablebindings*. Ou seja, a GetResponse PDU é gerada inserindo-se os pares até que o tamanho máximo seja alcançado.

b) Se todos os pares subsequentes descritos na lista de variáveis possuírem o valor *endOfMibView* (significando que não existe um sucessor da variável na ordem da MIB), o campo *variable-bindings* pode ser truncado neste ponto, retornando nenhum dos pares requisitados.

c) Se o processamento de uma GetBulkRequest PDU necessitar de uma grande quantidade de tempo de processamento pelo agente, então o agente pode particionar o processamento, retornando resultados parciais.

Figura 12: Passagem da GetBulkRequest PDU.



Fonte: (Cohen, 2000)

Se o processamento de alguma variável falhar, por qualquer motivo que não seja *endOfMibView*, então nenhum dos pares requisitados é retornado. Em vez disso, o agente envia ao gerente uma GetResponse PDU com o campo *error-status* indicando *genErr* e o valor do campo *error-index* apontando para a variável da lista que gerou o problema. O código de erro *tooBig* nunca é retornado por um agente SNMPv2 pois, como já foi dito, quando a mensagem de resposta é muito grande nem todas as variáveis requisitadas são retornadas, diminuindo seu tamanho.

3.5.4. SETREQUEST PDU

A SetRequest PDU é utilizada por um gerente SNMP na realização da operação Set, indicando ao agente que o valor de determinada(s) variável(is) deve(m) ser alterado(s). Este tipo de PDU possui o mesmo formato da GetRequest PDU. No entanto,

é utilizada para escrever uma variável na MIB, ao invés de lê-la. Diferente da GetRequest PDU e da GetNextRequest PDU, cujo valor das variáveis referenciadas na lista (*variablebindings*) têm valor zero, neste tipo de PDU o valor de cada instância da lista representa o valor a ser modificado na MIB em questão.

Figura 13: Estrutura da SetRequest PDU.

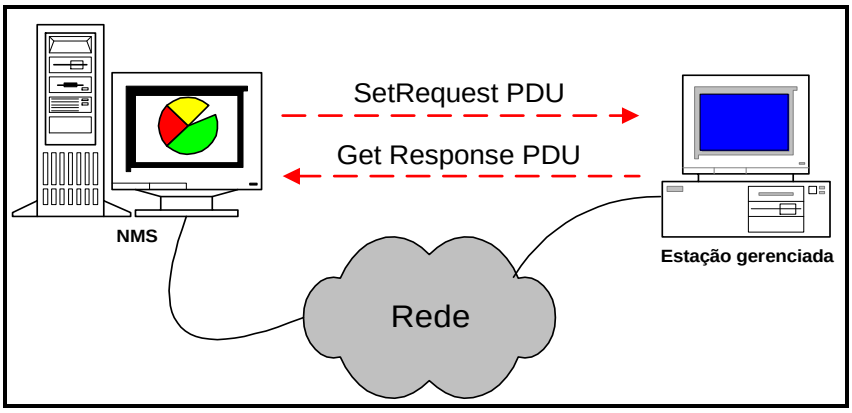
PDU type	request-id	0	0	Variable-bindings

Quando um agente recebe uma SetRequest PDU ele efetua, quando possível, a operação Set, modificando o valor das variáveis correspondentes, e retorna ao gerente da NMS emissora uma GetResponse PDU contendo o mesmo *request-id*. Assim como a operação Get, a operação Set é atômica: ou todas as variáveis são modificadas/atualizadas ou nenhuma delas é.

Se um agente que recebeu uma SetRequest PDU puder alterar o valor de todas as variáveis especificadas na lista (*variablebindings*), então ele constrói uma GetResponse PDU incluindo a mesma lista de variáveis, com seus respectivos valores atualizados, enviando-a ao gerente da NMS emissora. Se ao menos um dos valores não puder ser modificado, então nenhum valor é retornado e nem escrito na MIB em questão. As mesmas condições de erro usadas na GetRequest PDU podem ser retornadas (*noSuchName*, *tooBig*, *genErr*).

Outro erro que pode ocorrer durante a operação Set é o *badValue*. Esta condição de erro ocorre quando algum dos pares (variável, valor) contidos na SetRequest PDU estiver inconsistente no que diz respeito ao tipo, tamanho ou valor. Quando algum erro ocorre o agente descarta a GetResponse PDU e constrói uma nova com o campo *status-error* indicando o tipo de erro que ocorreu durante a operação e com o campo *index-error* apontando para a variável da lista que ocasionou o problema.

Figura 14: Passagem da SetRequest PDU.



Fonte: (Cohen, 2000)

Um curioso código de erro que pode ser retornado por um agente durante a execução da operação Set é o *readOnly*, que praticamente não é implementado. O problema é que o *readOnly* significa que uma determinada variável na MIB em questão cujo valor deveria ser modificado possui *status* de apenas leitura (*read only*), não podendo ser alterado. No entanto, quando um gerente pede a um agente para modificar uma variável que seja do tipo *read only*, o agente não podendo executar esta operação, constrói e envia uma GetResponse PDU com o código de erro indicando *noSuchName* e não *readOnly*.

O grande problema desta confusão é o fato de que quando um gerente recebe uma GetResponse PDU de uma operação de Set com o código de erro indicando *noSuchName*, ele não tem como saber se a instância da variável realmente não foi encontrada na MIB (o que justifica o *noSuchName*) ou se o problema, na verdade, é que a instância da variável possui *status read only* na MIB em questão.

Neste caso, para ter certeza da origem do problema, o gerente teria que enviar uma GetRequest PDU para a variável que causou o problema a fim de verificar se realmente a variável não existe na MIB ou se ela apenas possui o *status* de *read only*.

3.5.5. TRAP PDU

A Trap PDU é utilizada pelo agente SNMP na execução da operação Trap, notificando de forma assíncrona que eventos extraordinários tenham ocorrido no objeto

gerenciado pelo mesmo. A Trap PDU possui uma estrutura diferente das outras PDUs. Ela possui a estrutura mostrada na Figura 14.

Figura 15: Estrutura da Trap PDU - SNMPv1.

PDU type	enterprise	Agent-addr	generic-trap	Specific-trap	Time-stamp	variable-bindings

a) ***PDU type*** - indica o tipo de PDU, neste caso indica que trata-se de uma Trap PDU.

b) ***enterprise*** - indica o tipo do objeto que gerou o Trap; este campo é preenchido a partir do *sysObjectID*.

c) ***agent-addr*** - endereço do objeto que gerou o Trap.

d) ***generic-trap*** - indica o tipo do Trap; os valores possíveis para este campo são:

- ***coldStart (0)*** - Significa que a entidade emissora do Trap está sendo inicializada, de tal modo que a configuração do agente ou da entidade de protocolo podem ser alteradas;

- ***warmStart (1)*** - Significa que a entidade emissora do Trap está sendo reinicializada, de tal modo que nenhuma configuração do agente nem da entidade é alterada;

- ***linkDown (2)*** - Significa que a entidade emissora do Trap reconheceu que o estado de alguma de suas linhas de comunicação representadas na configuração do agente diminui;

- ***linkUp (3)*** - Significa que a entidade emissora do Trap reconheceu que o estado de uma de suas linhas de comunicação representadas na configuração do agente aumentou;

- ***authentication-Failure (4)*** - Significa que a entidade emissora do Trap foi o destinatário de uma mensagem de protocolo que não estava

corretamente autenticada. Quando da implementação do gerente, esta propriedade de enviar este tipo de Trap deve ser configurável, podendo em determinados casos ser desativada;

- ***egpNeighborLoss (5)*** - Significa que um vizinho EGP da entidade emissora do Trap foi o mesmo EGP de quem a entidade emissora marcou a queda e cujo relacionamento foi perdido;

- ***enterprise-Specific (6)*** - Significa que a entidade emissora do Trap reconhece que algum evento avançado específico ocorreu.

e) ***specific-trap*** - possui o código específico do Trap.

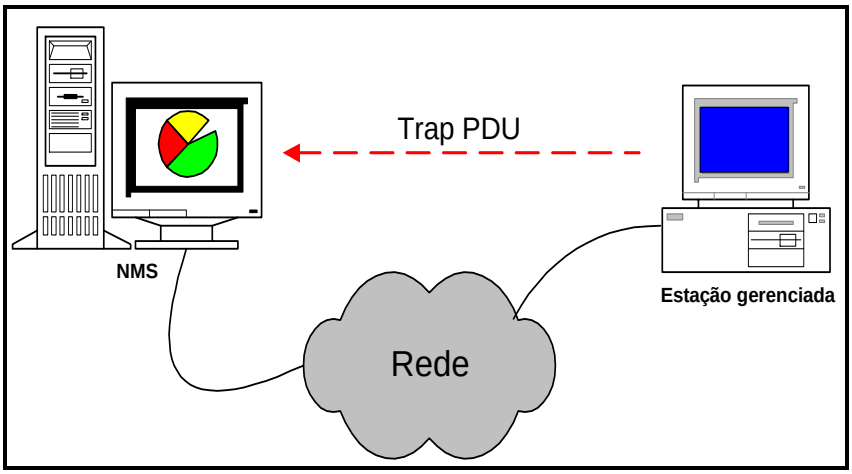
f) ***time-stamp*** - armazena o tempo decorrido entre a última reinicialização da entidade que gerou o Trap e a geração do Trap; contém o valor de *sysUpTime*.

g) ***variable-bindings*** - Uma lista de nomes de variáveis e seus respectivos valores.

Toda vez que uma máquina da rede é inicializada, o agente responsável deve enviar ao gerente um Trap do tipo *coldStart (0)*, indicando que a máquina "entrou" na rede.

Outra característica que difere a Trap PDU das outras PDUs é o fato de que ela não necessita de uma resposta, como mostra a Figura 16.

Figura 16: Passagem da Trap PDU.



Fonte: (Cohen, 2000)

3.5.6. TRAP PDU-V2

A Trap PDU-v2 segue as mesmas regras da Trap PDU utilizada na versão básica do SNMP, mas com um formato diferente. Este formato é o mesmo de todas as outras PDUs utilizadas no SNMPv2, exceto a GetBulkRequest PDU, o que facilita o processamento das mensagens pelo agente SNMPv2.

Figura 17: Estrutura da Trap PDU - SNMPv2

PDU type	request-id	0	0	Variable-bindings

.

3.5.7. INFORMREQUEST PDU

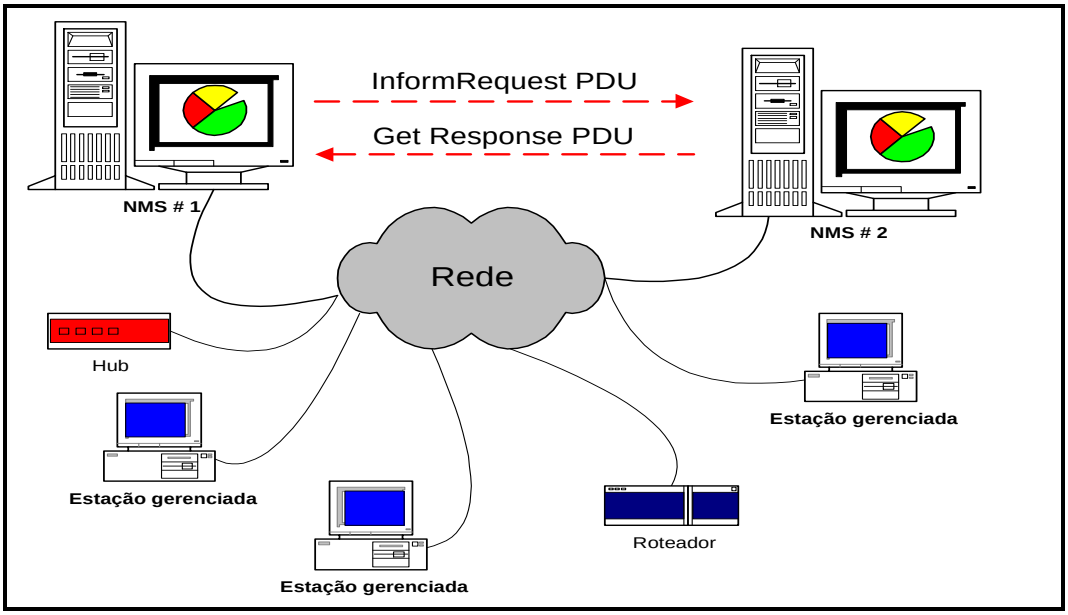
A InformRequest PDU é enviada por um gerente SNMPv2 a outro gerente SNMPv2, fornecendo informações de gerenciamento ao último gerente. Esta PDU inclui o campo *variable-bindings* com os mesmos elementos da Trap PDU-v2, como mostrado na figura 18.

Figura 18: Estrutura da InformRequest PDU.

PDU type	request-id	0	0	Variable-bindings

Ao receber uma InformRequest PDU, o gerente determina o tamanho da mensagem de resposta que encapsula uma GetResponse PDU com os mesmos valores dos campos *request-id*, *error-status*, *error-index* e *variable-bindings* da InformRequest PDU recebida. Se este tamanho exceder a limitação local de tamanho ou exceder o tamanho máximo da mensagem de resposta, então uma GetResponse PDU é construída com o código de erro (*error-status*) indicando *tooBig*, o índice de erro (*error-index*) com valor zero e com a lista de variáveis vazia (*variablebindings*).

Figura 19: Passagem da InformRequest PDU.



Fonte: (Cohen, 2000)

Caso o tamanho da mensagem seja aceitável, seu conteúdo é passado para a aplicação destino e é construída e enviada para o gerente origem uma GetRequest PDU com o código de erro indicando *noError* e o índice de erro com valor zero.

3.5.8. GETRESPONSE PDU

O formato da GetResponse PDU é idêntico à GetRequest PDU a não ser pela indicação do tipo (PDU Type). Uma GetResponse PDU é gerada por uma entidade SNMP sempre que esta recebe uma GetRequest PDU, GetNextRequest PDU ou SetRequestPDU, como descrito anteriormente.

4. DESENVOLVIMENTO DO PROTÓTIPO

Para viabilizar a implementação deste protótipo de software agente SNMP foi utilizada a linguagem de programação Delphi 4.0 para Windows, uma vez que os componentes auxiliares utilizados como o SNMP Tool da empresa Dart Communications utiliza componentes do Delphi 4.0, não sendo possível a utilização do ambiente de programação Delphi 3.0

Este ambiente de programação Borland Delphi 4.0 baseia-se na linguagem de Object Oriented Pascal (OOP). Para a visualização do conteúdo das MIBs utilizou-se do componente *MIB Browser* da empresa *MG-SOFT Corporation*

Os softwares modernos procuram ter uma interface mais simples, proporcionando ao usuário um melhor entendimento e facilidade de seu uso. Essas interfaces permitem o programador criar visualmente a interface com o usuário utilizando-se apenas do mouse, uma vez que as linguagens não orientadas a objetos, são construídas somente em linhas de código.

4.1. ESPECIFICAÇÃO DO PROTÓTIPO

Para a especificação do protótipo deste Trabalho de Conclusão de Curso foram utilizadas linguagens gráficas livres (fluxogramas), onde representam o funcionamento dos processos e funções deste protótipo de software agente SNMP.

As linguagens gráficas livres expressam seus processos e fluxos através de um conjunto de cadeias de símbolos alfanuméricos ou especiais. As informações e ações a serem tomadas podem ser vistas com maior rapidez e clareza através da sua representação precisa e detalhada, bem como, o entendimento dos processos pode ser visualizado passo a passo (Rekowsky, 1999).

As linguagens gráficas livres não estão baseadas em princípios, não tem uma forma especificamente definida, sua simbologia gráfica é arbitrária.

O protótipo de agente SNMP foi desenvolvido para informar ao gerente SNMP requisições da estação de gerenciamento, que pode ser o envio de informações de gerência ou ações sobre as variáveis do dispositivo onde está com atuação em elementos da rede.

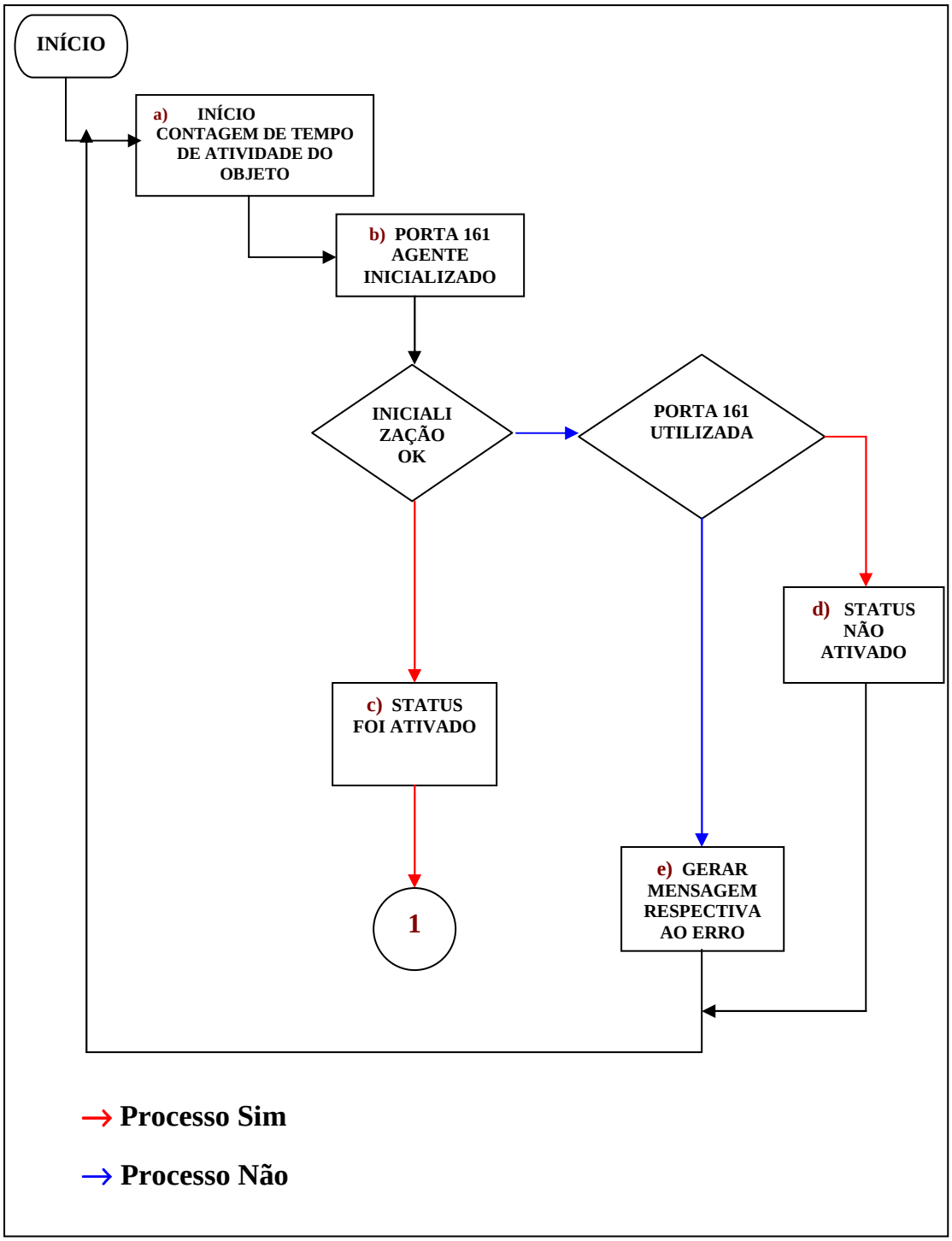
Para isto foi utilizado os componentes SNMP Tool da empresa Dart Communications, um gerente SNMP para que a comunicação entre eles fosse estabelecida.

Para um melhor entendimento dos processos que ocorrerão no protótipo seguem as descrições dos processos especificados.

4.1.1. INICIALIZAÇÃO DO AGENTE

Tem por finalidade mostrar a inicialização do módulo agente.

Figura 20: Inicialização do agente.



Descrição dos processos:

Processo a: Contagem feita quando o agente SNMP é inicializado (somente se estiver ativo).

Processo b: Porta padrão do protocolo SNMP.

Processo c: Agente inicializado (escutando) esperando requisições.

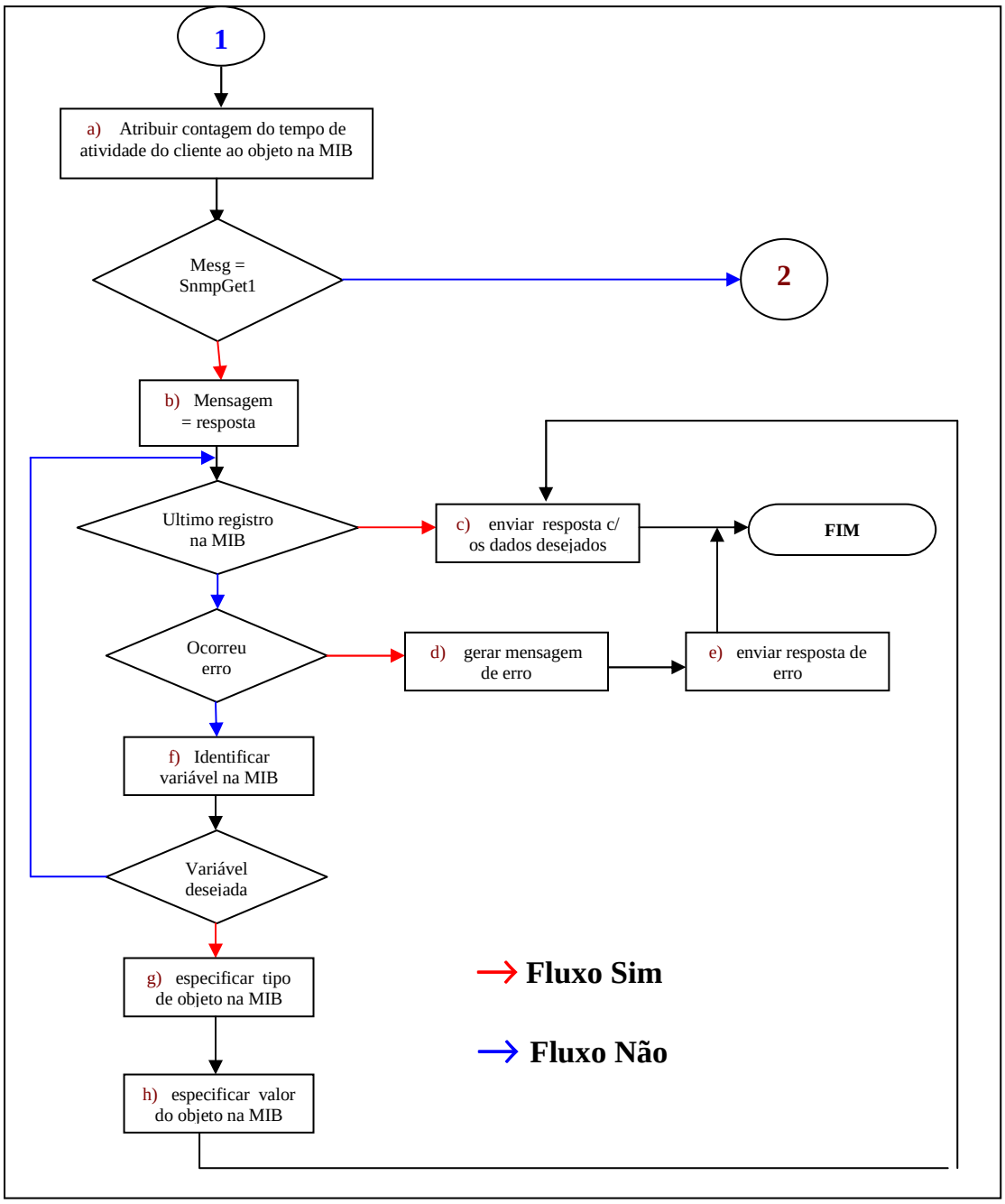
Processo d: Se a porta 161 estiver em uso, agente não é ativado.

Processo e: Gerada mensagem de erro ocorrido.

4.1.2. INTERPRETAR MENSAGEM RECEBIDA

Esta especificação corresponde a: Interpretar mensagem recebida, analisar o contexto do Objetos contidos na MIB, executar operação, gerar o erro, bem como, enviar a resposta com os dados desejados. A visualização destas ocorrências podem ser vistas nas figuras 21 e 22.

Figura 21: Interpretação da mensagem como ***snmpGet1***



Descrição dos processos:

Processo a: Contagem de tempo de atividade do cliente é inicializada.

Processo b: A mensagem atribuída é *snmpResponse* para poder enviar o retorno dos dados requisitados.

Processo c: Os dados são enviados para o gerente responsável pela estação monitorada.

Processo d: a ocorrência de erro é identificada para que o erro possa ser mostrado.

Processo e: O erro encontrado no processo d é mostrado.

Processo f: O objeto a ser identificado é comparado através de sua identificação (OID)na MIB do módulo cliente, verificando se este objeto está na MIB e pode ser monitorado.

Processo g: Quando identificado o objeto na MIB, é atribuído a ele seu tipo.

Processo h: É realizada a atribuição do valor do objeto no segmento gerenciado

Segue abaixo parte do código fonte da figura 21, interpretação da mensagem como snmpget1

```
procedure Tform_principal.Agent1Request(Sender: TObject);
var
  variavel : ISnmpVariable;
  i, ind: integer;
begin

   ObtemDadosMib;

   // so eh disparado quando o agente recebe uma requisicao do monitor
   // valor para sysUpTime
   varsysUpTime:=Agent1.Mib.Variables.Item('sysUpTime');
   varsysUpTime.value:=IntToStr(GetTickCount-TotalTickCount);
```

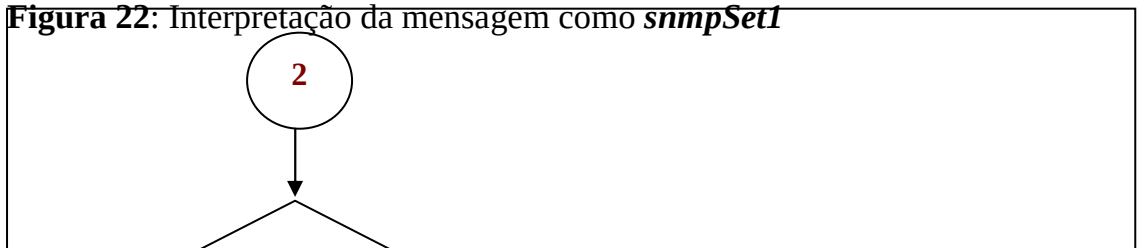
```
// Obtem valores para mensagem Get dos protocolos SNMP
if (Agent1.Message.Type_ = snmpGet1) then
```

```
begin
  i := 1;
  Agent1.Message.Type_ := snmpresponse1;
  for ind := 1 to Agent1.Message.Variables.Count do
    begin
      Variavel := Agent1.Message.Variables.Item(ind);
      if variavel.oid = '1.3.6.1.2.1.1.1.0' then
      begin
         Variavel.Type_ := snmpOctetString;
         Variavel.value:=  sysDescr.text;
      end
      else
      if variavel.oid = '1.3.6.1.2.1.1.2.0' then
      begin
         Variavel.Type_ := snmpOctetString;
         Variavel.value:= sysObjectID.text;
      end
      else
      if variavel.oid = '1.3.6.1.2.1.1.3.0' then
         begin
         Variavel.Type_ := snmpTimeTicks;
         variavel.value:= varsysUpTime.value;
        end
      else
      if variavel.oid = '1.3.6.1.2.1.1.4.0' then
      begin
         Variavel.Type_ := snmpOctetString;
         Variavel.value:= sysContact.text;
      end
      else
      if Variavel.oid = '1.3.6.1.2.1.1.5.0' then
      begin
         Variavel.Type_ := snmpOctetString;
         Variavel.value:= sysName.text;
      end
      else
      if Variavel.oid = '1.3.6.1.2.1.1.6.0' then
      begin
         Variavel.Type_ := snmpOctetString;
      end
      else
      if variavel.oid = '1.3.6.1.2.1.5.1.0' then
      begin
         Variavel.Type_ := snmpCounter32;
         Variavel.value:= EdICmpInMsgs.text;
      end
      else
      if variavel.oid = '1.3.6.1.2.1.5.14.0' then
      begin
         Variavel.Type_ := snmpCounter32;
         Variavel.value:= EdICmpOutMsgs.text;
      end
      else
```

```
      if variavel.oid = '1.3.6.1.2.1.4.3.0' then
      begin
```

```
        Variavel.Type_ := snmpCounter32;
        Variavel.value:= EdIPInReceives.text;
      end
      else
      if variavel.oid = '1.3.6.1.2.1.4.10.0' then
      begin
        Variavel.Type_ := snmpCounter32;
        Variavel.value:= EdIPOutRequests.text;
      end
      else
     if variavel.oid = '1.3.6.1.2.1.7.1.0' then
      begin
        Variavel.Type_ := snmpCounter32;
        Variavel.value:= EdUDPInDatagrams .text;
      end
      else
      if variavel.oid = '1.3.6.1.2.1.7.4.0' then
      begin
        Variavel.Type_ := snmpCounter32;
        Variavel.value:= EdUDPOutDatagrams.text;
      end
      else
      if variavel.oid = '1.3.6.1.2.1.6.10.0' then
      begin
        Variavel.Type_ := snmpCounter32;
        Variavel.value:= EdTCPInSegs.text;
      end
      else
      if variavel.oid = '1.3.6.1.2.1.6.11.0' then
      begin
        Variavel.Type_ := snmpCounter32;
        Variavel.value:= EdTCPOutSegs.text;
      end
      else
      begin
        Agent1.Message.Exception := snmpNoSuchName;
        Agent1.Message.ExceptionIndex := i;
         Agent1.Send;
         Exit;
      end;
      i:= i + 1;
    end;
    Agent1.Send;
    end;
```

Figura 22: Interpretação da mensagem como ***snmpSet1***



→ Fluxo Sim

→ Fluxo Não

Descrição dos processos:

Processo a: Mensagem atribuída como snmpResponse para que possa enviar o retorno dos dados requisitados.

Processo b: A ocorrência de erro é identificada para que o erro possa ser mostrado.

Processo c: O erro encontrado no processo b é mostrado.

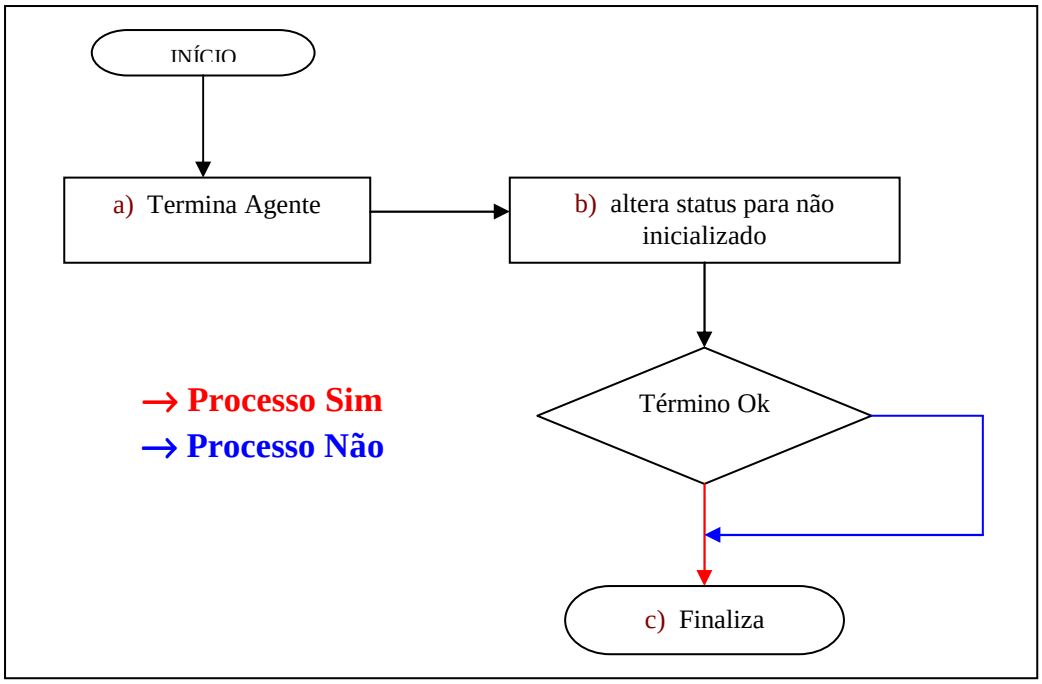
Processo d: O objeto a ser alterado é comparado através de sua identificação (OID) na MIB do agente, para ver se este objeto está contido na MIB e pode assim ser alterado.

Processo e: Se houver permissão de alteração, o valor do objeto local recebe o valor do objeto que foi alterado no módulo gerente.

Processo f: Quando identificado o objeto na MIB, é atribuído a ele seu tipo.

Processo g: Os dados requisitados são enviados para o gerente responsável pela estação gerenciada.

Figura 23: Término do módulo Agente.



Descrição dos processos:

Processo a: Termina o agente na porta padrão do protocolo snmp 161.

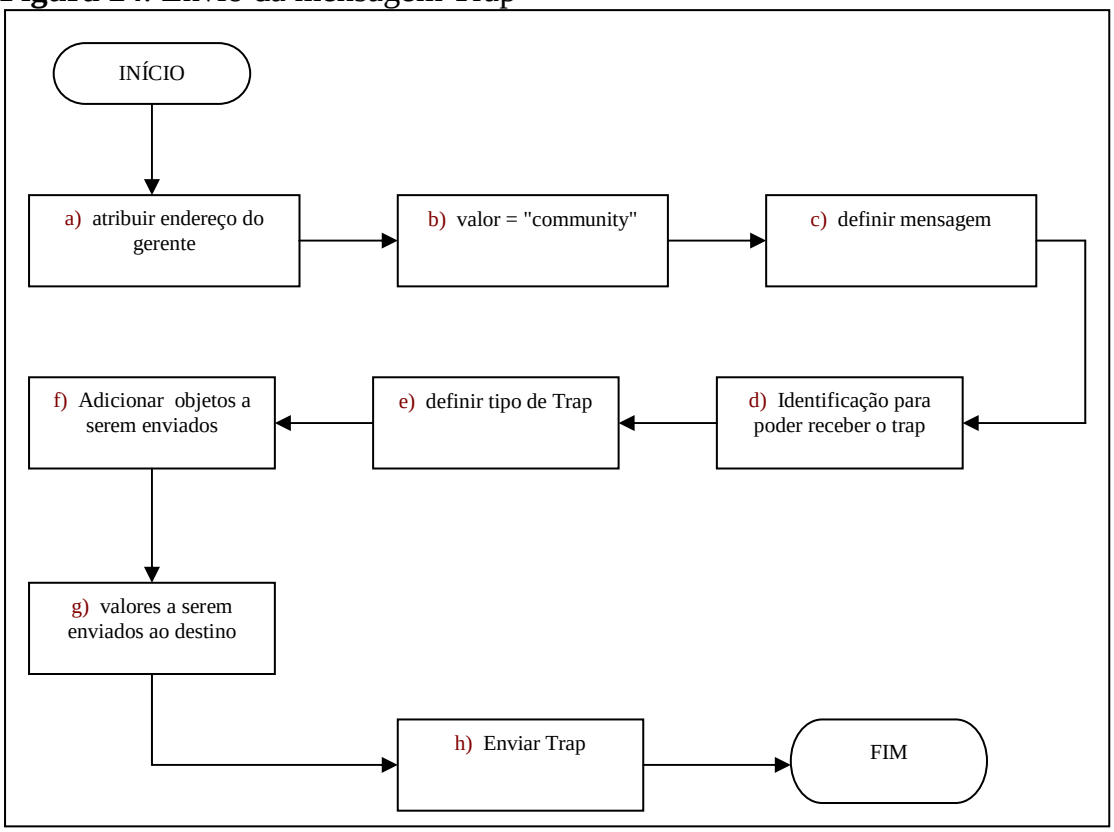
Processo b: Altera o status do agente de inicializado para não inicializado.

Processo c: Encerrar e fechar o programa.

4.1.3. ENVIO DA MENSAGEM TRAP

Através das mensagens de Traps o agente pode notificar o gerente que ocorreu alguma alteração ou outro evento em seu contexto da MIB. Sua visualização de como é composto pode ser vista no esquema abaixo representado pela figura 24.

Figura 24: Envio da mensagem Trap



Descrição dos Processos:

Processo a: Especificar o endereço de destino para o envio da mensagem Trap.

Processo b: Definir o valor para quem deve ter permissão de ver/receber a mensagem. Por default a mensagem é dada como public (pública) e todos podem recebe-la.

Processo c:Especificar a mensagem como ***snmpTrap1*** ou outro ***Trap*** correspondente.

Processo d: Especifica o receptor da mensagem enviada.

Processo e: Especificar o tipo de mensagem genérica do Trap.

Processo f: Os objetos que devem ser enviados recebem a atribuição de sua identificação.

Processo g: Os objetos especificados no processo anterior recebem a atribuição de seus valores.

Processo h: Envio do Trap para seu destino com os objetos e valores desejados.

Segue abaixo parte do código fonte da figura 24 do envio da mensagem Trap

```
procedure Tform_principal.btnTrapClick(Sender: TObject);
var
  variavel: ISnmpVariable;

begin
    Agent1.TrapManagers.Clear;
    Agent1.TrapManagers.Add(txtDest.Text,0);
    Agent1.Message.Reset;
    Agent1.Message.Community := 'public';
    Agent1.Message.Type_ := snmpTrap1;
    Agent1.Message.Enterprise := sysObjectID.Text;
    Agent1.Message.GenericTrap := snmpWarmStart;
```

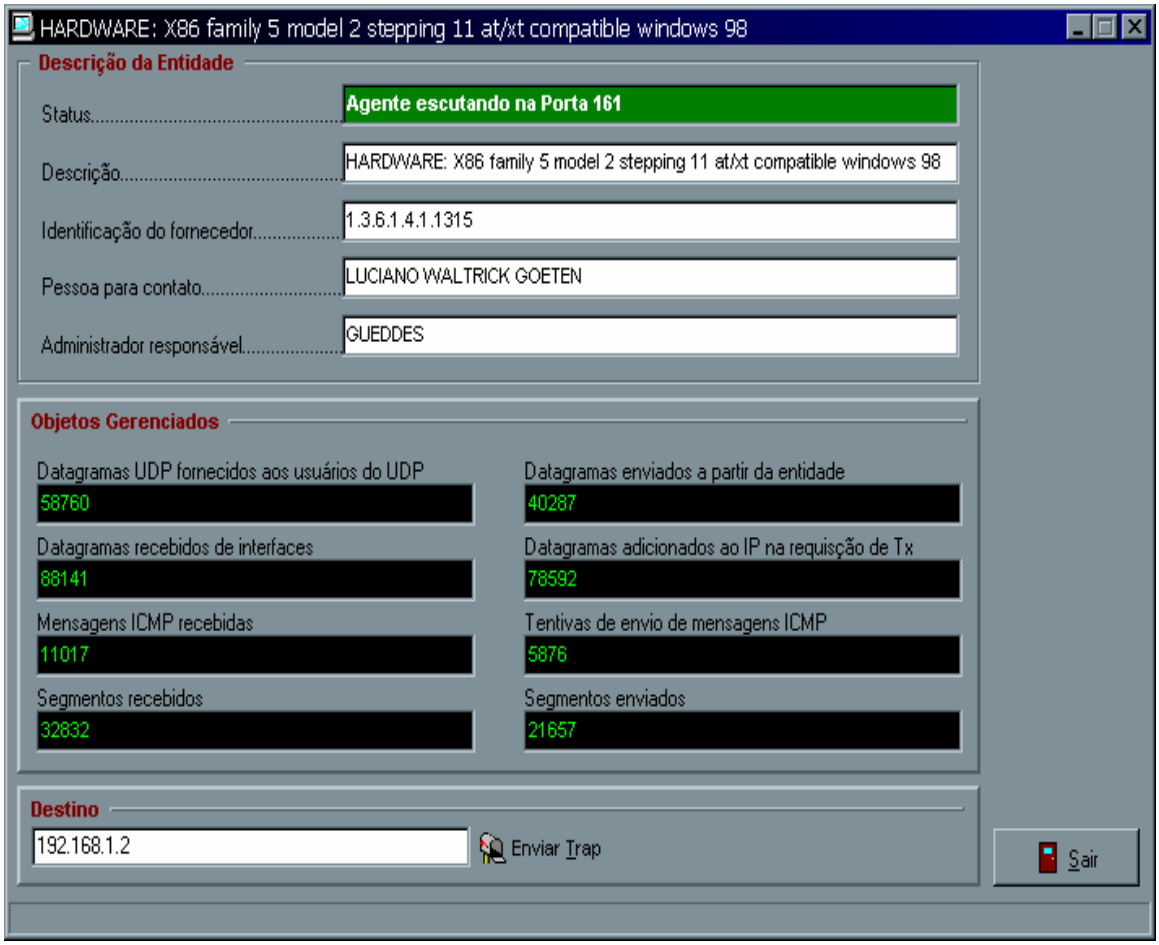
```
if trim(EdUDPInDatagrams.Text) <> '' then

     ObtemDadosMib;
   { sysDescr... }
   variavel:= CoSnmpVariable.Create;
   variavel.Oid:= '1.3.6.1.2.1.1.1.0';
   variavel.Type_  := snmpOctetString;
   Variavel.value  := Agent1.Mib.Variables.Item('sysDescr').Value;
   Agent1.Message.Variables.Add (variavel);

   { sysObjectID... }
   variavel:= CoSnmpVariable.Create;
   variavel.Oid:= '1.3.6.1.2.1.1.2.0';
   variavel.Type_  := snmpOctetString;
   Variavel.value  := Agent1.Mib.Variables.Item('sysObjectID').Value;
   Agent1.Message.Variables.Add (variavel);

   { sysUpTime... }
   variavel:= CoSnmpVariable.Create;
   variavel.Oid:= '1.3.6.1.2.1.1.3.0';
```

```
    variavel.Type_  := snmpTimeTicks;
    Variavel.value  := Agent1.Mib.Variables.Item('sysUpTime').Value;
    Agent1.Message.Variables.Add (variavel);

    { sysContact... }
    variavel:= CoSnmpVariable.Create;
    variavel.Oid:= '1.3.6.1.2.1.1.4.0';
    variavel.Type_  := snmpOctetString;
    Variavel.value  := Agent1.Mib.Variables.Item('sysContact').Value;
    Agent1.Message.Variables.Add (variavel);


    { sysName... }
    variavel:= CoSnmpVariable.Create;
    variavel.Oid:= '1.3.6.1.2.1.1.5.0';
    variavel.Type_  := snmpOctetString;
    Variavel.value  := Agent1.Mib.Variables.Item('sysName').Value;
    Agent1.Message.Variables.Add (variavel);

    { ipInReceives... }
    variavel:= CoSnmpVariable.Create;
    variavel.Oid:= '1.3.6.1.2.1.4.3.0';
    variavel.Type_  := snmpCounter32;
    Variavel.value  := EdIPInReceives.Text;
    Agent1.Message.Variables.Add (variavel);

    { icmpInMsgs... }
    variavel:= CoSnmpVariable.Create;
    variavel.Oid:= '1.3.6.1.2.1.5.1.0';
    variavel.Type_  := snmpCounter32;
    Variavel.value  := EdICmpInMsgs.Text;
    Agent1.Message.Variables.Add (variavel);

    { udpInDatagrams... }
    varUDPInDatagrams       :=  Agent1.Mib.Variables.Item('udpInDatagrams');
    varUDPInDatagrams.Oid   :=  '1.3.6.1.2.1.7.1.0';
    varUDPInDatagrams.Type_  :=  snmpCounter32;
    varUDPInDatagrams.value  :=EdUDPInDatagrams.text;
    Agent1.Message.Variables.Add (varUDPInDatagrams);

    { envia trap... }
    Agent1.Send;
end;
```

4.2. IMPLEMENTAÇÃO DO PROTÓTIPO

O módulo agente SNMP, após ativado com sucesso, monitora objetos contidos no contexto da MIB e, através de requisições do módulo gerente, manda informações dos objetos gerenciados para o gerente especificando o endereço da estação que fez a requisição, como mostrado na figura 25.

Figura 25: Tela principal do protótipo agente SNMP

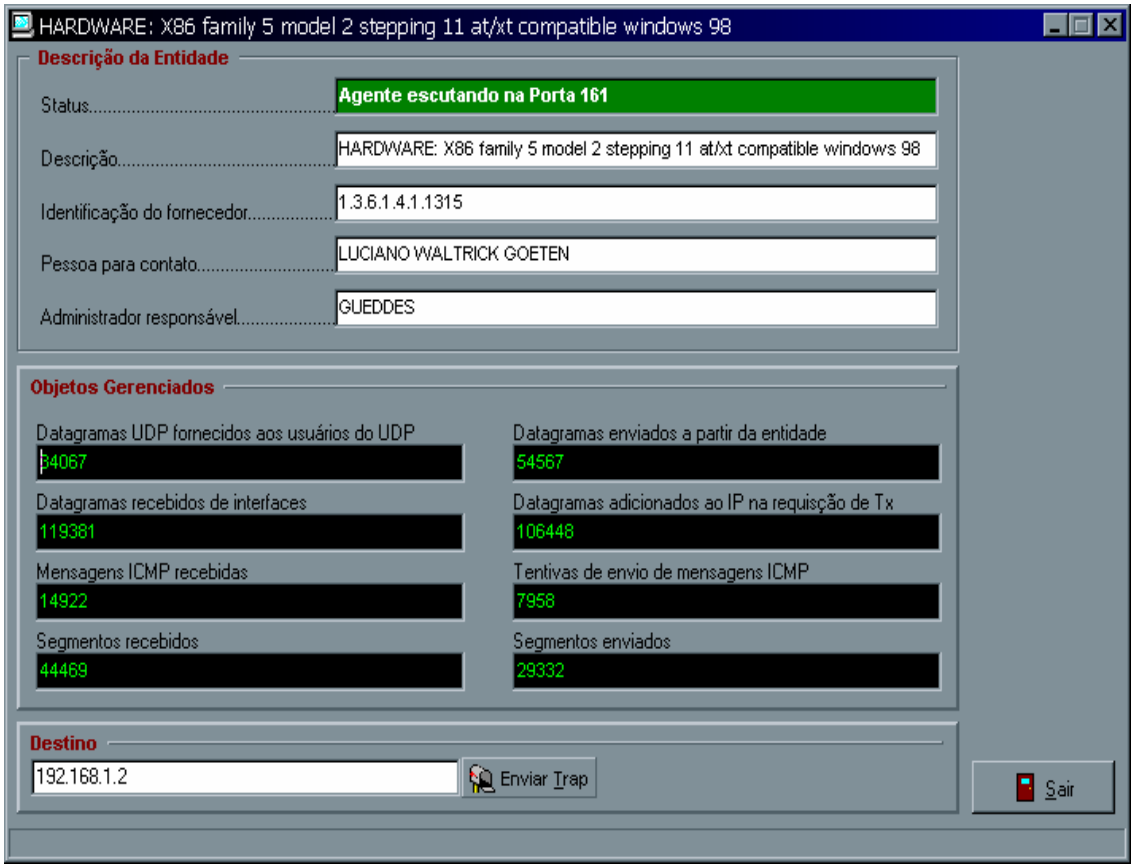


Segue abaixo parte do código fonte do protótipo agente mostrando seu funcionamento no momento em que é inicializado.

```
procedure Tform_principal.ValoresRegistro;
var
  defReg: TRegistry; // definições do registro
begin
    // pegar nome do Host do registro
    defReg := TRegistry.Create;
    defReg.RootKey := HKEY_LOCAL_MACHINE;
    if (defReg.OpenKey(gKey,true)) then
      begin
          // obter valores do registro para as variáveis
          // valor para sysDescr
          sysDescr.Text := defReg.ReadString('sysDescr');
          if sysDescr.Text = '' then
            sysDescr.Text := 'HARDWARE X86 family 5 model 2 Stteping 11 at/xt compatible windows 98';
          // valor para sysObjectID
          sysObjectID.Text := defReg.ReadString('sysObjectID');
          if sysObjectID.Text = '' then
            sysObjectID.Text := '1.3.6.1.4.1315';
          // valor para SysContact
          sysContact.Text := defReg.ReadString('sysContact');
          if sysContact.Text = '' then
            sysContact.Text := 'LUCIANO WALTRICK GOETEN';
          // valor para sysName
          sysName.Text := defReg.ReadString('sysName');
          if sysName.Text = '' then
            sysName.Text := 'GUEDDES';
      end;
      defReg.Free; // libera objeto da memória --> valores para as variáveis definidos
end;
```

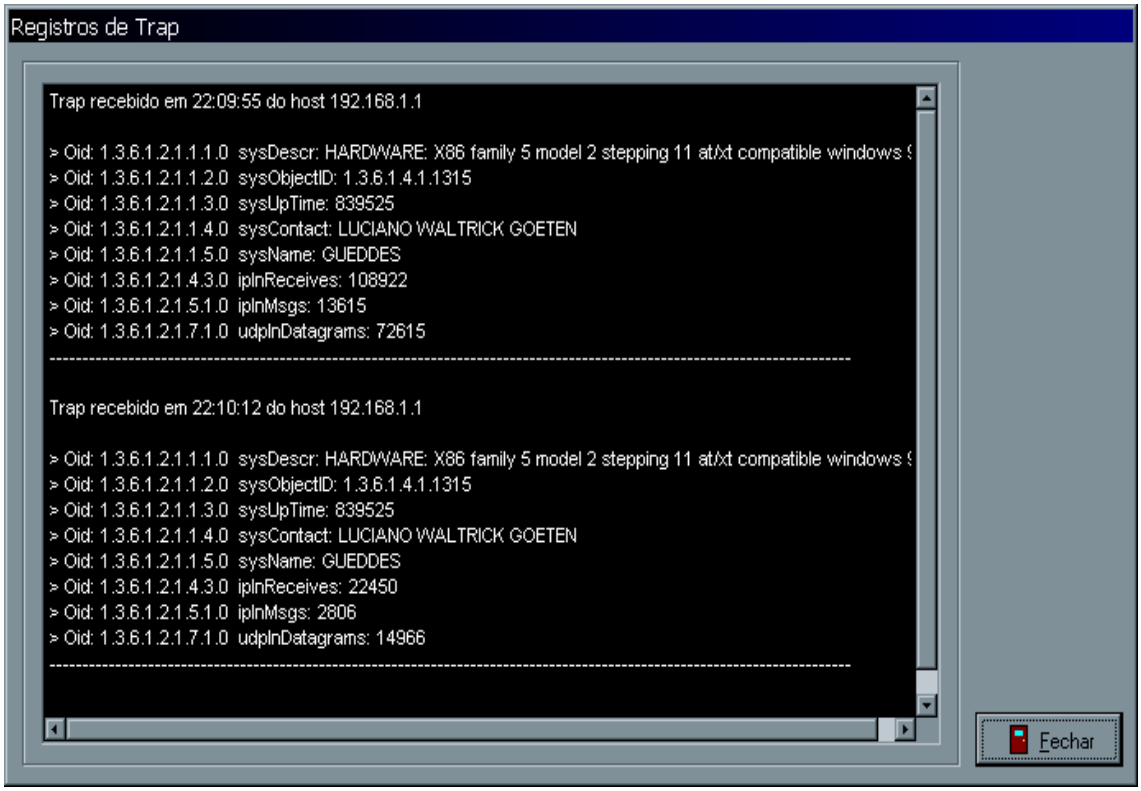
Através das mensagens de Traps o agente SNMP inicializado na porta virtual 161 pode notificar o gerente que ocorreu alguma alteração ou outro evento em seu contexto da MIB. A visualização do funcionamento do Trap é mostrada na figura 26, onde o mesmo monitora alguns objetos.

Figura 26: Envio do Trap



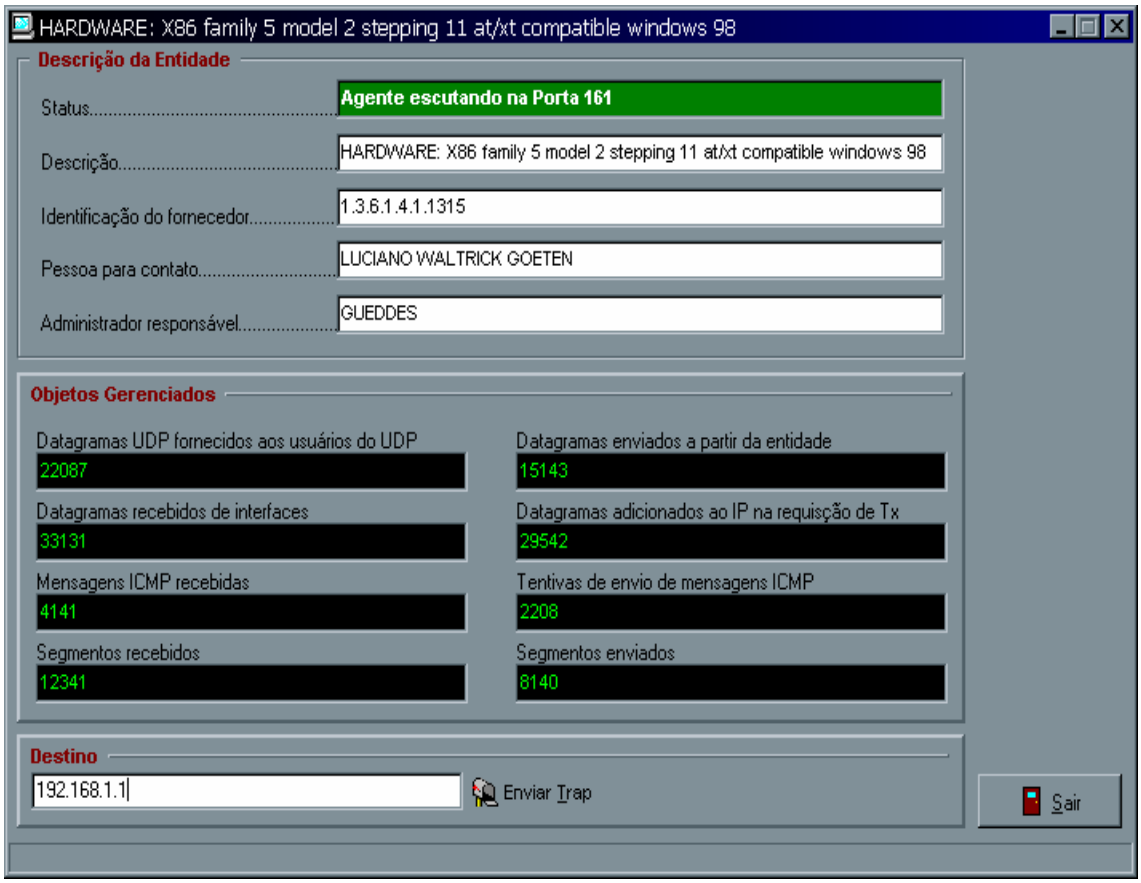
O módulo gerente, após, contactar agentes ativos na rede, pode analisar o contexto da MIB para ver se ocorreu alguma alteração de valores da variáveis contidas na MIB através do botão Visualizar Registros de Trap, como mostrado na figura 27.

Figura 27: Visualização de Registros de Trap


Registros de Trap
Trap recebido em 22:09:55 do host 192.168.1.1
> Oid: 1.3.6.1.2.1.1.1.0 sysDescr: HARDWARE: X86 family 5 model 2 stepping 11 at/xt compatible windows
> Oid: 1.3.6.1.2.1.1.2.0 sysObjectID: 1.3.6.1.4.1.1315
> Oid: 1.3.6.1.2.1.1.3.0 sysUpTime: 839525
> Oid: 1.3.6.1.2.1.1.4.0 sysContact: LUCIANO WALTRICK GOETEN
> Oid: 1.3.6.1.2.1.1.5.0 sysName: GUEDDES
> Oid: 1.3.6.1.2.1.4.3.0 ipInReceives: 108922
> Oid: 1.3.6.1.2.1.5.1.0 ipInMsgs: 13615
> Oid: 1.3.6.1.2.1.7.1.0 udpInDatagrams: 72615
Trap recebido em 22:10:12 do host 192.168.1.1
> Oid: 1.3.6.1.2.1.1.1.0 sysDescr: HARDWARE: X86 family 5 model 2 stepping 11 at/xt compatible windows
> Oid: 1.3.6.1.2.1.1.2.0 sysObjectID: 1.3.6.1.4.1.1315
> Oid: 1.3.6.1.2.1.1.3.0 sysUpTime: 839525
> Oid: 1.3.6.1.2.1.1.4.0 sysContact: LUCIANO WALTRICK GOETEN
> Oid: 1.3.6.1.2.1.1.5.0 sysName: GUEDDES
> Oid: 1.3.6.1.2.1.4.3.0 ipInReceives: 22450
> Oid: 1.3.6.1.2.1.5.1.0 ipInMsgs: 2806
> Oid: 1.3.6.1.2.1.7.1.0 udpInDatagrams: 14966
Fechar

Para encerrar o módulo agente SNMP, basta clicar no botão Sair na parte inferior direita da tela, que o mesmo irá finalizar como mostrado na figura 28.

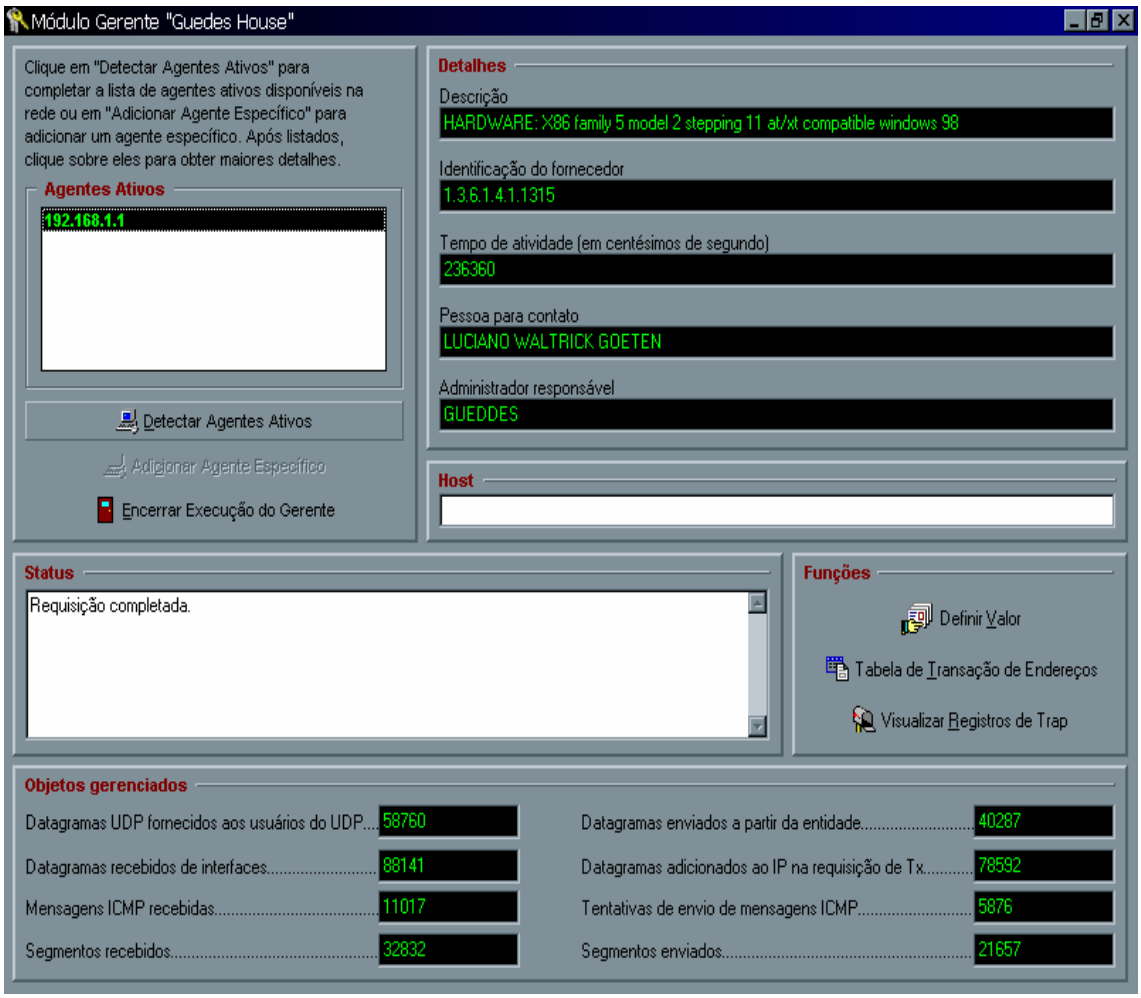
Figura 28: Encerrar módulo Agente



Inicialmente é inicializado o módulo gerente, para que o mesmo, possa contactar agentes ativos na rede.

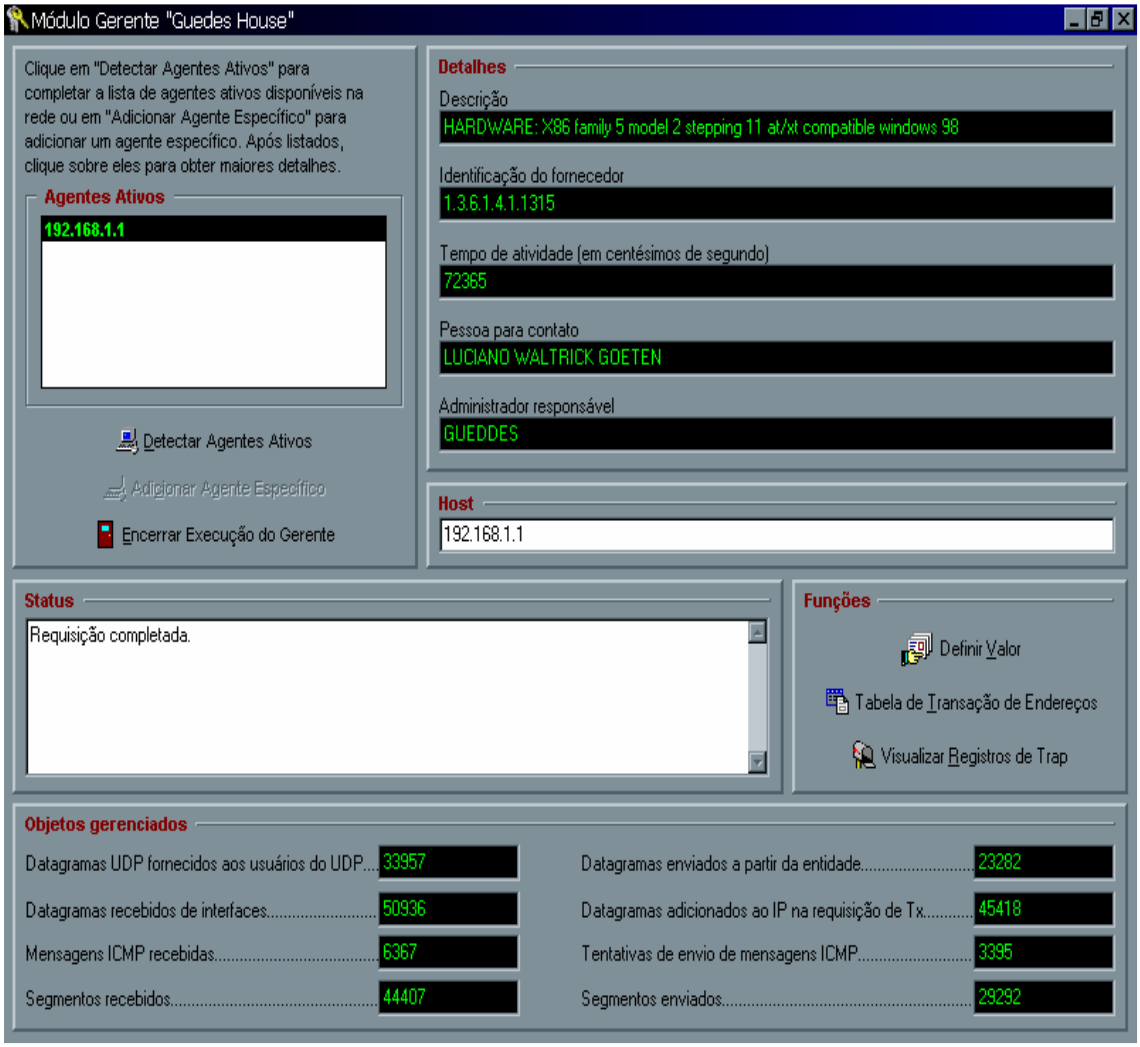
Após a detecção de agentes ativos na rede,outras funções tornam-se disponíveis na tela do módulo gerente, para que possam ser exploradas.

Figura 29: Detecção de agentes ativos na rede



Após serem detectados agentes ativos na rede, pode-se obter a passagem apenas de agentes específicos que estejam trafegando na rede.

Figura 30: Detecção de agentes específicos na rede.



5. CONCLUSÃO

Este TCC foi conseqüência de trabalho contínuo e disciplina no cumprimento de todas as etapas estabelecidas conforme o cronograma de trabalho apresentado e aprovado na proposta de TCC.

O Gerenciamento de Redes de Computadores é uma área com evolução contínua, mas é um tema pouco conhecido e difundido no meio acadêmico e profissional. Devido a pouca difusão do tema encontrou-se dificuldade no levantamento de material referente ao assunto proposto no TCC apresentado.

A especificação e implementação do protótipo foram as etapas que exigiram um conhecimento mais aprofundado das características do protocolo e operacionalidade do mesmo, bem como os conceitos envolvidos na gerência de redes.

Para a implementação do protótipo proposto neste TCC, foi utilizado o componente SNMP Tool da empresa Dart Communications (versão shareware), o qual, foi de fundamental importância para a confecção deste TCC. O componente utilizado oferece diversos recursos, como por exemplo, o acesso ao protocolo SNMP, que viabiliza a comunicação entre os módulos Gerente e Agente, bem como a compilação de MIBs, contento os objetos que devem ser gerenciados pelo módulo Gerente. Sem a utilização deste componente, ficaria inviável estabelecer a comunicação entre os módulos Gerente e Agente, uma vez que para que seu funcionamento ocorresse, seria necessário a implementação do protocolo SNMP e outras funções auxiliares para a concretização do mesmo.

Existem outros componentes que podem exercer a mesma funcionalidade para a comunicação utilizando o protocolo SNMP, como por exemplo a biblioteca WinSnmp, a qual, tem um enfoque maior para o módulo Gerente.

No propósito deste trabalho, foi de extrema importância o conhecimento sobre o funcionamento das Redes de Computadores no que diz respeito a Gerência de Redes,

uma vez que, o agente varre informações na MIB a fim de ler o contexto e mandar informações para a estação gerenciadora.

A implementação desde protótipo utilizando-se da linguagem Delphi na sua versão 4.0 oferece recursos e suporte à aplicações deste gênero, fornecendo uma interface de fácil interpretação com o usuário.

5.1. SUGESTÕES DE CONTINUIDADE

Como complemento para este trabalho, sugere-se uma implementação de um Agente SNMP para o Sistema Operacional Unix.

Uma outra alternativa seria, a implementação de um agente SNMP orientado a objetos.

6. REFERÊNCIAS BIBLIOGRÁFICAS

BRISA, Sociedade Brasileira para Interconexão de Sistemas Abertos **Arquitetura de redes de computadores OSI e TCP/IP**. São Paulo: Makron Books; Rio de Janeiro: , 1994.

BRISA, Sociedade Brasileira para Interconexão de Sistemas Abertos. **Gerenciamento de Redes**: uma abordagem de sistemas abertos. Makron Books do Brasil, 1993.

CASE, J. et al **A simple network management protocol (SNMP)**, Network Working Group, RFC1157 Request for Comments:1157,[s. l.]maio,1990. disponível em: <http://gxsnmp.org/cgi-bin/getrfc?1157>. Acesso em: 28 mar. 2001.

CHISANE, Fabrício. **Gerenciamento de redes SNMP**. 03/2001. Disponível em < http://www.inf.ufrgs.br/~chisane >. Acesso em mar. 2001.

COHEN, Yoran. **SNMP simple network management protocol**. 02/2000. Disponível em < http://www-dos.uniinc.msk.ru/tech1/1995/snmp/snmp.htm >. Acesso em mar. 2001

MAFISNKI, André. **Protótipo de software de gerência SNMP para o ambiente Windows NT**. 1999, 58 f. Trabalho de Conclusão de Curso (Bacharelado em Ciências da Computação) – Centro de Ciências Exatas e Naturais, Universidade Regional de Blumenau, Blumenau.

MEIRELES, L.F.T. **Introdução ao modelo de gerência SNMP,** Pelotas, Set, 1999. disponível em: <http://redes.ucpel.tche.br/documentos/snmp>. Acesso em 28 mar. 2001.

MELLO, Jorge Lucas. **Protótipo de um agente SNMP para uma rede local utilizando a plataforma JDMK**. 2000, 88 f. Trabalho de Conclusão de Curso (Bacharelado em Ciências da Computação) - Centro de Ciências Exatas e Naturais, Universidade Regional de Blumenau, Blumenau.

ODA, Cybelle Suemi. **Gerenciamento de redes de computadores**. 04/2001. Diponível em < http://www.gt-er.cg.org.br/operacoes/gerencia-redes >. Acesso em 04/2001

REISDORPH, Kent. Aprenda em 21 dias Delphi 4. Rio de Janeiro: Cmpus, 1999.

REKOWSKY, Ricardo Henrique. **Protótipo de software para a monitoração de desempenho de redes, utilizando o RMON**. 1999, 88 f. Trabalho de Conclusão de Curso (Bacharelado em Ciências da Computação) – Centro de Ciências Exatas e Naturais Universidade Regional de Blumenau, Blumenau.

RIGNEY, Steve. **Planejamento e Gerenciamento de Redes**. Rio de Janeiro: Campus, 1996.

SOARES, Luiz Fernando Gomes. **Redes de computadores**: das LANs, MANs e WANs às redes de computadores. Rio de Janeiro: Campus, 1995.

STALLINGS, W. **SNMP, SNMP v2 and RMOM**. First Edition. Addison Wesley, 1996.